Mathematical Programming Computation (2019) 11:95–118
https://doi.org/10.1007/s12532-018-0141-x

FULL LENGTH PAPER

Computing feasible points for binary MINLPs with MPECs

Lars Schewe[1,2] · **Martin Schmidt**[1,2]

Received: 21 December 2016 / Accepted: 4 June 2018 / Published online: 23 August 2018
© Springer-Verlag GmbH Germany, part of Springer Nature and The Mathematical Programming Society 2018

Abstract
Nonconvex mixed-binary nonlinear optimization problems frequently appear in practice and are typically extremely hard to solve. In this paper we discuss a class of primal heuristics that are based on a reformulation of the problem as a mathematical program with equilibrium constraints. We then use different regularization schemes for this class of problems and use an iterative solution procedure for solving series of regularized problems. In the case of success, these procedures result in a feasible solution of the original mixed-binary nonlinear problem. Since we rely on local nonlinear programming solvers the resulting method is fast and we further improve its reliability by additional algorithmic techniques. We show the strength of our method by an extensive computational study on 662 MINLPLib2 instances, where our methods are able to produce feasible solutions for 60% of all instances in at most 10 s.

Keywords Mixed-integer nonlinear optimization · MINLP · MPEC · Complementarity constraints · Primal heuristic

Mathematics Subject Classification 90-08 · 90C11 · 90C33 · 90C59

1 Introduction

In this paper we consider nonconvex mixed-binary nonlinear optimization problems (MINLP) and develop and test primal heuristics that are based on reformulations of the MINLP as a mathematical program with equilibrium constraints (MPEC). After this transformation we apply standard regularization-based MPEC solution algorithms. In the case of success, these algorithms yield MPEC stationary points that are feasible for the original MINLP.

✉ Lars Schewe
lars.schewe@fau.de

1 Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Discrete Optimization, Cauerstr. 11, 91058 Erlangen, Germany

2 Energie Campus Nürnberg, Fürther Str. 250, 90429 Nuremberg, Germany

MINLPs are an important class of optimization problems. They combine the capability of modeling both nonlinearities and discrete aspects using integer variables. This combination is frequently required in practical applications. On the other hand it is exactly this combination that makes MINLPs often extremely hard to solve in practice. Thus, in many cases it is reasonable to try to find feasible solutions of good quality quickly instead of solving the problem to proven global optimality. Moreover, global optimization solvers typically make use of primal heuristics to reduce their search space by using feasible solutions that have been found early in the solution process; see, e.g., the publications about the global solvers SCIP [1,28] or BARON [32,38–40]. These reasons yield a large variety of primal heuristics that can be found in the literature and that are used either as stand-alone methods or as subroutines in global solvers.

Many ideas of primal heuristics for MINLPs are direct generalizations of successful methods that have been known for mixed-integer linear problems (MIPs). Examples can be found, e.g., in [9], where the authors generalize diving heuristics, the feasibility pump, and a relaxation induced neighborhood search (RINS) from the MIP context to convex MINLPs. Feasibility pumps for MIPs have also been generalized for MINLPs in [8] or, for nonconvex MINLPs, in [10,11]. Other generalizations are given in [30], where local branching is carried over from MIP (see [17]) to nonconvex MINLP or in [7], where classical heuristics for MIP and constrained programming are used to improve the performance of a constraint integer programming framework for solving mixed-integer quadratically constrained problems.

Another type of heuristics that are used for solving MINLPs are more general classes of methods like metaheuristics. For instance, Berthold et al. [6] extended classical local neighborhood search (LNS) heuristics for MIPs in order to be applied to nonlinear problems. Similar work is presented in [26] where the authors discuss a general-purpose heuristic based on variable neighborhood search, local branching, a local nonlinear programming solver, and branch-and-bound. Like for feasibility pumps for MINLPs, the proposed method RECIPE also solves NLPs, although the main idea is clearly MIP-inspired. Metaheuristics and (variants of) local search methods are also applied as stand-alone solution procedures both in academic as well as in commercial fields. See, e.g., the OptQuest/NLP solver presented in [41] or LocalSolver described in [3].

However, in contrast to the number of generalized MIP heuristics and metaheuristics, less research has been done to develop primal heuristics that are genuinely tailored for MINLPs. Examples for the latter are the Undercover heuristic of [5], which is an MIP-based MINLP heuristic that does not carry over an idea of a well-known MIP heuristic or the rounding procedures given in [29] that also do not have MIP counterparts.

For a more comprehensive and detailed recent survey on the literature about primal heuristics for MINLPs we refer the interested reader to [4]. Summarizing the brief review above one can make the following observations: many methods are direct generalizations of heuristics from mixed-integer linear programming and a lot of methods use mixed-integer linear solvers as subroutines. Hence, not many general-purpose heuristics for MINLP exist that are based on concepts from continuous optimization. Our contribution is to develop and test such an NLP-based primal heuristic.

To be more precise we make use of different MPEC reformulations of a given MINLP by replacing integrality constraints with suitable complementarity constraints. Due to their inherent lack of constraint regularity, these MPECs cannot be solved directly by local NLP solvers in a rigorous way. Instead, different regularization schemes are utilized and a tailored solution procedure is applied that successively solves these regularized problems. If this sequence of regularized problems is chosen carefully, the series of obtained solutions converges to a stationary point of the MPEC and, thus, to a feasible point of the original MINLP. Our working hypothesis is that this yields fast methods since we rely on local NLP solvers instead of global MIP solvers. On the contrary, local NLP solvers are typically not as reliable as MIP solvers. For example, NLP solvers may terminate in practice without a feasible solution even if the problem is feasible. This is why we also apply additional algorithmic techniques to increase the reliability of the solution process.

The idea of using continuous reformulations of discrete-continuous problems is not new. One scientific field in which this concept is frequently applied is chemical engineering. For instance, [36] studies continuous reformulations of generalized disjunctive programs (GDPs). The authors propose different reformulations of GDPs and apply them to problems from process synthesis. Similar techniques for GDP-type MINLPs are discussed in [24] and in [25], where the authors study the special reformulation using a regularized version of the Fischer–Burmeister function and apply the resulting regularization scheme to parts of the general-purpose instance library MINLPLib and to problems from the synthesis of distillation systems. Moreover, very recently, a new relaxation scheme for binary variables has been proposed in [44], where the authors develop a tailored ADMM method for integer programming.

Recently, MPEC-based reformulations of nonconvex MINLPs have also been considered in the field of gas transport optimization; see [31,33,34]. For general information about MPECs we refer to the textbook [27] and to the survey paper [2]. The latter also contains a detailed computational study of different regularization schemes for MPECs that are solved with different NLP solvers.

The rest of the paper is structured as follows: in Sect. 2 we present a simple reformulation of MINLPs as MPECs, review standard MPEC regularization strategies, and discuss a general solution framework for MPECs from the literature. Section 3 then builds upon these concepts, presents enhanced techniques to increase the reliability of the solution process, and states the overall primal heuristic. This method is then tested extensively in Sect. 4. We test different NLP solvers, MPEC regularizations, and the impact of the enhanced techniques. Moreover, we also compare the proposed method with different state-of-the-art implementations for primal heuristics for MINLPs. The paper closes with some concluding remarks and some comments on possible future work in Sect. 5.

2 Problem statement, MPEC reformulations, and regularization

In this paper we consider 0-1-MINLPs of the form

$$\min_{x,z} \quad f(x,z) \tag{1a}$$
$$\text{s.t.} \quad g(x,z) \geq 0, \tag{1b}$$
$$x \in \mathbb{R}^n, \tag{1c}$$
$$z \in \{0,1\}^m, \tag{1d}$$

for which we allow the objective function $f : \mathbb{R}^n \times \{0,1\}^m \to \mathbb{R}$ and the constraints $g : \mathbb{R}^n \times \{0,1\}^m \to \mathbb{R}^k$ to be nonlinear and nonconvex. We denote the feasible set of (1) by Ω, i.e.,

$$\Omega := \left\{(x,z) \,\middle|\, g(x,z) \geq 0,\ x \in \mathbb{R}^n,\ z \in \{0,1\}^m\right\} \subseteq \mathbb{R}^n \times \{0,1\}^m.$$

In what follows we assume that Ω is bounded such that Problem (1) is decidable. MINLPs form a very challenging class of optimization problems because they combine both the difficulties arising from the integrality constraints (1d) as well as from the nonlinearities in the objective function (1a) and the constraints (1b).

Many approaches that tackle MINLPs like (1) in an exact or heuristic way try to get rid of one of these difficulties. In this paper we propose a general-purpose primal heuristic that is based on special types of continuous reformulations of (1) in which the integrality constraints (1d) are replaced by certain constraints only involving continuous variables or by penalizing violations of (1d) in suitably chosen penalty terms that are added to the objective function f.

To this end, we apply a two-stage reformulation strategy. First, we replace the MINLP (1) by an equivalent mathematical program with complementarity or equilibrium constraints (MPCC or MPEC, for short). Since MPECs are known to violate constraint qualifications (CQs) that are typically required in nonlinear optimization, we then, in a second step, replace the MPEC by a regularized problem for which the relevant CQs hold.

For the first step we make the obvious observation that the integrality constraint $z_i \in \{0,1\}$ is equivalent to the constraints

$$z_i(1 - z_i) = 0, \quad z_i \in [0,1] \subseteq \mathbb{R}. \tag{2}$$

Thus, we can replace the MINLP (1) by the model

$$\min_{x,z} \quad f(x,z) \tag{3a}$$
$$\text{s.t.} \quad g(x,z) \geq 0, \tag{3b}$$
$$x \in \mathbb{R}^n, \tag{3c}$$
$$0 \leq z_i \perp 1 - z_i \geq 0 \quad \text{for all } i = 1,\ldots,m, \tag{3d}$$

where for variables $\alpha, \beta \in \mathbb{R}$ the notation $0 \leq \alpha \perp \beta \geq 0$ means that both variables are required to be nonnegative and at least one of them is at its bound, i.e., $\alpha = 0$ or $\beta = 0$. It is easy to see that the feasible set Ω_C of (3) coincides with the feasible set Ω of the original MINLP. Constraint (3d)

is a so-called complementarity constraint, i.e., Model (3) is an MPCC, or, in a more general notion, an MPEC. These models are of increasing importance both because of their numerous applications in, e.g., economics and engineering, and their relevance in mathematical optimization itself—for instance due to their relation to bilevel optimization. We refer the interested reader to the monograph [27] and the references therein. By replacing the complementarity constraint (3d) by the NLP-like constraints of Type (2) the resulting model looks, at a first glance, like a standard NLP. However, the applied reformulation of the integrality constraints comes at the price that standard CQs like the Mangasarian–Fromovitz (MFCQ) or the Linear Independence CQ (LICQ) fail to hold at every feasible point of (3); see [45]. This yields the fact that standard NLP solvers typically cannot be applied without losing their convergence theory that often relies on the satisfaction of, e.g., MFCQ or LICQ. See, e.g., [14,19,42,43] for the convergence properties of the local solvers Ipopt, SNOPT, and CONOPT4.

A remedy is the application of tailored regularization schemes for MPECs. All of these schemes share a common structure. In our presentation of these schemes, we follow Hoheisel et al. [21]. These schemes solve regularized problems $\mathrm{R}(\tau)$ where the regularization depends on a regularization parameter $\tau \geq 0$. For each problem with $\tau > 0$ standard CQs hold again. Starting with an initial regularization parameter τ_0 they solve the problem $\mathrm{R}(\tau_0)$, compute a new parameter $\tau_1 < \tau_0$, solve $\mathrm{R}(\tau_1)$ (typically by using the old solution as initial value), etc. Specific instantiations differ in the concrete update rule for the regularization parameter and the concrete regularization strategy yielding the problems $\mathrm{R}(\tau_k)$ for $k = 0, 1, \ldots$ Algorithm 1 displays a generic version of this algorithm. For the analysis of these schemes, it is necessary to introduce tailored

Algorithm 1 A generic MPEC regularization algorithm

1: Choose initial values $(x^0, z^0) \in \mathbb{R}^n \times [0,1]^m$, an initial regularization parameter $\tau_0 > 0$, a minimum regularization parameter $\tau_{\min} > 0$, and a constant $\sigma \in (0,1)$.
2: Set $k = 0$.
3: **while** $\tau_k > \tau_{\min}$ **do**
4: Solve the regularized problem $\mathrm{R}(\tau_k)$ using (x^k, z^k) as initial value. Let (x^{k+1}, z^{k+1}) be the solution.
5: Update $\tau_{k+1} \leftarrow \sigma \tau_k$ and set $k \leftarrow k+1$.
6: **end while**
7: **return** last iterate (x^k, z^k).

constraint qualifications and stationarity concepts. In this article, we do not discuss these topics further but briefly mention the relevant theoretical results for the used schemes. For details, we again refer to [21]. In the remainder of this section we discuss three well-studied regularization strategies that yield different specific instantiations of Algorithm 1.

Among the first regularization techniques that have been proposed for MPECs was [35]. Applied to our situation the regularization discussed by the author reads

$$\min_{x,z} \quad f(x,z) \tag{4a}$$
$$\text{s.t.} \quad g(x,z) \geq 0, \tag{4b}$$
$$x \in \mathbb{R}^n,\ z \in \mathbb{R}^m, \tag{4c}$$
$$z_i(1-z_i) \leq \tau,\ z_i \in [0,1] \quad \text{for all } i = 1,\ldots,m. \tag{4d}$$

Obviously, the feasible set $\Omega_S(\tau)$ of (4) depends on the regularization parameter $\tau \geq 0$. It holds $\Omega_S(\tau_1) \subset \Omega_S(\tau_2)$ for $\tau_1 < \tau_2$ and the original MPCC feasible set Ω_C is obtained with $\tau = 0$, i.e., $\Omega = \Omega_C = \Omega_S(0)$. To the best of our knowledge, the strongest convergence result for the regularization of Scholtes is shown in [21]: if for a given series of regularization parameters $(\tau_k)_k \searrow 0$ the sequence of solutions of the regularized problems converges to a point (x^*, z^*) for which the MPEC-MFCQ holds, then the latter point is a C-stationary point of the original MPEC.

In addition to the component-wise regularization of Constraint (4d), variations thereof like

$$\sum_{i=1}^{m} z_i(1-z_i) \leq \tau,\ z_i \in [0,1] \quad \text{for all } i = 1,\ldots,m$$

or

$$z_i(1-z_i) = \tau,\ z_i \in [0,1] \quad \text{for all } i = 1,\ldots,m$$

can be found in the literature; see, e.g., [2].

Another frequently used strategy for replacing the complementarity constraints (3d) is using so-called NCP functions $\varphi : \mathbb{R}^2 \to \mathbb{R}$ that satisfy $\varphi(a,b) = 0$ iff $a, b \geq 0$ and $ab = 0$; see, e.g., [37]. A popular NCP function is the Fischer–Burmeister function

$$\phi(a,b) := a + b - \sqrt{a^2 + b^2}$$

that has been proposed by Fischer in [16]. Its drawback is the non-differentiability in $(a,b) = (0,0)$, which is typically resolved by regularizing the square root yielding

$$\phi(a,b;\tau) := a + b - \sqrt{a^2 + b^2 + \tau}, \quad \tau > 0.$$

Applied to $a = z_i$ and $b = 1 - z_i$ we obtain

$$\phi(z_i, 1-z_i; \tau) = 1 - \sqrt{z_i^2 + (1-z_i)^2 + \tau}$$

and, finally, the regularized problem

$$\min_{x,z} \quad f(x,z) \tag{5a}$$
$$\text{s.t.} \quad g(x,z) \geq 0, \tag{5b}$$
$$x \in \mathbb{R}^n,\ z \in [0,1]^m, \tag{5c}$$
$$\sqrt{z_i^2 + (1-z_i)^2 + \tau} \geq 1 \quad \text{for all } i = 1,\ldots,m. \tag{5d}$$

This regularization has also been applied for reformulating discrete-continuous optimization problems in [25].

Both the Scholtes and the Fischer–Burmeister approach used additional constraints involving relaxed versions of the original binary variables to cope with the complementarity constraints (3d). The last regularization that we apply in this paper follows a different idea: It completely removes the complementarity constraints—i.e., the integrality conditions—from the set of constraints and penalizes their violation in additional penalty terms in an extended objective function. This yields

$$\min_{x,z} \quad f(x,z) + \frac{1}{\tau}\sum_{i=1}^{m} z_i(1-z_i) \tag{6a}$$

$$\text{s.t.} \quad g(x,z) \geq 0, \tag{6b}$$

$$x \in \mathbb{R}^n,\ z \in [0,1]^m. \tag{6c}$$

This is somehow the strongest regularization of the discussed ones since it obviously holds $\Omega_P \supset \Omega_C$ for the feasible set Ω_P of (6). This approach has been proposed in [22] where the authors show that the corresponding sequence of stationary points generated by Algorithm 1 converges to a C-stationary point if the limit point is feasible for the original MPEC and if it satisfies the MPEC-LICQ.

Up to this point we introduced three different instantiations of Algorithm 1 depending on which regularization out of (4), (5), and (6) is used. In the next section we describe enhanced algorithmic techniques that lead to a more reliable algorithm for solving MPEC reformulations of MINLPs.

3 Enhanced solution techniques and the entire algorithm

From a theoretical point of view, Algorithm 1 equipped with one of the regularized problems (4), (5), or (6) already constitutes a proper algorithm for computing (suitably generalized) stationary points of the MPEC at hand. In other words, it already forms a primal heuristic for general, i.e., nonconvex, MINLPs. However, our first numerical experiments showed that the overall performance and reliability of the algorithm can be increased further by extending the techniques discussed in Sect. 2 by the enhanced algorithmic strategies described in the following sections.

The final algorithm, for which we then present computational results in Sect. 4, is stated in Sect. 3.3.

3.1 Re-initialization and variable fixing

An iterative tightening of the regularization parameter within the constraint-based regularization schemes yields disjoint feasible regions for every binary variable after sufficiently many tightenings. Consider, for instance, the regularization scheme $z_i(1-z_i) \leq \tau$ with $z_i \in [0,1]$ of Scholtes. For $\tau \geq 1/4$ we simply obtain the NLP relaxation, i.e., the feasible region is completely described by $z_i \in [0,1]$. However, for $\tau < 1/4$ we obtain two disjoint regions—one containing $z_i = 0$ and the other containing $z_i = 1$. For local NLP solvers, it is folklore knowledge that this property

of the feasible set often leads to convergence failures; especially if the NLP algorithm has to jump from one side to the other.

Since we use the solution of the preceding NLP to initialize the next problem in our solution algorithm, it may be the case that we enter "the wrong side" of the later splitted feasible region, making it unlikely that later iterations will be able "to correct" this issue. This observation has already been reported by [25] for the Fischer–Burmeister regularization of MINLPs. Intuitively, this can happen, if, e.g., the model contains a constraint of the type $0 \leq x \leq Cz$, where x is a continuous and z a binary variable. It can be advantageous to set z to a value close to 1 to allow x to be nonzero. If then, however, the choice of $z = 1$ does not lead to a feasible solution, a simple rounding strategy will fail. This is why an initialization $z = 0$ can be helpful. We follow the ideas of [25] for resolving the problem: Whenever an NLP $\mathrm{R}(\tau_k)$ cannot be solved during the course of Algorithm 1, we apply the following *re-initialization strategy*:

We determine all binary variables $z_i, i \in I \subseteq \{1, \ldots, m\}$, for which the regularization constraint, i.e., (4d) for the regularization scheme of Scholtes or (5d) for the Fischer–Burmeister scheme, is violated. Let $z_i^k \in [0, 1]$ for $i \in I$ be these infeasible values. We then try to solve the failed NLP $\mathrm{R}(\tau_k)$ again but with $z_i, i \in I$, initialized to $\hat{z}_i$ as follows:

$$\hat{z}_i = \begin{cases} 0, & \text{if } z_i^k \geq 0.5, \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

If $\mathrm{R}(\tau_k)$ still cannot be solved after re-initialization such that we again have infeasible variable values z_j^k for some $j \in J \subseteq \{1, \ldots, m\}$, we fix the corresponding variables z_j following the rule given in (7). The intuition behind this is that sometimes just setting the starting point is not enough from preventing a solver to converge to a nonintegral solution. That is why, as a last resort, we try to fix the "offending" variables.

If this *variable fixing strategy* or the above mentioned re-initialization strategy yields a feasible solution of $\mathrm{R}(\tau_k)$ we revert the applied initializations and/or variable fixations and continue in Line 5 of Algorithm 1.

The described strategies can be applied in a straightforward manner both for the regularization scheme of Scholtes and for the Fischer–Burmeister scheme. In these cases, the violation of regularization constraints leads to natural choices for the re-initialization candidates $i \in I$. The application to the penalty regularization scheme (6) needs some additional explanations. Since this regularization does not possess any explicit integrality constraints, we need a different idea for computing the re-initialization candidates. In our implementation we simply consider the regularization constraints (4d) as a proxy model for computing the index set $I \subseteq \{1, \ldots, m\}$. That is, we apply the re-initialization strategy whenever we encounter a solution of $\mathrm{R}(\tau_k)$ violating

$$z_i^k(1 - z_i^k) \leq \tau_k \quad \text{for some } i = 1, \ldots, m.$$

Afterward, we check whether these conditions are satisfied and, if not, apply the variable fixing strategy as described above.

3.2 Regularization parameter backtracking

Algorithm 1 can be seen as a homotopy method solving a series of regularized NLPs $\mathrm{R}(\tau_k)$ with the goal to obtain an approximate solution of $\mathrm{R}(0)$. Our preliminary numerical results showed that such kind of methods can fail in practice if the parameter τ is updated too aggressively. This leads us to the following extension of our "backup strategies" described in the preceding section.

Whenever both the re-initialization and the variable fixing does not yield a feasible solution for $\mathrm{R}(\tau_k)$ a reason for this failure may also be a too aggressive update from τ_{k-1} to τ_k. Thus, we enter a backtracking phase in which we backtrack the regularization parameter via

$$\tau_{k,\ell} = \frac{\tau_{k,\ell-1}}{\sigma^{1/2^\ell}}, \quad \tau_k^0 = \tau_k, \quad \ell = 1, \ldots, \ell^{\max},$$

and solve the corresponding problem $\mathrm{R}(\tau_{k,\ell})$ until we obtain a feasible solution or the (user-specified) backtracking iteration limit $\ell^{\max}$ is reached. In case that we find a feasible solution of $\mathrm{R}(\tau_{k,\bar{\ell}})$ for some $\bar{\ell}$ we resume the overall algorithm for $\tau = \tau_k$ and solve $\mathrm{R}(\tau_k)$ initialized with the solution of $\mathrm{R}(\tau_{k,\bar{\ell}})$.

3.3 The final algorithm

We now collect all enhanced solution techniques and combine them with the generic Algorithm 1 of Sect. 2. In order to give a precise but neat exposition we present the entire algorithm in a state-machine-like manner. The states of the algorithm are the following:

Basic	The solution procedure is applied as in Algorithm 1, i.e., the regularized problems $\mathrm{R}(\tau_k)$ are solved (Line 4) and the regularization parameters are updated as given in Line 5.
Re-Init	The NLP solver failed to solve a regularized problem $\mathrm{R}(\tau_k)$. In this state, re-initialization candidates are determined and re-initialized as described in Sect. 3.1.
Fix	The NLP solver failed to solve the re-initialized problem of state Re-Init. The variable fixing strategy of Sect. 3.1 is applied and the resulting NLP is solved.
Backtrack	The NLP solver failed to solve the problem with fixed variables in state Fix. A regularization parameter backtracking is applied as described in Sect. 3.2 until a feasible solution has been found or a backtracking iteration limit is reached.

The overall solution procedure is illustrated in Fig. 1 together with the main state transitions. In addition to these transitions we have some "global" transitions: Whenever we have computed a new feasible point to any $\mathrm{R}(\tau)$ we check whether this solution is also feasible for the original MINLP. If this is the case, we stop with an MINLP feasible solution. Moreover, we terminate if a maximum number of NLPs has been solved or if a given time limit is reached.

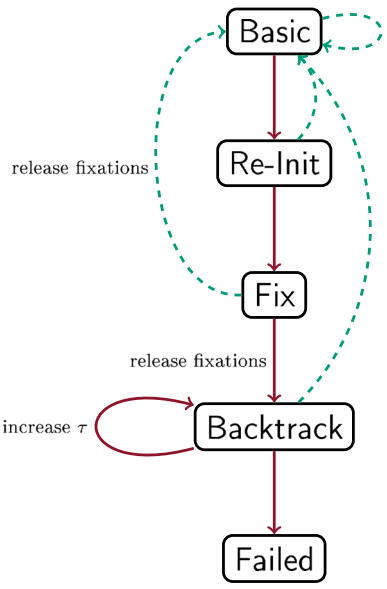

Fig. 1 The entire algorithm. Solid arcs denote state transitions after failures; dashed arcs denote transitions if an NLP has been solved successfully

4 Computational results

In this section we present an extensive computational study of the family of methods described in the last two sections. To be more precise, we evaluate the impact of the following aspects:

1. The chosen NLP solver: We test the reduced gradient method CONOPT4 [13–15], the SQP code SNOPT [19], and the interior-point solver Ipopt [42].
2. The chosen regularization scheme: We compare the regularization of Scholtes (4), the Fischer–Burmeister scheme (5), and the penalty approach (6).
3. The impact of the enhanced solution techniques from Sect. 3, i.e., re-initialization, variable fixing, and regularization parameter backtracking.
4. The success rate and performance of the different regularization strategies and NLP solvers in dependence of specific properties of the MINLP models like the overall number of variables, the number of binary variables, or the (non)convexity of the problem.

The analysis of these aspects is given in Sect. 4.1. Finally, we compare our methods with different other state-of-the-art primal heuristics for MINLP in Sect. 4.2.

4.1 Analysis of the proposed methods

We implemented the algorithm as well as all regularizations in C++11 as a GAMS solver using the GAMS Expert-level API with GAMS 24.5.6 [18]. The code has been compiled with gcc 4.8.4. All computations have been executed on a 48-core machine with AMD Opteron^{TM} 6176 SE processors running at 2300 MHz with 264 GB RAM.

Our test set consists of 662 instances from the MINLPLib2 library. Out of the entire MINLPLib2 library of 1388 instances we excluded all instances that only contain continuous variables, that contain general, i.e., non-binary, integer variables, SOS-1 sets, or semicontinuous variables. This results in a set of 681 test instances from which we excluded additional 19 instances for which none of the local NLP solvers CONOPT4, SNOPT, and Ipopt could solve the NLP relaxation.

Throughout this section we use log-scaled performance profiles as proposed in [12] to compare running times and solution quality. Following [23], the latter is measured by the primal gap that is defined by

$$\text{gap} = \frac{b_\text{p} - b_\text{d}}{\inf\{|z| : z \in [b_\text{d}, b_\text{p}]\}}, \tag{8}$$

where b_p is the primal and b_d is best known solution according to the MINLPLib2 website, respectively. Obviously, (8) is only valid for minimization problems but can be easily adapted for maximization problems. Two special cases merit particular mention: It holds that gap $= +\infty$ whenever $b_\text{d} < 0 \leq b_\text{p}$, and we set gap $= 0$ if $b_\text{d} = b_\text{p} = 0$.

In all computations we used the standard GAMS initialization of all primal variables to 0 except for the cases in which the instance itself contains a non-zero starting point. In these cases we used the provided point. The regularization parameter τ is always initialized to 100. By taking a closer look at the Scholtes regularization constraint (4d) and the Fischer–Burmeister regularization constraint (5d) one can then easily see that solving the initial regularized MPEC directly corresponds to solving the NLP relaxation of the underlying MINLP. The time limit is set to 15 min and the integrality tolerance is chosen to be 10^{-5}.

We now turn to the presentation and discussion of the numerical results. We first discuss the results of the constraint-based regularization schemes in detail and then turn to the discussion of the results for the penalty-based reformulation. In Table 1 we see the number of instances that are solved to feasibility for all tested combinations of constraint-based schemes (first column) and NLP solvers (second column). The third column (Basic) states the number of instances for which a feasible solution can be found by using the plain version of Algorithm 1 without using any additional techniques as described in Sect. 3. Using only the Basic state of the algorithm, the least successful combination of all constraint-based strategies (Scholtes' reformulation solved with SNOPT) only solves 287 instances (i.e., 43.35%) whereas the most successful constraint-based strategy (Scholtes' reformulation solved with Ipopt) computes a feasible solution for 372 instances (i.e., 56.19%). Moreover, it can be seen that the rate of success depends more strongly on the choice of the NLP solver than on the choice of the constraint-based reformulation scheme. The fourth column then states the number of additional instances for which a feasible solution can be computed if

Table 1 Number of feasible solutions found for different combinations of constraint-based MPEC reformulations and NLP solvers in different states of the algorithm

Regularization	Solver	Basic	Re-Init	Fix	Backtrack	Total
Scholtes	CONOPT4	365	77	13	12	467
Scholtes	Ipopt	372	67	5	6	450
Scholtes	SNOPT	287	112	17	5	421
Fischer–Burmeister	CONOPT4	354	80	11	18	463
Fischer–Burmeister	Ipopt	362	65	4	11	442
Fischer–Burmeister	SNOPT	294	90	21	10	415

the Re-Init state is used as well. The numbers vary between 65 and 112 and depend, again, more on the choice of the NLP solver than on the specific constraint-based regularization scheme. The fifth and sixth column finally state the number of additional instances that can be solved to feasibility if the Fix and Backtrack states are used as well. The numbers for Fix and Backtrack states are comparable and significantly lower then those obtained for Re-Init. In total, Table 1 shows that CONOPT4 solves the largest number of instances (70.5% for the Scholtes scheme and 69.9% for the Fischer–Burmeister reformulation). The least successful solver is SNOPT yielding to 63.6% (Scholtes) respectively 62.7% (Fischer–Burmeister) feasible solution.

We further remark that some instances fail due to internal program errors in CONOPT4 and Ipopt.[1] CONOPT4 leads to the largest number of such errors (35 in total), whereas Ipopt only fails in 5 cases. For both solvers it can be seen that they crash on certain families of instances of the MINLPLib2. These families are mainly the crudeoil_lee and telecomsp instances for CONOPT4 and the faclay instances for Ipopt. Using SNOPT, no such errors occur.

We now study the behavior of different constraint-based MPEC reformulation schemes solved with different NLP solvers in more detail. In Fig. 2, bar plots are given that illustrate how many feasible solutions have been found in which iteration and how many instances failed in each iteration. The first insight is that the behavior is qualitatively comparable for all constraint-based reformulations and all NLP solvers.

The most successful iteration is Iteration 5. This can be easily explained: Given an initial regularization parameter τ_0 and assuming that the algorithm stays in the Basic state, the update rule for τ leads to a regularization parameter τ_5 such that (both for the reformulation scheme of Scholtes and the Fischer–Burmeister scheme) the

[1] For CONOPT4, the failed instances are crudeoil_lee1_08, crudeoil_lee2_05, crudeoil_lee2_06, crudeoil_lee3_05, crudeoil_lee3_06, crudeoil_lee4_10, nuclear25a, telecomsp_njlata, telecomsp_pacbell for the reformulation scheme of Scholtes (9); crudeoil_lee1_08, crudeoil_lee3_06, crudeoil_lee3_10, gasprod_sarawak81, nuclear25a, sepasequ_convent, telecomsp_njlata, telecomsp_pacbell for the Fischer–Burmeister reformulation scheme (8) and crudeoil_lee1_06, crudeoil_lee1_09, crudeoil_lee2_06, crudeoil_lee2_07, crudeoil_lee2_08, crudeoil_lee2_09, crudeoil_lee2_10, crudeoil_lee3_06, crudeoil_lee3_08, crudeoil_lee4_05, crudeoil_lee4_06, crudeoil_lee4_07, crudeoil_lee4_08, nuclear49a, nuclear49b, squfl025-040persp, telecomsp_njlata, telecomsp_pacbell for the penalty-based reformulation (18). For Ipopt, the failed instances are faclay60, faclay70, faclay80 for the reformulation scheme of Scholtes (3), whereas it fails on faclay75 for the Fischer–Burmeister reformulation and on faclay33 and for the penalty-based reformulation.

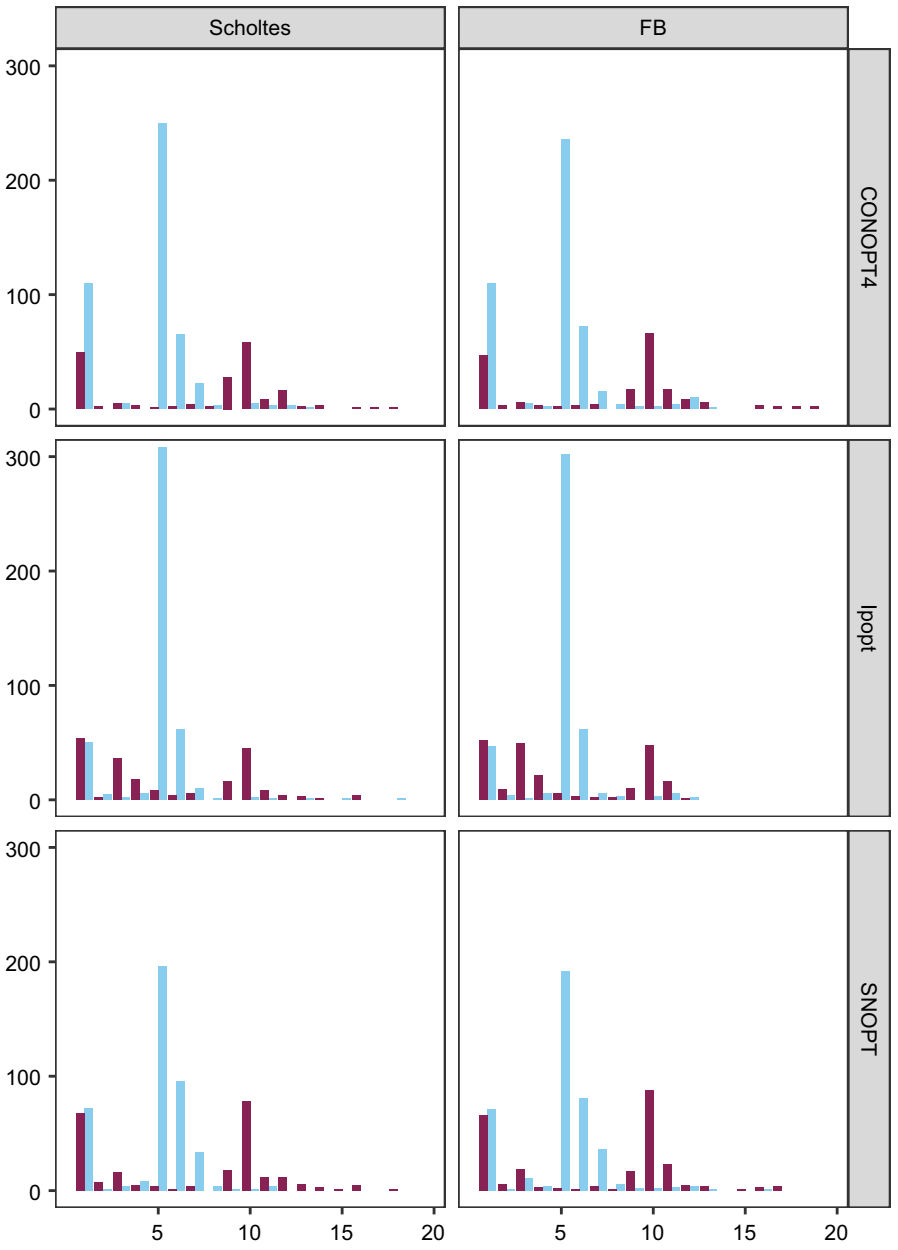

Fig. 2 Bar plots for all combinations of constraint-based MPEC reformulations and NLP solvers: number of instances (y-axes) that are solved to feasibility (blue) or fail (red) in iteration k (x-axes) (color figure online)

satisfaction of the respective regularization constraints leads to an approximation of an integer value better than the given integrality tolerance. The second-most successful iteration is the initial one. This is a little bit more surprising since it says that there

Table 2 Number of feasible solutions found for the penalty-based reformulation and different NLP solvers in different states of the algorithm

Regularization	Solver	Basic	Re-Init	Fix	Backtrack	Total
Penalty	CONOPT4	146	316	5	7	474
Penalty	Ipopt	129	335	3	–	467
Penalty	SNOPT	133	342	1	–	476

is a significant amount of instances for which the NLP solution of the initial MPEC regularization (i.e., the NLP relaxation) directly yields an integer feasible solution of the original MINLP. In the constraint-based reformulations, this roughly corresponds to the solution of the NLP relaxation as the regularization parameters are chosen in such a way that the constraints cannot be active in any solution. Thus, we can see that even though just solving the NLP relaxation locally can yield a feasible solution, our methods improve upon this simple procedure. We note that in contrast to LP relaxations for MIPs such a solution does not need to be the optimal solution as we only solve the NLP to local optimality. This also makes it difficult to give a sound theoretical reasoning for which models we can expect to find a good solution this way. Comparing the first iteration with respect to the used NLP solver, however, one clearly sees that this (desirable) behavior can most distinctly be seen for CONOPT4 and can be seen the least for Ipopt. Our hypothesis is that this difference in the solvers stems from the different solution techniques they employ. It seems intuitively clear that an interior point method as employed by Ipopt should perform worse in finding solutions that "accidentally" are tight with respect to the bounds than, say, a reduced gradient method as used by CONOPT4, which starts from a feasible solution.

What can also be seen in the bar plots of Fig. 2 is the success rate of the re-initialization strategy and the variable fixing strategy in Iteration 6 and 7, respectively. The small amount of feasible solutions found in the backtracking phase is finally blurred over the Iterations $k \geq 8$. We also tried larger backtracking iteration limits. However, this does not yield significant improvements of the overall algorithm.

Regarding the negative cases (red bars) we have two main peaks: the first iteration and Iteration 10. For the former it is simply the case that the initial MPEC regularization cannot be solved whereas the reason for the latter is simply the number of maximal backtracking iterations (5 in our computations).

We now turn to the discussion of the results of the penalty-based reformulation scheme. As explained in Sect. 3 the overall strategy is somehow different since we have to use a proxy model in order to switch between different states of the algorithm. This can be also seen in the results given in Table 2: the number of feasible solutions found in the Basic state is significantly smaller than for the constraint-based reformulations and the number of feasible instances found using the re-initialization strategy is significantly larger. This, however, is not the case because the Basic state fails more often for the penalty-based case but because the Re-Init state is activated by far more aggressively than for the constraint-based schemes. Moreover, one clearly sees that the additional states, i.e., the Fix and the Backtrack state, do not give much more feasible instances. The bar plots of Fig. 3 approve these observations. After finding always

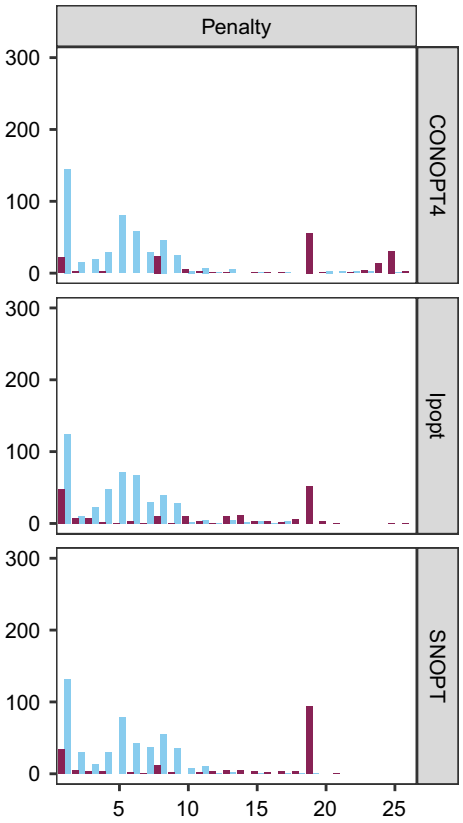

Fig. 3 Bar plots for all combinations of penalty-based MPEC reformulation and NLP solvers: number of instances (y-axes) that are solved to feasibility (blue) or fail (red) in iteration k (x-axes) (color figure online)

more than 100 feasible solutions in the initial penalty iteration, the success rates are more blurred around Iteration 5, in contrast to the significant peak at Iteration 5 for the constraint-based approaches. Comparing these results with those of the constraint-based formulations it is especially visible that the number of feasible solutions found in the first iteration is much larger. In summary, these observations lead to the fact that more feasible solutions are found in early iterations. However, the number of failures in late iterations is larger as well. In particular, the maximum red peak has moved from Iteration 10 (resulting from the maximum backtracking iterations limit) in Fig. 2 to Iteration 19. The reason is that the lower bound $\tau_{\min}$ for the regularization parameter is reached.

Altogether, the most successful method, i.e., the combination of the penalty reformulation solved with SNOPT computes 72% of all instances to feasibility. Taken over all regularization schemes and NLP solvers we are able to solve 597 out of 662 instances, i.e., 90% to feasibility.

We now consider running times and first discuss the performance profile given in Fig. 4. We see that all MPEC regularizations solved with Ipopt lead by far to the slowest method, followed by the constraint-based reformulations solved with SNOPT. The fastest NLP solver is CONOPT4, which is also comparable to SNOPT applied to the

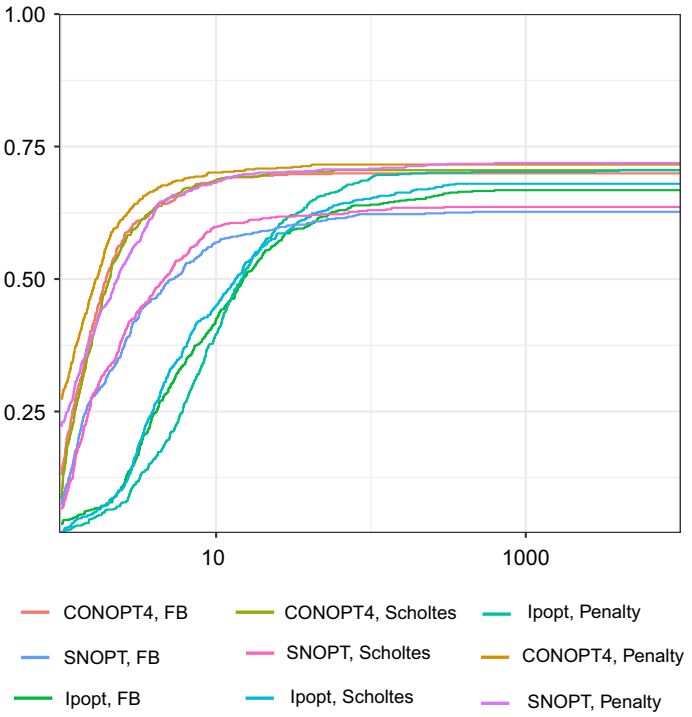

Fig. 4 Log-scaled performance profiles of running times for all 9 combinations of MPEC reformulation schemes and NLP solvers

penalty-based reformulation. Thus, we see that it is important to study the performance of the chosen NLP solver in dependence of the formulation at hand. Especially for SNOPT, the specific formulation is of great importance: Not only are the penalty-based reformulations solved faster, also the overall success rate strongly depends on the formulation. While the constraint-based formulations solved with SNOPT lead to the worst results in terms of overall feasible solutions found, the penalty-based reformulation solved with SNOPT is the most successful variant of all combinations.

Empirical distribution functions for all combinations are given in Fig. 5. Only considering the variants performing best, one can see that approximately 50% of all instances have been solved to feasibility within 1 s and more than 60% of all instances have been solved in 10 s. Recalling that the most successful method solves slightly more than 70% of all instances to feasibility, this gives a clear guidance on how much time should be reserved for the described methods if they are used as primal heuristics in a global mixed-integer nonlinear optimization solver: in order to not block valuable resources for other processes of the solver, a reasonable time limit would be approximately 10 s.

Lastly, we discuss the quality of the solution in terms of their objective value. In Fig. 6 we see the empirical distribution functions for the primal gap of all tested methods. The overall result is that all methods are quite comparable with respect to this measure. Interestingly, Ipopt—which was by far the slowest method—computes

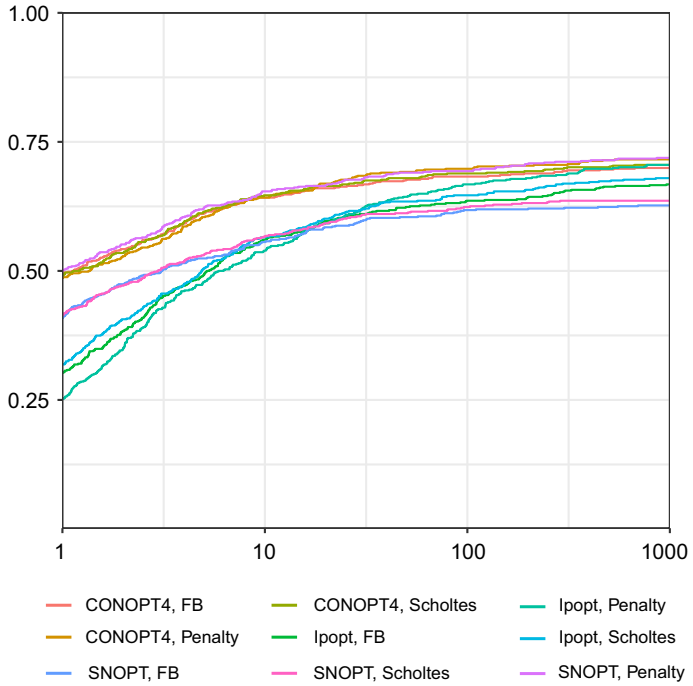

Fig. 5 Empirical distribution functions of absolute running times (x axes; in s) for all 9 combinations of MPEC reformulation schemes and NLP solvers

Table 3 Number of instances that are solved with an objective value equal to the best known objective value (with respect to the primal tolerance 10^{-5})

	CONOPT4	Ipopt	SNOPT
Scholtes	47	61	37
Fischer–Burmeister	47	64	42
Penalty	64	81	52

the feasible points with the best objective values: For 25% of the 662 instances Ipopt applied to the penalty regularization yields a feasible point with a primal gap of no more than 10^{-2}. What we also see is that, again, the constraint-based reformulations solved with SNOPT perform worst. Finally, Table 3 displays the number of feasible solutions found with an objective value equal to the best known solution (as they can be found on the MINLPLib2 website). Hence, the best combination, i.e., the penalty model solved with Ipopt, yields 81 best-known solutions, which corresponds to 12.4%.

We now turn to the question whether specific combinations of MPEC regularization and NLP solver are preferable for special types of MINLPs. If this is the case, a given model could be analyzed a priori and the most suitable parameterization of the heuristic can then be applied afterward. We start by analyzing the impact of the model's overall size (in terms of the total number of variables) and the number of binary variables. To this end, we divide the overall test set into different classes. On the one hand, we have the instances with [0, 75], (75, 200], (200, 500], (500, 1.5×10^3], and

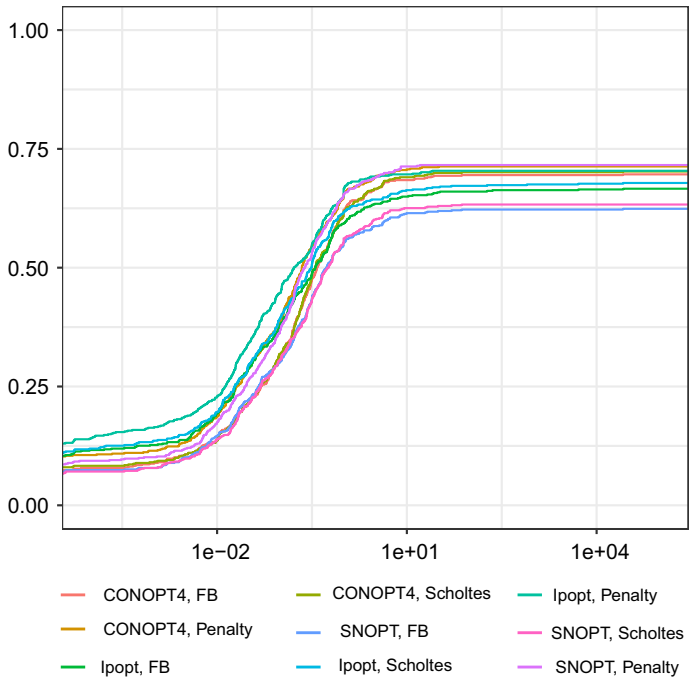

Fig. 6 Empirical distribution function of the primal gap for all 9 combinations of MPEC reformulation schemes and NLP solvers

Table 4 Number of instances in the considered classes

	# binary variables				
	[0, 25]	(25, 50]	(50, 100]	(100, 200]	(200, ∞)
# variables					
[0, 75]	84	40	17	0	0
(75, 200]	30	31	46	47	0
(200, 500]	10	22	37	39	33
(500, 1.5×10^3]	20	11	18	23	70
(1.5×10^3, ∞)	12	10	6	17	39

(1.5×10^3, ∞) variables. On the other hand, we have instances with [0, 25], (25, 50], (50, 100], (100, 200], and (200, ∞) binary variables. In the following, we consider the cross product of these classes. The number of instances per class is given in Table 4.

In Table 5 we compare the running times for all nine combinations of MPEC regularization and NLP solver in dependence of the total number of MINLP variables and the number of binary variables. Thus, for every combination of the two types of classes described above, we have a 3×3 table block where the 3 rows correspond (always from top to bottom) to the Scholtes, the Fischer–Burmeister, and the penalty reformulation and where the 3 columns correspond (always from left to right) to CONOPT4, Ipopt,

and SNOPT. Every table entry corresponds to the respective average running time. A dash indicates that there are no instances for a specific combination of the overall number of variables and the number of binary variables. An asterisk indicates that the specific combination of instance classes contains an instance that could not be solved by the considered combination of NLP solver and MPEC regularization strategy. The fastest method per 3×3 table block is printed in bold. We first study the impact of the used NLP solver. First of all, it can be seen that Ipopt is never the fastest method (in average). This is in line with the results shown in Fig. 4. Moreover, CONOPT4 is the fastest solver for smaller numbers of (binary) variables, whereas SNOPT is clearly faster for larger numbers of (binary) variables. The choice of the regularization method is also of interest. Here, a clear conclusion can only be drawn for large numbers of (binary) variables. In these classes of instances, the penalty regularization turns out to yield the fastest method.

Finally, we remark that for fixed NLP solver and fixed MPEC regularization strategy, no monotonicity can be seen for the running times in dependence of the total number of variables or binary variables. This indicates that the test set is quite heterogeneous and that the hardness of the instances cannot be fully characterized by the number of variables. Thus, we also analyzed the aspects whether the MINLP is convex or nonconvex or whether it contains only quadratic or arbitrary nonlinearities. It turns out that the proposed heuristics perform comparably independent of the type of nonlinearities but that convex problems are solved faster than nonconvex MINLPs of equal size.

In total, the MPEC-based heuristics perform comparably for different classes of MINLPs. The idea can thus be used as a general-purpose heuristic. It is typically fast if the resulting MPEC regularization can be solved fast, which is usually the case for smaller and/or convex models.

4.2 Comparison with other MINLP heuristics

In this section, we compare the performance of our primal heuristic with different state-of-the-art heuristics for MINLPs. Since feasibility pumps belong to the most powerful heuristics we compare the most successful parameterizations of our heuristic (penalty-based MPEC reformulation solved with Ipopt and SNOPT) against the six feasibility pump variants for MINLPs presented in [11]. Empirical distribution functions for the primal gaps and running times are given in Fig. 7. Here, we used the results as published in [11] and remark that, as a consequence, different computational setups are used. Although this has a clear impact on running times, the quality of the obtained feasible points depends less on the different setups. This also applies to the remaining comparisons in this section. Moreover, some running times reported in [11] are 0 s, which is not the case for the data of our methods. This also slightly introduces inexactness to the comparison. However, the qualitative behavior on the entire test set should not be influenced by these aspects. For what follows, we used the intersection of our test set with the one used in the given paper. It can be seen that the MPEC-based heuristic dominates the feasibility pump implementations both in terms of solution quality and running times.

Table 5 Average running times depending on (i) the chosen MPEC regularization, (ii) the NLP solver, (iii) the number of binary variables, and (iv) the overall number of variables

	Number of binary variables														
	[0, 25]			(25, 50]			(50, 100]			(100, 200]			(200, ∞)		
[0, 75]	0.08	0.66	15.00	0.10	25.06	1.22	**0.13**	1.68	0.20	–	–	–	–	–	–
	0.07	0.81	11.66	**0.08**	8.00	5.02	**0.13**	1.81	0.20	–	–	–	–	–	–
	0.08	16.06	2.53	**0.08**	3.65	23.58	0.16	1.38	0.17	–	–	–	–	–	–
(75, 200]	0.23	83.23	**0.17**	1.28	13.68	62.33	0.33	29.49	0.70	**0.38**	176.61	0.54	–	–	–
	0.22	76.17	0.27	0.96	4.86	47.16	0.28	84.47	20.80	0.37	225.55	2.34	–	–	–
	0.33	22.52	13.13	**0.68**	34.27	29.47	**0.24**	43.39	41.70	0.39	65.36	0.63	–	–	–
(200, 500]	0.44	1.31	0.35	0.88	17.92	55.89	**0.66**	32.94	64.44	**1.44**	63.61	53.29	25.40	174.94	5.19
	0.39	1.98	**0.37**	**0.83**	54.18	23.64	0.68	32.66	50.35	1.45	57.62	70.49	25.62	193.56	5.72
	1.21	3.63	90.43	1.02	192.84	8.13	0.78	47.85	76.36	1.55	92.35	10.84	24.77	179.75	**3.52**
(500, 1.5×10^3]	6.64	7.62	168.73	**1.99**	108.48	121.25	⋆	393.12	84.90	4.95	162.59	6.18	⋆	121.81	26.59
	⋆	51.91	95.10	2.16	101.24	145.99	⋆	396.84	**61.36**	4.69	180.06	16.71	⋆	146.33	62.96
	6.44	58.31	112.70	⋆	193.02	62.80	⋆	373.31	172.12	**4.24**	242.29	44.07	34.08	195.43	**18.41**
(1.5×10^3, ∞)	44.57	73.09	655.90	**96.31**	473.31	281.96	⋆	645.51	324.10	⋆	702.71	130.53	⋆	⋆	398.84
	44.35	83.24	512.44	⋆	470.18	351.39	⋆	762.81	**192.13**	⋆	667.40	183.35	⋆	⋆	364.08
	⋆	200.36	568.89	138.87	753.98	461.61	⋆	824.67	194.25	⋆	633.05	**94.48**	⋆	1541.29	**276.57**

The inner 3×3 table blocks correspond (from top to down) to the Scholtes, the Fischer–Burmeister, and the penalty reformulation and (from left to right) to CONOPT4, Ipopt, and SNOPT

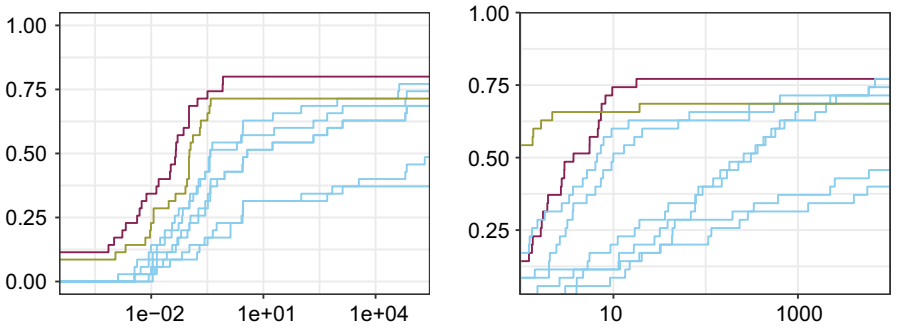

Fig. 7 Empirical distribution functions of primal gaps (left) and absolute running times (in s; right). Comparison of penalty-based MPEC reformulation solved with Ipopt (magenta) and SNOPT (green) and all 6 variants of the feasibility pumps (blue) presented in [11] (color figure online)

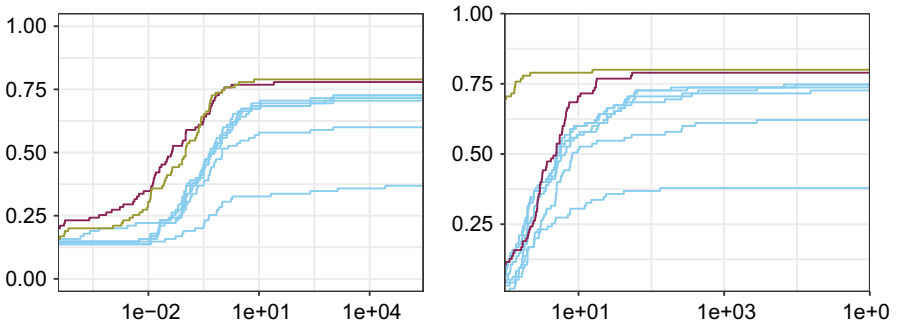

Fig. 8 Empirical distribution functions of primal gaps (left) and absolute running times (in s; right). Comparison of penalty-based MPEC reformulation solved with Ipopt (magenta) and SNOPT (green) and the feasibility pumps for MINLP presented in [4] (blue) (color figure online)

We further compare the same two parameterizations of our method with the results of the feasibility pump implementations discussed in [4] and the Undercover heuristic presented in [5]. In Fig. 8 we plot the empirical distribution functions for primal gaps (left) and running times (right) for the comparison with the feasibility pumps of [4]. Our methods outperform the feasibility pump results both in terms of solution quality and running times. The latter is particularly significant for the penalty-based MPEC reformulation solved with SNOPT. Finally, Fig. 9 displays the empirical distribution functions for the primal gaps of our method and the Undercover heuristic. Both penalty-based MPEC strategies yield better results than Undercover. Here, a comparison of running times is not possible because the required solution times are not reported in [4]. We again see that the MPEC-based heuristics dominate the heuristic from the literature. In summary, our analysis shows that the presented primal heuristics are very successful in terms of the total number of solved instances and the required running times. Moreover, the MPEC-based heuristics perform comparably for different MINLPs and are successful for problems with large numbers of (binary) variables, for convex and nonconvex problems, and for all appearing types of nonlinearities. Finally, the method can, at least, compete with other state-of-the-art heuristics for MINLPs. This also led

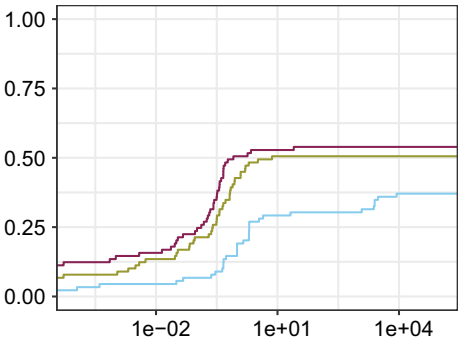

Fig. 9 Empirical distribution functions of primal gaps. Comparison of penalty-based MPEC reformulation solved with Ipopt (magenta) and SNOPT (green) and the Undercover heuristic (blue) (color figure online)

to the integration of the method in the general-purpose global MINLP solver SCIP 5.0; see [20], where it is activated by default with conservative working limits.

5 Conclusion

In this paper we described how feasible points for binary MINLPs can be computed using MPEC-type reformulations. We have shown that these methods are fast enough to be incorporated in global MINLP solvers. On 662 mixed-integer nonlinear optimization problems from the MINLPLib2, the proposed techniques are capable of finding feasible solutions of more than 70% of the instances and that 50% can be solved within 1 s (or more than 60% in less than 10 s).

Even though a simple MPEC-reformulation yields NLPs for which standard constraint qualifications do not hold, we have seen that appropriate regularization schemes and careful backup strategies can resolve these difficulties. In our opinion, these results suggest that a good next question is whether MPEC-reformulations should replace the simple NLP-relaxation heuristic that is commonly used by global MINLP solvers.

Despite the good results presented in this paper, some more improvements can be considered. One of the main omissions of our method is that we do not do any preprocessing of our problem. In combination with heuristically fixing variables that are integral after a few iterations of our procedure this could be used in a diving-like method. Finally, further techniques for improving the solution quality in additional polishing steps might be useful.

Acknowledgements This research has been 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. Both authors acknowledge funding through the DFG Transregio TRR 154, subprojects A05, B07, and B08. Last but not least, we want to express our sincere gratefulness to Stefan Vigerske from GAMS. Without his patient help, the implementations underlying this paper would not have been possible. Thanks a lot, Stefan.

References

1. Achterberg, T.: SCIP: solving constraint integer programs. Math. Program. Comput. **1**(1), 1–41 (2009). https://doi.org/10.1007/s12532-008-0001-1

2. Baumrucker, B.T., Renfro, J.G., Biegler, L.T.: MPEC problem formulations and solution strategies with chemical engineering applications. Comput. Chem. Eng. **32**(12), 2903–2913 (2008). https://doi.org/10.1016/j.compchemeng.2008.02.010
3. Benoist, T., Estellon, B., Gardi, F., Megel, R., Nouioua, K.: Localsolver 1.x: a black-box local-search solver for 0–1 programming. 4OR **9**(3), 299 (2011). https://doi.org/10.1007/s10288-011-0165-9
4. Berthold, T.: Heuristic algorithms in global MINLP solvers. Ph.D. thesis, Technische Universität Berlin (2014)
5. Berthold, T., Gleixner, A.M.: Undercover: a primal MINLP heuristic exploring a largest sub-MIP. Math. Program. **144**(1), 315–346 (2014). https://doi.org/10.1007/s10107-013-0635-2
6. Berthold, T., Heinz, S., Pfetsch, M.E., Vigerske, S.: Large neighborhood search beyond MIP. In: Proceedings of the 9th Metaheuristics International Conference (MIC 2011), pp. 51–60 (2011)
7. Berthold, T., Heinz, S., Vigerske, S.: Extending a CIP framework to solve MIQCPs. In: Lee, J., Leyffer, S. (eds.) Mixed Integer Nonlinear Programming, pp. 427–444. Springer, New York (2012). https://doi.org/10.1007/978-1-4614-1927-3_15
8. Bonami, P., Cornuéjols, G., Lodi, A., Margot, F.: A feasibility pump for mixed integer nonlinear programs. Math. Program. **119**(2), 331–352 (2009). https://doi.org/10.1007/s10107-008-0212-2
9. Bonami, P., Gonçalves, J.P.M.: Heuristics for convex mixed integer nonlinear programs. Comput. Optim. Appl. **51**(2), 729–747 (2012). https://doi.org/10.1007/s10589-010-9350-6
10. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: Experiments with a feasibility pump approach for nonconvex MINLPs. In: Festa, P. (ed.) Experimental Algorithms. Lecture Notes in Computer Science, vol. 6049, pp. 350–360. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-13193-6_30
11. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: A storm of feasibility pumps for nonconvex MINLP. Math. Program. **136**(2), 375–402 (2012). https://doi.org/10.1007/s10107-012-0608-x
12. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002). https://doi.org/10.1007/s101070100263
13. Drud, A.S.: CONOPT—a large-scale GRG code. INFORMS J. Comput. **6**(2), 207–216 (1994). https://doi.org/10.1287/ijoc.6.2.207
14. Drud, A.S.: CONOPT: a system for large scale nonlinear optimization, tutorial for CONOPT subroutine library. Technical Report, ARKI Consulting and Development A/S, Bagsvaerd, Denmark (1995)
15. Drud, A.S.: CONOPT: a system for large scale nonlinear optimization, reference manual for CONOPT subroutine library. Technical Report, ARKI Consulting and Development A/S, Bagsvaerd, Denmark (1996)
16. Fischer, A.: A special Newton-type optimization method. Optimization **24**(3–4), 269–284 (1992). https://doi.org/10.1080/02331939208843795
17. Fischetti, M., Lodi, A.: Local branching. Math. Program. **98**(1–3), 23–47 (2003). https://doi.org/10.1007/s10107-003-0395-5
18. GAMS Development Corporation: General Algebraic Modeling System (GAMS) Release 24.5.4. Washington, DC, USA (2015). http://www.gams.com. Accessed 10 Aug 2018
19. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**(1), 99–131 (2005). https://doi.org/10.1137/S0036144504446096
20. Gleixner, A., Eifler, L., Gally, T., Gamrath, G., Gemander, P., Gottwald, R.L., Hendel, G., Hojny, C., Koch, T., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schlösser, F., Serrano, F., Shinano, Y., Viernickel, J.M., Vigerske, S., Weninger, D., Witt, J.T., Witzig, J.: The SCIP Optimization Suite 5.0. Technical Report 17-61, ZIB, Takustr.7, 14195 Berlin (2017)
21. Hoheisel, T., Kanzow, C., Schwartz, A.: Theoretical and numerical comparison of relaxation methods for mathematical programs with complementarity constraints. Math. Program. **137**(1), 257–288 (2013). https://doi.org/10.1007/s10107-011-0488-5
22. Hu, X.M., Ralph, D.: Convergence of a penalty method for mathematical programming with complementarity constraints. J. Optim. Theory Appl. **123**(2), 365–390 (2004). https://doi.org/10.1007/s10957-004-5154-0
23. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: MIPLIB 2010. Math. Program. Comput. **3**(2), 103–163 (2011). https://doi.org/10.1007/s12532-011-0025-9
24. Kraemer, K., Kossack, S., Marquardt, W.: An efficient solution method for the MINLP optimization of chemical processes. Comput. Aided Chem. Eng. **24**, 105 (2007). https://doi.org/10.1016/S1570-7946(07)80041-1

25. Kraemer, K., Marquardt, W.: Continuous reformulation of MINLP problems. In: Diehl, M., Glineur, F., Jarlebring, E., Michiels, W. (eds.) Recent Advances in Optimization and Its Applications in Engineering: The 14th Belgian-French-German Conference on Optimization, pp. 83–92. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-12598-0_8
26. Liberti, L., Mladenović, N., Nannicini, G.: A recipe for finding good solutions to MINLPs. Math. Program. Comput. **3**(4), 349–390 (2011). https://doi.org/10.1007/s12532-011-0031-y
27. Luo, Z.Q., Pang, J.S., Ralph, D.: Mathematical Programs with Equilibrium Constraints. Cambridge University Press, Cambridge (1996)
28. Maher, S.J., Fischer, T., Gally, T., Gamrath, G., Gleixner, A., Gottwald, R.L., Hendel, G., Koch, T., Lübbecke, M.E., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Weninger, D., Witt, J.T., Witzig, J.: The SCIP optimization suite 4.0. Technical Report 17-12, ZIB, Takustr.7, 14195 Berlin (2017)
29. Nannicini, G., Belotti, P.: Rounding-based heuristics for nonconvex MINLPs. Math. Program. Comput. **4**(1), 1–31 (2012). https://doi.org/10.1007/s12532-011-0032-x
30. Nannicini, G., Belotti, P., Liberti, L.: A local branching heuristic for MINLPs (2008). arXiv preprint arXiv:0812.2188
31. Rose, D., Schmidt, M., Steinbach, M.C., Willert, B.M.: Computational optimization of gas compressor stations: MINLP models versus continuous reformulations. Math. Methods Oper. Res. **83**(3), 409–444 (2016). https://doi.org/10.1007/s00186-016-0533-5
32. Sahinidis, N.V.: BARON 14.3.1: Global Optimization of Mixed-Integer Nonlinear Programs, User's Manual (2014)
33. Schmidt, M., Steinbach, M.C., Willert, B.M.: A primal heuristic for nonsmooth mixed integer nonlinear optimization. In: Jünger, M., Reinelt, G. (eds.) Facets of Combinatorial Optimization, pp. 295–320. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-38189-8_13
34. Schmidt, M., Steinbach, M.C., Willert, B.M.: An MPEC based heuristic. In: Koch, T., Hiller, B., Pfetsch, M.E., Schewe, L. (eds.) Evaluating Gas Network Capacities, SIAM-MOS series on Optimization, Chapter 9, pp. 163–180. SIAM (2015). https://doi.org/10.1137/1.9781611973693.ch9
35. Scholtes, S.: Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. SIAM J. Optim. **11**(4), 918–936 (2001). https://doi.org/10.1137/S1052623499361233
36. Stein, O., Oldenburg, J., Marquardt, W.: Continuous reformulations of discrete-continuous optimization problems. Comput. Chem. Eng. **28**(10), 1951–1966 (2004). https://doi.org/10.1016/j.compchemeng.2004.03.011. (Special Issue for Professor Arthur W. Westerberg)
37. Sun, D., Qi, L.: On NCP-functions. Comput. Optim. Appl. **13**(1), 201–220 (1999). https://doi.org/10.1023/A:1008669226453
38. Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications. Kluwer Academic Publishers, Dordrecht (2002)
39. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: a theoretical and computational study. Math. Program. **99**(3), 563–591 (2004). https://doi.org/10.1007/s10107-003-0467-6
40. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**(2), 225–249 (2005). https://doi.org/10.1007/s10107-005-0581-8
41. Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., Martí, R.: Scatter search and local NLP solvers: a multistart framework for global optimization. INFORMS J. Comput. **19**(3), 328–340 (2007). https://doi.org/10.1287/ijoc.1060.0175
42. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006). https://doi.org/10.1007/s10107-004-0559-y
43. Wächter, A., Biegler, L.T.: Line search filter methods for nonlinear programming: motivation and global convergence. SIAM J. Optim. **16**(1), 1–31 (2005). https://doi.org/10.1137/S1052623403426556
44. Wu, B., Ghanem, B.: l_p-box ADMM: a versatile framework for integer programming. Technical Report (2016). http://arxiv.org/abs/1604.07666. Accessed 10 Aug 2018
45. Ye, J.J., Zhu, D.L.: Optimality conditions for bilevel programming problems. Optimization **33**(1), 9–27 (1995). https://doi.org/10.1080/02331939508844060