Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

JESCO HUMPOLA, ARMIN FÜGENSCHUH, THOMAS LEHMANN

A Primal Heuristic for MINLP based on Dual Information

ZIB-Report 13-49 (November 2013)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

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

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

A Primal Heuristic for MINLP based on Dual Information

Jesco Humpola[1], Armin Fügenschuh[2], and Thomas Lehmann[1]

[1] Zuse Institute Berlin,
Takustr. 7, 14195 Berlin, Germany
{humpola,lehmann}@zib.de
http://www.zib.de

[2] Helmut Schmidt University / University of the Federal Armed Forces Hamburg,
Holstenhofweg 85, 22043 Hamburg, Germany
fuegenschuh@hsu-hh.de
http://am.hsu-hh.de

Abstract. We present a novel heuristic algorithm to identify feasible solutions of a mixed-integer nonlinear programming problem arising in natural gas transportation: the selection of new pipelines to enhance the network's capacity to a desired level in a cost-efficient way. We solve this problem in a linear programming based branch-and-cut approach, where we deal with the nonlinearities by linear outer approximation and spatial branching. At certain nodes of the branching tree, we compute a KKT point for a nonlinear relaxation. Based on the information from the KKT point we alter some of the integer variables in a locally promising way. We describe this heuristic for general MINLPs and then show how to tailor the heuristic to exploit our problem-specific structure. On a test set of real-world instances, we are able to increase the chance of identifying feasible solutions by some order of magnitude compared to standard MINLP heuristics that are already built in the general-purpose MINLP solver SCIP.

Keywords: Mixed-Integer Nonlinear Programming; Relaxations; Heuristics; Duality; Nonlinear Network Design Applications.

1 Introduction

After several decades of progress, the numerical solution of linear programming problems (LP) nowadays is a straight-forward issue. LPs with up to several millions of variables and constraints can routinely be solved to optimality. For a historical survey on the development of LP solvers we refer to [34, 9, 10]. The computational steps that need to be carried out in an efficient implementation of such solver are well-known and documented in the literature [42, 28]. Among others, these include the dual simplex method, a bound-flipping ratio test, dual steepest edge pricing, LU factorization and update, LP preprocessing, and the exploitation of sparsity in the constraint matrix, in order to obtain a fast code. Further, one has to deal with numerical stability issues and degeneracy, in order to obtain a stable implementation.

In contrast, the numerical solution of general mixed-integer linear programming problems (MILP) requires "a whole bag of tricks" (Bob Bixby, private communication). The main ingredients are presolving, heuristics, cutting planes, branching rules (including both, variable and node selection), and solving LPs as a subroutine. For a survey on MILP techniques we refer to [33, 22].

In recent years the interest has extended to also solving mixed-integer nonlinear problems (MINLPs) to proven global optimality. Solvers such as SCIP [1] that started as MILP (and constraint programming) solvers offer now also the ability to solve certain types of MINLPs [8]. SCIP first approximates the nonlinear functions by linear inequality constraints. This initial approximation is adaptively refined during the branch-and-cut process by adding further cutting planes (for convex nonlinear inequality restrictions), or spatial branching (for nonconvex inequality restrictions). Besides these features that deal with the nonlinear parts of the given model, all classical MILP techniques mentioned before are applied in the solution process. For more details we refer to [37–39]. Other MINLP solvers besides SCIP that make use of these ideas (and others) are, for example, Antigone [20], Baron [40], Bonmin [13], Couenne [5], or Minotaur [31].

In this work we focus on heuristics for MINLPs. Historically, heuristics were first developed for MILP solving. Examples include "simple" rounding strategies [4, 3], feasibility pumps [18, 2, 19], RINS [16], its

variant RENS [6], and evolutionary metaheuristics [36], to name just a few of today's most successful strategies for finding good feasible solutions at the beginning of the solution process, as well as later in the branching phase. Recently, several of the MILP heuristics were adapted for the solution of general MINLPs, for example, rounding strategies [32], the feasibility pump [11, 12, 15], or specially developed for MINLPs, for example [29, 30]. For a further discussion we refer to the survey [7].

1.1 Our Contribution

We describe a new heuristic for MINLPs that is based on dual information. Dual information is gathered from the dual linear program corresponding to the current (primal) linear programming relaxation, or in general, from the Lagrange multipliers of a primal-dual solution fulfilling the Karush-Kuhn-Tucker (KKT) conditions. Similar to the restrict-and-relax search for 0-1 mixed-integer programs from Guzelsoy et al. [25], our heuristic also relaxes and unfixes previously fixed variables, and is able to make use of additional problem-specific information.

We show that our heuristic is able to identify high-quality feasible solution for the problem of extending the topology of a given gas network. The flow of gas inside the pipelines is described by a nonlinear functional equation. Since the desired flow of gas exceeds the network's capacity, it is necessary to extend the network's topology by adding further pipelines. The construction cost for these additional pipes should be minimal. This problem can be formulated as a constraint integer program, more precisely an MINLP with indicator constraints. When applying standard solvers, such as SCIP, to instances of realistic size (i.e., graphs having several 100 nodes and arcs), no feasible solution is found within a reasonable timeframe. Our heuristic is able to find feasible solutions for at least some of the instances. We then demonstrate how problem specific knowledge can be exploited in order to fine-tune the heuristic. Finally, we show that this tuned heuristic is much more capable in identifying feasible solutions.

2 The General Heuristic

We consider a mixed-integer non-linear optimization problem with indicator constraints of the following form:

$$\begin{aligned}
\min \quad & f(x,y,z) && && \text{(2.1a)}\\
\text{s.t.} \quad & g(x,y,z) \le 0 && && \text{(2.1b)}\\
& h_i(x,y) - \Delta_i \le 0 && (i \in I) && \text{(2.1c)}\\
& z_i = 1 \Rightarrow \Delta_i = 0 && (i \in I) && \text{(2.1d)}\\
& \underline{x} \le x \le \overline{x} && && \text{(2.1e)}\\
& \underline{y} \le y \le \overline{y} && && \text{(2.1f)}\\
& x \in \mathbb{R}^n && && \text{(2.1g)}\\
& y \in \mathbb{Z}^m && && \text{(2.1h)}\\
& \Delta \in \mathbb{R}^I_+ && && \text{(2.1i)}\\
& z \in \{0,1\}^I, && && \text{(2.1j)}
\end{aligned}$$

with $m, n \in \mathbb{N}$, and a finite, nonempty index set $I = \{1, 2, \ldots\}$ with cardinality $|I|$. Let $f, g : \mathbb{R}^n \times \mathbb{Z}^m \times \{0,1\}^I \to \mathbb{R}$ and $h_i : \mathbb{R}^n \times \mathbb{Z}^m \to \mathbb{R}$ (for $i \in I$) be $\mathcal{C}^1$ (continuously differentiable) functions. These functions depend on continuous variables $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ integer variables $y = (y_1, \ldots, y_m) \in \mathbb{Z}^m$ and binary variables $z = (z_1, \ldots, z_{|I|}) \in \{0,1\}^I$. The variables x are bounded from below and above by $\underline{x}, \overline{x} \in \mathbb{R}^m$ and y are bounded from below and above by $\underline{y}, \overline{y} \in \mathbb{Z}^m$, respectively. The binary variables z represent the decision whether the constraint $h_i(x,y) \le 0$ is active (for $z_i = 1$) or inactive (for $z_i = 0$). A constraint h_i is inactive, if and only if a slack variable Δ_i is positive, where $\Delta = (\Delta_1, \ldots, \Delta_{|I|}) \in \mathbb{R}^I_+$ denotes the vector of continuous slack variables. These constraints (2.1d) are called indicator constraints.

2.1 Theoretical Motivation

As described in Section 1, we solve problem (2.1) in a linear programming based branch and cut framework. Now let us consider a node of this branching tree. In the following we describe our heuristic that

complements the solution process. The heuristic is invoked if the current LP relaxation has integer values for all integer and binary variables (y and z).

When the heuristic is invoked, there exists an index set $I' \subseteq I$ (potentially empty) such that the binary variables z_i are fixed to 1 for all $i \in I'$. The fixation of the binary variables is denoted by z^* and the fixation of the integer variables by y^*. We introduce two vectors of slack variables $s^+, s^- \in \mathbb{R}^n_+$ and consider the following NLP relaxation of (2.1):

$$\min\ \sigma f(x, y^*, z^*) + (1-\sigma)\sum_{j=1}^{n}(s_j^+ + s_j^-) + (1-\sigma)\sum_{i\in I'}\Delta_i \tag{2.2a}$$

$$\text{s.t.}\quad g(x, y^*, z^*) \le 0 \tag{2.2b}$$

$$h_i(x, y^*) - \Delta_i \le 0 \qquad (i \in I') \tag{2.2c}$$

$$x - s^+ \le \overline{x} \tag{2.2d}$$

$$x + s^- \ge \underline{x} \tag{2.2e}$$

$$x \in \mathbb{R}^n \tag{2.2f}$$

$$\Delta \in \mathbb{R}^{I'}_+ \tag{2.2g}$$

$$s^{\pm} \in \mathbb{R}^n_+ \tag{2.2h}$$

The parameter $\sigma \in [0, 1]$ in the objective function (2.2a) controls the compromise between improving feasibility and optimality. A lower value of σ puts a higher emphasis on feasibility.

We denote the objective function in (2.2a) by $\tilde{f}(x, y^*, z^*, s, \Delta)$. The term

$$(1-\sigma)\sum_{j=1}^{n}(s_j^+ + s_j^-) + (1-\sigma)\sum_{i\in I'}\Delta_i \tag{2.3}$$

is called the slack part of the objective function. We consider a local optimum of (2.2) with primal solution $(\hat{x}, \hat{s}, \hat{\Delta})$ and dual solution $(\hat{\lambda}, \hat{\eta}, \hat{\mu})$. Our goal is to apply equation (5.4) from Fiacco and Ishizuka [17], which states a relation between a modification of a constraint and its impact on the objective function value via Lagrange duals from a corresponding KKT point. In the following, we briefly outline the main steps of their derivation.

We consider problem (2.2) as a parametric nonlinear optimization problem of the general form

$$\min_r\ F(r, p), \tag{2.4a}$$

$$G_i(r, p) \le 0, \quad i \in I, \tag{2.4b}$$

where $r \in \mathbb{R}^{N_n}$ are variables and $p \in \mathbb{R}^I$ are parameters. We note that $r = (x, s, \Delta)$ and $p = 0$ for problem (2.2). Assume that F and G are twice continuously differentiable, real valued functions. A KKT point $(\hat{r}, \hat{\eta})$ of (2.4) with Lagrangian multipliers $\eta \in \mathbb{R}^{|I|}$ depends on the chosen parameter p, hence we denote it $(\hat{r}(p), \hat{\eta}(p))$, in order to emphasize this dependency. We select a parameter p_0 and obtain by solving (2.4) a solution $\hat{r} = \hat{r}(p_0)$. In order to estimate the change in the objective function when altering the parameter p in a small neighborhood of p_0, we differentiate the objective function $F(\hat{r}(p), p)$ with respect to p and obtain:

$$\frac{dF}{dp}(\hat{r}, p_0) = \nabla_r F(\hat{r}, p_0)\frac{dr}{dp}(p_0) + \frac{\partial F}{\partial p}(\hat{r}, p_0). \tag{2.5}$$

For a KKT point the gradient of the objective function is a linear combination of the gradient of the active constraints. Hence we can rewrite the first summand on the right-hand side of (2.5) as

$$\nabla_r F(\hat{r}, p_0) = \eta \nabla_r G(\hat{r}, p_0),$$

where Lagrangian multipliers η are the coefficients of the linear combination. We use this to rewrite (2.5) as

$$\frac{dF}{dp}(\hat{r}, p_0) = \eta \nabla_r G(\hat{r}, p_0)\frac{dr}{dp}(p_0) + \frac{\partial F}{\partial p}(\hat{r}, p_0). \tag{2.6}$$

We denote by $I' \subseteq I$ the subset of active constraints from I for $(\hat{r}, p_0)$, i.e., $G_i(\hat{r}, p_0) = 0$ for $i \in I'$. By Fiacco and Ishizuka [17] there exists a neighborhood of p_0 such that $G(\hat{r}(p), p) = 0$ for all p within this p_0-neighborhood. This means, the function $G(\hat{r}(\cdot), \cdot)$ is locally constant, which implies that $\frac{d}{dp}G(\hat{r}(p), p) = 0$ for all p in the neighborhood of p_0. We compute this derivation in p_0 and obtain

$$0 = \frac{d}{dp}G(\hat{r}, p_0) = \nabla_r G(\hat{r}, p_0)\frac{dr}{dp}(p_0) + \frac{\partial}{\partial p}G(\hat{r}, p_0). \tag{2.7}$$

We put (2.7) into (2.6) and obtain

$$\frac{dF}{dp}(\hat{r}, p_0) = -\eta \nabla_p G(\hat{r}, p_0) + \frac{\partial F}{\partial p}(\hat{r}, p_0).$$

The Lagrange function of (2.4) is $L(r, \eta, p) = F(r, p) - \eta G(r, p)$. We consider its derivation with respect to p:

$$\frac{dL}{dp}(\hat{r}, \eta_0, p_0) = \underbrace{\nabla_r L(\hat{r}, \eta_0, p_0)}_{=0, \text{by KKT}}\frac{dr}{dp}(p_0) + \underbrace{\nabla_\eta L(\hat{r}, \eta_0, p_0)}_{=G(\hat{r}, \eta_0, p_0)=0}\frac{d\eta}{dp}(p_0) + \frac{\partial L}{\partial p}(\hat{r}, \eta_0, p_0).$$

From

$$\frac{\partial L}{\partial p}(\hat{r}, \eta_0, p_0) = -\eta \nabla_p G(\hat{r}, p_0) + \frac{\partial F}{\partial p}(\hat{r}, p_0)$$

we conclude that

$$\frac{dF}{dp}(\hat{r}, p_0) = \frac{dL}{dp}(\hat{r}, \eta_0, p_0). \tag{2.8}$$

In order to apply this equation, we first write down the Lagrange function $L(x, y^*, z^*, s, \Delta, p^*, \mu, \eta, \lambda)$ of (2.2), which is defined as follows:

$$\begin{aligned} L(x, y^*, z^*, s, \Delta, p^*, \mu, \eta, \lambda) = {} & \sigma f(x, y^*, z^*) + (1-\sigma)\sum_{j=1}^{n}(s_j^+ + s_j^-) + (1-\sigma)\sum_{i \in I'}\Delta_i \\ & + \lambda\, g(x, y^*, z^*) \\ & + \sum_{i \in I'} \eta_i \left(h_i(x, y^*) - \Delta_i - p_i^*\right) \\ & + \sum_{j=1}^{n} \mu_j^+(x_j - s_j^+ - \overline{x}_j) + \mu_j^-(\overline{x}_j - x_j - s_j^-) \end{aligned}$$

where the terms corresponding to constraints (2.2g) and (2.2h) are neglected, since they are not needed in the sequel and $p^* = 0$ as the parameter does not occur in problem (2.2).

Applying equation (2.8) to this Lagrange function, we obtain:

$$\partial_{p_i}\tilde{f}(\hat{x}, y^*, z^*, \hat{s}, \hat{\Delta}, p^*) = -\hat{\eta}_i. \tag{2.9}$$

Note that we do not need to check the assumptions of this equation, because our goal is to develop a heuristic method. That means, even if the assumptions are violated and the equality (2.8) might not be guaranteed, we can still try to apply the conclusions drawn from it.

2.2 The Basic Dual Value MINLP Heuristic

For a general MINLP with indicator constraints as in (2.1), the heuristic now works as follows. An iterative process is started at problem (2.2) based on the decisions in y^* and z^*. In each step the heuristic selects one fixed integer variable, and assigns a different integer value to it (in the special case of binary variables, this corresponds to a flip, from upper to lower bound, or vice versa). To this end, all potential integer variables are ranked according to the current absolute values of the dual solutions, i.e., the right-hand sides in (2.9). The variable corresponding to the constraint with the highest rank is then the most promising candidate for an assignment of a different value. That is, if $|\hat{\eta}_i| \geq \max\{|\hat{\eta}_j| : j \in I'\}$

for some $i \in I'$, then the corresponding variable z_i (which is currently fixed to $z_i^* = 1$) will be fixed to 0 in the next iteration. We also keep track of the variables which were changed in previous steps, so that we do not repeat the same decision again in later steps, in order to prevent the heuristic from cycling.

In the next iteration, we solve the slack model (2.2) again. Note that the set I' changes from one iteration to the next. The objective function values $\tilde{f}$ from the previous iteration is compared with the objective function value of the next iteration, denoted by $\tilde{f}_+$. We allow a slight increase (worsening) of at most 20% (or any other user-defined parameter) to accept this move. If the increase is more than that, the heuristic terminates without result.

On the long run, the objective function value of the slack model typically decreases. Two cases can occur: First, at some iteration, we reach a point where the slack part of the objective function value is 0. In this case we obtain a feasible solution and the heuristic terminates successfully. The other, second, case that can occur is that the slack part of the objective function does not converge to 0. If after a user-defined number of rounds the slack part is still non-zero, then the heuristic terminates. It failed to construct a feasible solution.

2.3 Embedding the Dual Value Heuristic in a Branch-and-Prune Search

The above outlined heuristic can be embedded in a tree search. The heuristic can run into a dead end, if the slack part of the objective does not converge to 0 after a certain number of iterations. The idea of the tree search is to restart the heuristic at an earlier stage, and thus to cover a wider range of potential alternative decision about which integer variable to change.

To this end, we do not only consider the single best variable with respect to the dual values as before. Instead, we take a small pool (of a user-defined size, typically 5-10 variables), and take the best dual variable until the pool is full. Each iteration of the heuristic is considered as a node in the tree (and the start as the root node), and for each variable in the pool a child node is created and inserted in the tree.

We then travers the tree in a depth-first search. If the heuristic fails, the node is pruned and a back-tracking to the previous unpruned nodes takes place.

3 A Specialization to a Nonlinear Network Design Application

The development of our heuristic method is motivated by a natural gas network topology optimization problem. Gas networks consists of passive elements, mainly pipelines, and active elements, such as compressors, valves, and control valves, that are controlled by human or automated dispatchers. The purpose of such network is to deliver gas from entry points (gas fields, harbors, gas storage tanks, or other networks) to exit points (end customers, gas storage tanks, or other networks). The control of this network is carried out in such way that the gas is transported obeying all physical, technical, and legal constraints. Physical constraints are given by the physics of gas, technical constraints are pressure bounds for pipelines and other technical equipment, and legal (or contractual) bounds are pressure and flow bounds due to legal agreements with customers and other connected network operators (called nomination). For a given nomination d, the topology optimization problem (3.3) is to find a cost optimal selection of pipe capacities for the transmission of the specific flow d in the transmission network. Otherwise, if this transport is not possible for any selection of pipe capacities, the nomination is infeasible. For more information on the application background we refer to [21, 26, 23].

3.1 Gas Network Topology Optimization

In the following we describe a mixed-integer nonlinear model for the topology optimization problem in a transmission network. Our model integrates two features: if the set of potential extensions is empty, it can be used to determine if a configuration of all active elements is possible such that all physical, technical, and contractual constraints are fulfilled [35]. For a non-empty set of potential extensions it can be used to find a subset of extensions having minimal cost and allowing a feasible flow. To this end, the model must be solved numerically with a solving technique that has the potential to give a certificate for optimality, or to proof that no solution exists. The details of the heuristic for identifying feasible solutions within our solution technique will be subject of the following sections. We remark that other heuristic approaches exist that can be used to complement our approach [35].

We use the following notation. A transmission network is modeled by a directed graph $G = (V, E)$ where V denotes the set of nodes and $E \subset V \times V$ the set of arcs. We define a set of potential extension pipes $E_X \subset V \times V \times \mathbb{N}_0$, where a new pipe can be built between any pair of existing nodes $v, w \in V, v \neq w$. By $G_X := (V, E_X)$ we denote the transmission network together with its potential extensions. Note that G_X is a graph with multiple arcs.

For the ease of the notation of the problem formulation, we embed the originally given arcs from E in E_X with index 1, that is, all arcs $(v, w, 1) \in E_X$ are not extension arcs, but original arcs from E. Furthermore, all arcs $(v, w) \in E$ that represent valves are additionally represented by the arc $(v, w, 0) \in E_X$.

Let us introduce the following variables. The flow on arc $(e, i) \in E_X$ is denoted by $q_{e,i} \in \mathbb{R}$, where a positive value means the flow is heading in the same direction as the arc, and a negative value indicates the opposite direction. We remark that there are no explicit bounds on $q_{e,i}$ that would specify a maximal absolute flow value in either direction, but implicitly given bounds by the bounds on the potential values and the pressure-flow-coupling constraints (3.1). The potential value of a vertex $v \in V$ is given by $\pi_v \in \mathbb{R}_+$. In a gas transmission network this variable refers to the squared pressure in this node. The pressure itself is given by $p_v \in \mathbb{R}_+$. The variable $y_{e,i} \in \mathbb{Z}$ specifies that additive component of the pressure loss term. For passive pipelines this variable is fixed to zero, whereas for active elements it defines the operating range. We introduce a binary decision variable $x_{e,i} \in \{0, 1\}$ for each potential extension $(e, i) \in E_X$.

All different types of technical network elements that an arc can represent (i.e., compressor, valve, control valve, or pipeline) are modeled by this basic equation:

$$\alpha_{e,i} q_{e,i} |q_{e,i}|^{k_e} + \beta_{e,i} y_{e,i} - (\pi_v - \pi_w) = 0, \tag{3.1}$$

which is then specified by defining the parameters $\alpha_{e,i}, \beta_{e,i}$ and k_e appropriately. For valves we set $\beta_{e,i} = 0$. All potential extensions are indexed with $i > 1$. For arcs e with $k_e = 0$ extensions are not allowed. For arcs with $\alpha_{e,i} = 0$ we assume w.l.o.g. $k_e = 0$. Besides the decision if a new pipe between two nodes should be built, we need to decide further design parameters. For this we can select from a finite set of possible configurations, such as pipe diameters or coatings, which is reflected by index i for $i > 1$. For practical purpose we restrict to loop extensions $E_X \subseteq E \times \mathbb{N}_0$, that is, increasing the capacity of an existing connection between two nodes v and w by building a second pipeline taking the same path. However, the model and methods we describe below would also allow for larger sets E_X, that is, it allows for extensions between nodes v and w that were not connected in E.

We assume the following data to be given as parameters. For each node $v \in V$ we have lower and upper bounds on the node potential, $\underline{\pi}_v, \overline{\pi}_v \in \mathbb{R}$ with $\underline{\pi}_v \leq \overline{\pi}_v$. For each node $v \in V$ the value $d \in \mathbb{R}$ denotes the amount of flow that is either led into the network (for $d_v > 0$), or taken out of the network (for $d_v < 0$). A node with $d_v > 0$ is also called source or entry node, and nodes with $d_v < 0$ are sinks or exit nodes. All other nodes with $d_v = 0$ are inner or transmission nodes. Vector v is also called *nomination*. In order not to pose a problem that is trivially infeasible, only those nominations are allowed that have equal entry and exit flows, that is,

$$\sum_{v \in V} d_v = 0.$$

Such nominations are said to be *balanced*. For each arc $(v, w, i) \in E_X$ we have a transmission coefficient $\alpha_{e,i} \in \mathbb{R}_+ \backslash \{0\}$, bounds on the range coefficient $\underline{y}_{e,i}, \overline{y}_{e,i} \in \mathbb{R}$ with $\underline{y}_e \leq \overline{y}_e$, a scaling factor $\beta_{e,i}$ for the range coefficient, and a cost coefficient $c_{e,i} \in \mathbb{R}_+$. For those arcs (e, i) representing compressors or control valves we assume data on their operating range being given as a $3 \times \nu_{e,i}$-dimensional matrix $A_{e,i}$ and $\nu_{e,i}$-dimensional vector $b_{e,i}$ (here $\nu_{e,i}$ is the number of linear constraints necessary to describe the operating range). For all other arcs, we set $A_{e,i} = 0$ and $b_{e,i} = 0$. For technical reasons only a subset of all possible discrete settings is allowed. This subset is given by $\mathcal{X} \subseteq \{0, 1\}^{E_X}$. The set $\mathcal{X}$ is described by linear inequalities in general, i.e.,

$$\mathcal{X} = \left\{ x \in \{0, 1\}^{E_X} \mid l(x) \leq 0 \right\}. \tag{3.2}$$

where $l(x)$ is a multidimensional linear function.

The following non-linear non-convex mixed-integer model with indicator constraints is called topology optimization problem:

$$\min \sum_{(e,i)\in E_X} c_{e,i}x_{e,i} \quad \text{s.t.} \tag{3.3a}$$

$$x_{e,i} = 1 \Rightarrow \alpha_{e,i} q_{e,i}|q_{e,i}|^{k_e} + \beta_{e,i} y_{e,i} - (\pi_v - \pi_w) = 0 \qquad \forall\, (e,i) \in E_X, i \neq 0, (v,w) = e, \tag{3.3b}$$

$$x_{e,i} = 1 \Rightarrow A_{e,i} \cdot (q_{e,i}, p_v, p_w)^T \leq b_{e,i} \qquad \forall\, (e,i) \in E_X, i \neq 0, (v,w) = e, \tag{3.3c}$$

$$x_{e,i} = 0 \Rightarrow q_{e,i} = 0 \qquad \forall\, (e,i) \in E_X, i \neq 0, \tag{3.3d}$$

$$x_{e,i} = 0 \Rightarrow y_{e,i} = 0 \qquad \forall\, (e,i) \in E_X, i \neq 0, \tag{3.3e}$$

$$\sum_{i:(v,w,i)\in E_X} x_{e,i} = 1 \qquad \forall\, e \in E, \tag{3.3f}$$

$$\sum_{\substack{w,i:(v,w,i)\in E_X \\ i\neq 0}} q_{v,w,i} - \sum_{\substack{w,i:(w,v,i)\in E_X \\ i\neq 0}} q_{w,v,i} = d_v \qquad \forall\, v \in V, \tag{3.3g}$$

$$|p_v| p_v - \pi_v = 0 \qquad \forall\, v \in V, \tag{3.3h}$$

$$\pi_v \leq \overline{\pi}_v \qquad \forall\, v \in V, \tag{3.3i}$$

$$\pi_v \geq \underline{\pi}_v \qquad \forall\, v \in V, \tag{3.3j}$$

$$x_{e,i} = 1 \Rightarrow y_{e,i} \leq \overline{y}_{e,i} \qquad \forall\, (e,i) \in E_X, i \neq 0, \tag{3.3k}$$

$$x_{e,i} = 1 \Rightarrow y_{e,i} \geq \underline{y}_{e,i} \qquad \forall\, (e,i) \in E_X, i \neq 0, \tag{3.3l}$$

$$q_{e,i} \in \mathbb{R} \qquad \forall\, (e,i) \in E_X, i \neq 0, \tag{3.3m}$$

$$\pi_v \in \mathbb{R}_+ \qquad \forall\, v \in V, \tag{3.3n}$$

$$p_v \in \mathbb{R}_+ \qquad \forall\, v \in V, \tag{3.3o}$$

$$y_{e,i} \in \mathbb{R} \qquad \forall\, (e,i) \in E_X, i \neq 0, \tag{3.3p}$$

$$x_{e,i} \in \{0,1\} \qquad \forall\, (e,i) \in E_X, \tag{3.3q}$$

$$x \in \mathcal{X}. \tag{3.3r}$$

The objective function (3.3a) calculates the extension costs for those new pipes that are actually built. The indicator constraints (3.3b) are switching on only those pressure-flow coupling constraints for potential arcs that are actually built. The indicator constraints (3.3c) are switching on active compressors or control valves. The operating range of compressors and control valves are described each by a family of linear inequalities. For details on compressor and control valve modeling we refer to [35]. The indicator constraints (3.3d) forbid flow on those arcs that are not used, that is, they are either not built or switched off by a closed valve. Exactly one pressure loss constraint (3.3b) must be selected which is guaranteed by constraints (3.3f). The node flow conservation constraints (also called Kirchhoff's constraints) are defined in (3.3g). Constraints (3.3h) are handling the transformation between pressures and squared pressure variables and constraints (3.3i) – (3.3l) define bounds on the variables. The last constraint (3.3r) represents the restriction of the discrete variables x to the technical feasible set $\mathcal{X}$ defined by (3.2).

3.2 Exploiting Problem Structure

A typical situation in the network is a subnet consisting of a compressor (or a control valve) that can be active (compressing or regulating), in bypass, or closed. Figure 3.1 shows this subnetwork, consisting of the compressor itself, a valve, and connecting short-pipes, i.e., pipes with negligible pressure loss. The gas can either flow through the compressor. In this case, the amount of flow is restricted by lower and upper bounds. The output pressure at the exit side of the compressor is higher than the input pressure, and the compressor ration between output and input pressure has to statisfy certain bounds (typical

bounds for the ratio are 1.2 ... 2.2). The operating range of a compressor (e, i) is thus described by $\nu_{e,i}$ linear inequalities in the system $(A_{e,i}, b_{e,i})$. They define a set of feasible points illustrated by the shaded area in the upper-right of Figure 3.2. If the subnetwork is in bypass, then the pressure at the input side is identical to the pressure at the output side, and the flow can vary arbitrarily (c.f. the horizontal line in Figure 3.2). Finally, if the subnet is in close mode, then the flow is zero, and the pressures at input and output side can take arbitrary values (c.f. the vertical line in Figure 3.2). The situation for a control valve subnetwork is analog, see Figure 3.3, with the only difference that the output pressure is now less than the input pressure in 'active' mode.

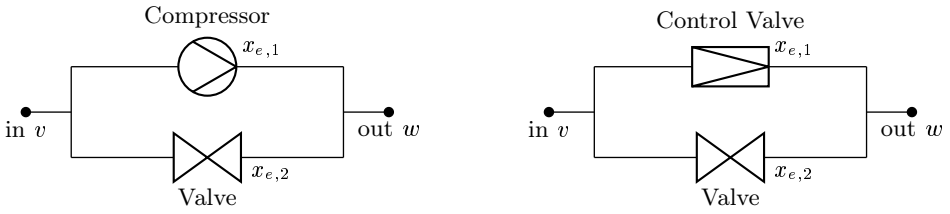

Fig. 3.1. A compressor subnet (left) and a control valve subnet (right) allowing the states 'active', 'closed', or 'bypass'.

We denote by $x_{e,1}, x_{e,2}$ the binary variables associated with the compressor and the valve on arc e, see Figure 3.1, respectively. The binary variable $x_{e,0}$ is used to represent the case that the active element is closed. Note that in Figure 3.1 the nodes are denoted by v and w and the arcs are denoted by $(e, 1), (e, 2)$ (the arc $(e, 0)$ that exists in our model formulation is not shown explicitely). The crucial part of this model that we will exploit in the sequel are the following three families of indicator constraints:

$$x_{e,0} = 1 \Rightarrow \begin{cases} q_{e,1} = 0 & \text{(3.4a)} \\ q_{e,2} = 0 & \text{(3.4b)} \end{cases}$$

$$x_{e,1} = 1 \Rightarrow \begin{cases} A_{e,1} \cdot (q_{e,1}, p_v, p_w)^T \leq b_{e,1} & \text{(3.5a)} \\ q_{e,2} = 0 & \text{(3.5b)} \end{cases}$$

$$x_{e,2} = 1 \Rightarrow \begin{cases} \pi_v - \pi_w = 0 & \text{(3.6a)} \\ q_{e,1} = 0 & \text{(3.6b)} \end{cases}$$

The heuristic creates a new branching tree $\mathcal{T}$. Its root node consists of some fixation of the discrete decision variables to integer values. If this setting results in an infeasible solution for (3.3), we solve the slack variant (2.2) of model (3.3) and obtain, after applying the steps described above, a ranking among the binary variables (according to the absolute value of the dual solution values) that indicates which binary variable should be flipped from zero to one or vice versa. The difference to the general case of the heuristic is that we consider the states of two binary from the subnetwork simultaneously.

Denote by ρ_f a certain user-specified threshold value for the flow (default value is 20), and by ρ_p a threshold value for the pressure difference (default value is 10).

New child nodes for tree $\mathcal{T}$ are created, where we distinguish the following cases.

Case 1: $x_{e,0}$ from 1 to 0. The subnet is currently "closed". The ranking indicates that the variable $x_{e,0}$, having the present value 0, should be set to 1.

- **Case 1.1:** "From closed to bypass." If the pressure difference at both end nodes $p_w - p_v$ is below ρ_p, then a new child node is created where $x_{e,2}$ is set from 0 to 1 and $x_{e,1}$ remains at 0.
- **Case 1.2:** "From closed to active." If the dual value of at least one of the constraints (3.4a) or (3.4b) is positive, then in the new child we flip $x_{e,1}$ from 0 to 1 and $x_{e,2}$ remains at 0.

Case 2: $x_{e,1}$ from 1 to 0. The subnet is currently "active". We collect all constraints of (3.5a) that are either fulfilled with equality or where the associated value of s is greater than zero. (Note that we solve a slack model, which ensures feasiblity.) Denote by $I \subseteq \{1, \ldots, \nu_{e,1}\}$ the corresponding index

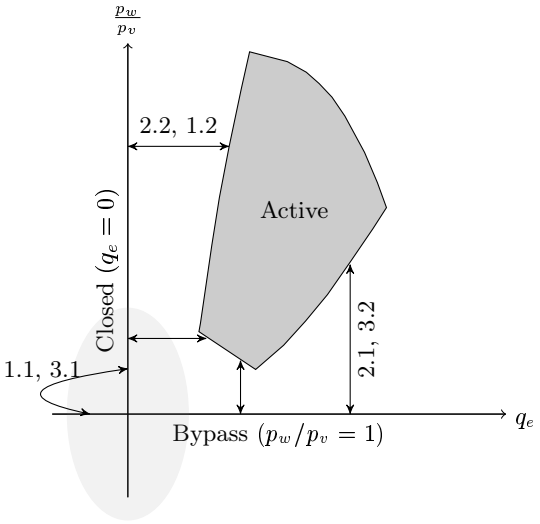

Fig. 3.2. Changes to binary variables for a compressor, indicated by the heuristic, labeled by cases.

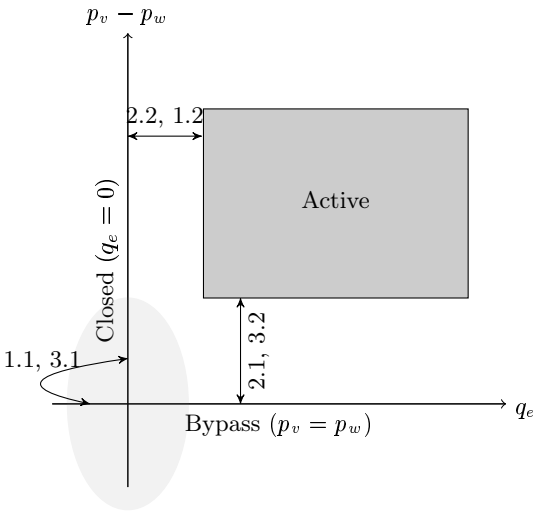

Fig. 3.3. Changes to binary variables for a control valve, indicated by the heuristic, labeled by cases.

set. For each $i \in I$ we consider the i-th constraint in (3.5a) and denote by (a_1, a_2, a_3) the coefficients from the i-th row of $A_{e,1}$. We fix the input pressure $p_v = 1$, and check in which quadrant the remaining two-dimensional vector (a_1, a_3) lies.

Case 2.1: "From active to bypass." If $a_3 < 0$, then we create a child node in which we set $x_{e,1}$ to 0 and $x_{e,2}$ to 1.

Case 2.2: "From active to closed." If $a_1 < 0$, then we create a child node in which we set $x_{e,1}$ to 0 and also $x_{e,2}$ to 0.

Case 3: $x_{e,2}$ from 1 to 0. The subnet is currently in "bypass".

Case 3.1: "From bypass to closed." If the absolute value of flow is below ρ_f, then a new child node is created with the decision to set $x_{e,2}$ also to 0, to leave $x_{e,1}$ at 0, and thus set $x_{e,0}$ to 1. Hence at this child node, the subnetwork is 'closed'.

Case 3.2: "From bypass to active." If the dual value of the constraint (3.6a) is positive, then a new child node is created, in which we set $x_{e,1}$ from 0 to 1 and $x_{e,2}$ from 1 to 0. (Note that both cases 1.1.1 and 1.1.2 can occur simultaneously, hence two child nodes are created.)

Otherwise: Consider the next best variable according to the ranking, and check if one of the above cases applies. If no more variable exists, do not create a child node.

The "otherwise" case can occur, for example, if the ranking favors an increase of compression for an active compressor, while the compression ratio is already at its upper limit.

For control valves we create new child nodes in the very same way as for compressor stations. For simple (non-control) valves, the state "active" does not apply; they can only be in "bypass" or "closed". Hence we apply the same child node creation routine as for compressor station, but skip those parts that are concerned with "active" states.

After inserting the new child nodes to the tree, we perform the following node selection strategy, in order to decide where to continue the search. We always proceed according to the rule: "active" first, "bypass" second, "closed" third. We always pass along from a parent node to one of its child nodes according to this rule (depth-first search). When entering the child node, at first the slack model (2.2) is solved again. This yields a new ranking of the decision variables, which lead to a new decision about which variable to flip in the next round.

This branching process terminates if at one node the slack model has a zero objective function value, or a certain user-defined branching depth (i.e., number of nodes in the path to the root node) is reached. Then the corresponding child node is pruned and the search procedure continues at another unfinished child node. Here we apply a simple backtracking rule, going back to the previous parent node from which we came before.

An example is shown in Figure 3.4. Here we consider a network (or a section of a larger network) that consists of one control valve (CV) and one compressor station (CS). Assume that either from the LP relaxation or by branching at the current node the CV is set to active mode and the CS is set to bypass mode and assume that this is not feasible. Now the heuristic is invoked. It creates a ranking of the corresponding decision variables, which means, a ranking of the network elements. We assume that the CV is higher ranked than the CS. Hence the heuristic creates a branch (shown on the left), where it first changes the stage of the CV from active to bypass (according to Case 2.1). Now we solve the NLP (2.2) again and obtain a new ranking. Among all elements that have not been changed before, the CS now has the highest ranking. Hence it is changed, once from bypass to active (left son), and once from bypass to closed (right son). For the left son, we solve the NLP relaxation (2.2), and assume that it still has a positive slack value. Since we cannot alter more elements in this small test network, this part of the tree is finished, and we track back to the parent node. For the right son, we solve the NLP relaxation, and again get an infeasible subproblem, so we also prune this node. So we are back at the root node of the search tree created by the heuristic. The subtree where CV was changed from active to bypass is finished, hence we now change the CV from active to closed (shown on the right). Now we solve the NLP (2.2) and assume, that we obtain a solution with slack zero. Then a new primal feasible solution was found.

4 Implementational Details

We implemented the algorithms described above in C on a cluster of 64bit Intel Xeon X5672 CPUs at 3.20 GHz with 12 MByte cache and 48 GB main memory, running an OpenSuse 12.1 Linux with

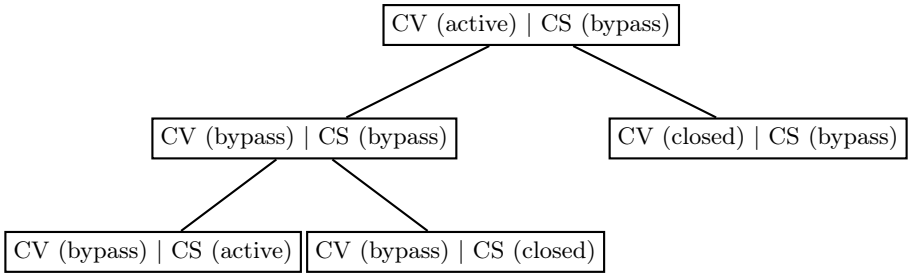

Fig. 3.4. Branching tree created by the heuristic for a network that contains one control valve (CV) and one compressor station (CS).

a gcc 4.6.2 compiler. We used the following software packages: SCIP 3.0.1 as mixed-integer nonlinear branch-and-cut framework (for details on SCIP we refer to [1]), CPLEX 12.1 [14] as linear programming solver, Ipopt 3.10 [41] as nonlinear solver, and Lamatto++ [24] as framework for handling the input data and for performing some problem specific presolving. Hyperthreading and Turboboost were disabled. In all experiments, we ran only one job per node to avoid random noise in the measured running time that might be caused by cache-misses if multiple processes share common resources.

The heuristic is implemented as a plug-in for the solver SCIP. The heuristic offers the following settings for the user:

Maxcalls. Specifies the maximal number of nodes for the branching tree within the heuristic. (Default value is 30.)
Backtrack. If objective function of slack model remains equal or increases within the unique path connecting the current node and the root for "Maxcalls" nodes, then move "Backtrack" nodes upwards in the branching tree. (Default value is 0.)
Maxequals. Abort the heuristic without a solution, if the highest-ranked "Maxequals" fraction of integer variables all have equal ranks. (Default value is 1/3, i.e., 33.33% of all integer variables.)
Mingap. Do not call the heuristic, if the gap is below the threshold value "Mingap". The gap is defined as: (best primal − best dual) / (best dual). (Default value is 0.05, i.e., a SCIP gap of 5%.)
Sigma. Sets the value for σ in objective (2.2a). (Default value is 0, i.e., emphasis only on feasibility.)
Leaves only. Call the heuristic only at leaves of the branching tree. (Default value is "No".)

We apply the heuristic to nonlinear network design instances (3.3). The heuristic is invoked at those nodes in the branch-and-bound tree where all binary decision variables $x_{e,i}$ have integer values (either by branching or by the solution of the LP relaxation). In order to make this happen as early as possible, we introduced a branching priority rule that first branches on integer variables, before spatial branching on continuous variables is applied. Furthermore, when changing x^* in our heuristic to x' in the next iteration, we ensure that x' is feasible for $\mathcal{X}$, i.e., $x' \in \mathcal{X}$. This is done as follows: The set $\mathcal{X}$ is defined by linear inequalities, see (3.2). We assume that $x_{e_0,i}$ is to be changed from zero to one. Then we solve the following MIP problem to obtain the next vector x':

$$\begin{aligned} \min\, & ||x' - x^*||_1 \\ \text{s.t.}\quad & l(x') \le 0, \\ & x'_{e_0,i} = 1, \\ & x' \in \{0,1\}^{E_X}. \end{aligned} \tag{4.1}$$

If this problem is infeasible, we prune the current node of the branching tree of our heuristic.

Let us write the relaxation (2.2) for our topology optimization problem (3.3). This relaxation is always feasible (see [27] for details). We define by $E' := \{(e,i) \in E_X \mid x_{e,i} = 1\}$ the sets of arcs that are selected at the current node, i.e., where the flow is not fixed to zero by (3.3d). We introduce $\Delta_v \in \mathbb{R}_+$ for all $v \in V$ and $\Delta_{e,i} \in \mathbb{R}_+^{\nu_{e,i}}$ as slack variables. Then the relaxation writes as follows:

$$\min \sum_{v \in V} \Delta_v + \sum_{(e,i) \in E_X} \sum_{k=1}^{\nu_{e,i}} (\Delta_{e,i})_k \quad \text{s.t.} \tag{4.2a}$$

$$\alpha_{e,i} q_{e,i} |q_{e,i}|^{k_e} + \beta_{e,i} y_{e,i} - (\pi_v - \pi_w) = 0 \qquad \forall\, (e,i) \in E', (v,w) = e, \tag{4.2b}$$

$$A_{e,i} \cdot (q_{e,i}, p_v, p_w)^T - \Delta_{e,i} \leq b_{e,i} \qquad \forall\, (e,i) \in E', (v,w) = e, \tag{4.2c}$$

$$\sum_{w,i:(v,w,i)\in E'} q_{v,w,i} - \sum_{w,i:(w,v,i)\in E'} q_{w,v,i} = d_v \qquad \forall\, v \in V, \tag{4.2d}$$

$$|p_v| p_v - \pi_v = 0 \qquad \forall\, v \in V, \tag{4.2e}$$

$$\pi_v - \Delta_v \leq \overline{\pi}_v \qquad \forall\, v \in V, \tag{4.2f}$$

$$\pi_v + \Delta_v \geq \underline{\pi}_v \qquad \forall\, v \in V, \tag{4.2g}$$

$$y_{e,i} \leq \overline{y}_{e,i} \qquad \forall\, (e,i) \in E', \tag{4.2h}$$

$$y_{e,i} \geq \underline{y}_{e,i} \qquad \forall\, (e,i) \in E', \tag{4.2i}$$

$$q_{e,i} \in \mathbb{R} \qquad \forall\, (e,i) \in E', \tag{4.2j}$$

$$y_{e,i} \in \mathbb{R} \qquad \forall\, (e,i) \in E', \tag{4.2k}$$

$$\Delta_v \in \mathbb{R}_+ \qquad \forall\, v \in V, \tag{4.2l}$$

$$\Delta_{e,i} \in \mathbb{R}_+^{\nu_{e,i}} \qquad \forall\, (e,i) \in E'. \tag{4.2m}$$

With this slack model we proceed as described in Section 3.2.

5 Computational Results

Using our implementation we obtain the results shown in Table 1. Test data was provided by Open Grid Europe GmbH, a major German gas network operator. The initial network `net1` consists of 4155 nodes and 4094 arcs, among them 3657 passive pipelines, 12 compressor stations, 120 control valves, and 305 valves. The industry partner defined a test set of 31 instances for `net1` that differ among each other in the pressure and flow bounds on the entry and exit nodes. The problem is a feasibility problem only (i.e., does there exist a feasible flow in the network), hence it is considered as "solved" as soon as a feasible solution is constructed, no objective function values for opening valves (i.e., adding further topology extensions) are imposed. The third column of Table 1 shows the number of instances that were solved with SCIP using only default settings. The fourth column of Table 1 shows the number of instances that were solved by the heuristic. The last column of Table 1 shows the number of instances that were solved by the heuristic exploiting the problem structure. For both, SCIP/default and SCIP/dualval (in its variants basic and modified), we impose a time limit of 4 hours.

The modified version of the heuristic is able to solve all problem instances for `net1`. In later iterations (`net2` – `net5`), the industrial project partner made minor alterations to the networks' topologies, and, more important, also made the instances more and more difficult by imposing tighter bounds on the minimum and maximum pressure levels at nodes. Hence it became more difficult for any heuristic to construct feasible solutions. However, the adapted heuristic was still able to solve around two thirds of the instances, whereas the standard heuristics of SCIP found fewer feasible solutions the more difficult the instances became. Also the basic heuristic that is not adapted to the problem's special structure is rather good in finding feasible solutions (comparable to SCIP/default).

network	no. of inst.	SCIP/default	SCIP/dualval(basic)	SCIP/dualval(adapted)
`net1`	31	9	24	31
`net2`	31	2	12	18
`net3`	31	2	9	19
`net4`	31	1	6	18
`net5`	31	0	8	19

Table 1. Solved instances on different test networks with each having 31 instances.

Figure 5.1 shows a comparison of SCIP with default settings (in particular, with all standard heuristics) and SCIP with the additional adapted heuristic on all instances from Table 1. Each cross (×) marks the runtime of these two solvers. A cross above the diagonal line indicates that the default SCIP is faster, and a cross below the diagonal line indicates that SCIP with our heuristic is faster. It turns out that for those instances that SCIP/default is already able to solve, the additional time spent in the dual value heuristic does not pay off, but instead slows down the overall solution process. For many of those instances that SCIP/default could not solve, SCIP/dualval is able to find a solution within the time limit.

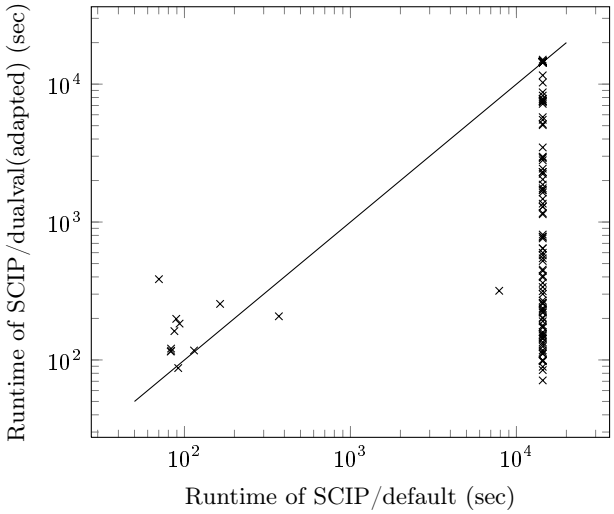

Fig. 5.1. Runtime comparison between SCIP/default and SCIP/dualval.

6 Outlook and Conclusion

We presented a new heuristic algorithm that exploits dual informations coming from KKT solutions in order to find feasible solutions for MINLP problems. We applied this algorithm to a problem where we have more information on the problem structure available, namely the topology optimization in natural gas networks. After exploiting the structure, the heuristic is able to identify much more feasible solutions than any other heuristic we are aware off.

It turned out that the structural analysis of the given MINLP with indicator constraints is an important step in making our heuristic applicable. We carried out initial computations on a general MINLP benchmark instance test set. The experiences we gathered show a mixed picture: On some instances our heuristic was able to find feasible solutions faster than previous heuristics implemented in SCIP, whereas on others it did not find feasible solutions and thus only wasted CPU time. To become more competitive it is necessary to investigate the behavior of our method in such general settings, and automatically detect and exploit problem structures.

Acknowledgements

We are grateful to Open Grid Europe GmbH (OGE, Essen/Germany) for supporting our work. Armin Fügenschuh conducted parts of this research under a Konrad-Zuse-Fellowship.

References

1. T. Achterberg. SCIP: Solving Constraint Integer Programs. *Math. Progr. Comp.*, 1(1):1–41, 2009.

2. T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, 2007.
3. T. Achterberg, T. Berthold, and G. Hendel. Rounding and propagation heuristics for mixed integer programming. In D. Klatte, H.-J. Lüthi, and K. Schmedders, editors, *Operations Research Proceedings 2011*, pages 71–76. Springer Verlag, Berlin, 2012.
4. E. Balas, S. Schmieta, and C. Wallace. Pivot and shift – a mixed integer programming heuristic. *Discrete Optimization*, 1(1):3–12, 2004.
5. P. Belotti. Couenne: a user's manual. 2009.
6. T. Berthold. RENS – the optimal rounding. Technical Report ZR-12-17, Zuse Institute Berlin, Takustraße 7, 14195 Berlin, Germany, 2012.
7. T. Berthold. Primal MINLP heuristics in a nutshell. ZIB-Report 13–2, Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, 2013.
8. Timo Berthold, Stefan Heinz, and Stefan Vigerske. Extending a CIP framework to solve MIQCPs. ZIB-Report 09–23, Zuse Institute Berlin, Takustr.7, 14195 Berlin, Germany, 2009.
9. R.E. Bixby. Commentary: Progress in linear programming. *ORSA J. Comput.*, 6:15–22, 1994.
10. Robert E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Res.*, 50(1):1–13, 2002.
11. P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programming. *Math. Programming*, 119(2):331–352, 2009.
12. P. Bonami and J. Goncalves. Heuristics for convex mixed integer nonlinear programs. *Comp. Optim. Appl.*, 51:729–747, 2012.
13. Pierre Bonami, Lorenz T. Biegler, Andrew R. Conn, Gérard Cornuéjols, Ignacio E. Grossmann, Carl D. Laird, Jon Lee, Andrea Lodi, François Margot, Nicolas Sawaya, and Andreas Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008.
14. CPLEX. *User's Manual for CPLEX*. IBM Corporation, Armonk, USA, 12.1 edition, 2011.
15. C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. A storm of feasibility pumps for nonconvex minlp. *Math. Programming*, 136:375–402, 2012.
16. E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Math. Programming*, 102:71–90, 2005.
17. A. V. Fiacco and Y. Ishizuka. Sensitivity and stability analysis for nonlinear programming. *Ann. Oper. Res.*, 27(1):215–236, 1990.
18. M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Math. Programming*, 104(1):91–104, 2005.
19. M. Fischetti and D. Salvagnin. Feasibility pump 2.0. *Math. Progr. Comp.*, 1:201–222, 2009.
20. C.A. Floudas and R. Misener. A framework for globally optimizing mixed-integer signomial programs. *J. of Optimization Theory and Applications*, 2013.
21. A. Fügenschuh, B. Hiller, J. Humpola, T. Koch, T. Lehman, R. Schwarz, J. Schweiger, and J. Szabó. Gas network topology optimization for upcoming market requirements. *IEEE Proc. 8th Intern. Conf. on EEM 2011*, pages 346–351, 2011.
22. A. Fügenschuh and A. Martin. Computational integer programming and cutting planes. In R. Weissmantel K. Aardal, G. Nemhauser, editor, *Handbook on Discrete Optimization*, pages 69–122. Elsevier, 2005.
23. Armin Fügenschuh, Björn Geißler, Ralf Gollmer, Christine Hayn, Rene Henrion, Benjamin Hiller, Jesco Humpola, Thorsten Koch, Thomas Lehmann, Alexander Martin, Radoslava Mirkov, Antonio Morsi, Werner Römisch, Jessica Rövekamp, Lars Schewe, Martin Schmidt, Rüdiger Schultz, Robert Schwarz, Jonas Schweiger, Claudia Stangl, Marc C. Steinbach, and Bernhard M. Willert. Mathematical optimization for challenging network planning problems in unbundled liberalized gas markets. ZIB-Report 13–13, Zuse Institute Berlin, Takustr.7, 14195 Berlin, Germany, 2013.
24. B. Geißler, A. Martin, and A. Morsi. LaMaTTO++. information available at URL `http://www.mso.math.fau.de/edom/projects/lamatto.html`, February 2013.
25. M. Guzelsoy, G. Nemhauser, and M. Savelsbergh. Restrict-and-relax search for 0-1 mixed-integer programs. *EURO J. Comp. Opt.*, 1(1-2):201–218, 2013.
26. Jesco Humpola and Armin Fügenschuh. A new class of valid inequalities for nonlinear network design problems. ZIB-Report 13–06, Zuse Institute Berlin, Takustr.7, 14195 Berlin, Germany, 2013.
27. Jesco Humpola and Armin Fügenschuh. A unified view on relaxations for a nonlinear network flow problem. ZIB-Report 13–31, Zuse Institute Berlin, Takustr.7, 14195 Berlin, Germany, 2013.
28. A. Koberstein. *The Dual Simplex Method, Techniques for a fast and stable implementation*. PhD thesis, Universität Paderborn, Fakultät für Wirtschaftswissenschaften, 2005.
29. L. Liberti, N. Mladenović, and G. Nannicini. A recipe for finding good solutions to MINLPs. *Math. Progr. Comp.*, 3:349–390, 2011.
30. A. Mahajan, S. Leyffer, and C. Kirches. Solving mixed-integer nonlinear programs by QP-diving. Technical Report ANL/MCS-2071-0312, Argonne National Laboratory, Mathematics and Computer Science Division, 2012.
31. A. Majahan, S. Leyffer, J.T. Linderoth, J. Luedtke, and T. Munson. MINOTAUR: A toolkit for solving mixed-integer nonlinear optimization. http://wiki.mcs.anl.gov/minotaur.

32. G. Nannicini and P. Belotti. Rounding-based heuristics for nonconvex MINLPs. *Math. Progr. Comp.*, 4:1–31, 2012.
33. G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, 1988.
34. W. Orchard-Hays. History of the development of LP solvers. *Interfaces*, 20(4):61–73, 1990.
35. Marc E. Pfetsch, Armin Fügenschuh, Björn Geißler, Nina Geißler, Ralf Gollmer, Benjamin Hiller, Jesco Humpola, Thorsten Koch, Thomas Lehmann, Alexander Martin, Antonio Morsi, Jessica Rövekamp, Lars Schewe, Martin Schmidt, Rüdiger Schultz, Robert Schwarz, Jonas Schweiger, Claudia Stangl, Marc C. Steinbach, Stefan Vigerske, and Bernhard M. Willert. Validation of nominations in gas network optimization: Models, methods, and solutions. ZIB-Report 12–41, Zuse Institute Berlin, Takustr.7, 14195 Berlin, Germany, 2012.
36. E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS J. Comp.*, 19(4):534–541, 2007.
37. E.M.B. Smith and C.C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimization of nonconvex MINLPs. *Comp. Chem. Eng.*, 23:457–478, 1999.
38. M. Tawarmalani and N.V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, 2002.
39. M. Tawarmalani and N.V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Math. Programming*, 99(3):563–591, 2004.
40. M. Tawarmalani and N.V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Math. Programming*, 103(2):225–249, 2005.
41. A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Programming*, 106(1):25–57, 2006.
42. R. Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus*. PhD thesis, Technische Universität Berlin, Fachbereich Mathematik, 2006.