Computing Feasible Points for MINLPs with MPECs

Lars Schewe¹, Martin Schmidt²

Abstract. Nonconvex mixed-integer 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-integer nonlinear problem. Since we only rely on local nonlinear programming solvers the resulting method is fast and we further improve its robustness by additional algorithmic techniques. We show the strength of our method by an extensive computational study on 662 MINPLib2 instances, where our methods are able to produce feasible solutions for 60% of all instances in at most 10 s.

1. Introduction

In this paper we consider nonconvex mixed-integer 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.

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. 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 Bonami and Gonçalves [8], 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 Bonami et al. [7] or, for nonconvex MINLPs, in D’Ambrosio et al. [10]. Other generalizations are given in Nannicini, Belotti, and Liberti [26], where local branching is carried over from MIP (see Fischetti and Lodi [16]) to nonconvex MINLP or in Berthold, Heinz, and Vigerske.

¹

Date: December 21, 2016.

2010 Mathematics Subject Classification. 90-08, 90C11, 90C33, 90C59.

Key words and phrases. Mixed-Integer Nonlinear Optimization, MINLP, MPEC, Complementarity Constraints, Primal Heuristic.
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. [5] extended classical local neighborhood search (LNS) heuristics for MIPs in order to be applied to nonlinear problems. Similar work is presented in Liberti, Mladenović, and Nannicini [23] where the authors discuss a general-purpose heuristic based on variable neighborhood search, local branching, a local nonlinear programming solver, and branch-and-bound. 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 Ugray et al. [33] or LocalSolver described in Benoist et al. [2].

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 Berthold and Gleixner [4], which is a MIP-based MINLP heuristic that does not carry over an idea of an well-known MIP heuristic or the rounding procedures given in Nannicini and Belotti [25] 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 Berthold [3]. 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 for 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 robust as MIP solvers, which is why we also apply additional algorithmic techniques to robustify 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, Stein, Oldenburg, and Marquardt [31] study 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 Kraener, Kossack, and Marquardt [21] and Kraemer and Marquardt study the special reformulation using a regularized version of the Fischer–Burmeister function and apply the resulting regularization scheme in [22] to parts of the general-purpose MINLPLib library and to problems from the synthesis of distillation systems.

Recently, MPEC-based reformulations of nonconvex MINLPs have also been considered in the field of gas transport optimization; see Schmidt, Steinbach, and Willert [28, 29] and Rose et al. [27]. For general information about MPECs we
refer to the textbook [23] of Luo, Pang, and Ralph and to the survey paper [1] by Baumrucker, Renfro, and Biegler. 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 Section 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 robustify the solution process, and states the overall primal heuristic. This method is then tested extensively in Section 4. We test different NLP solvers, MPEC regularizations, and the impact of the enhanced techniques. The paper closes with some concluding remarks and some comments on possible future work in Section 5.

2. Problem Statement, MPEC Reformulations, and Regularization

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

\[
\begin{align*}
\min_{x,z} & \quad f(x, z) \\
\text{s.t.} & \quad g(x, z) \geq 0, \\
& \quad x \in \mathbb{R}^n, \\
& \quad z \in \{0,1\}^m,
\end{align*}
\]

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 \( \Omega \), i.e.,

\[
\Omega := \{(x, z) \mid g(x, z) \geq 0, \ x \in \mathbb{R}^n, \ z \in \{0,1\}^m \} \subseteq \mathbb{R}^n \times \{0,1\}^m.
\]

In what follows we assume that \( \Omega \) 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

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

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

\[
\begin{align*}
\min_{x,z} & \quad f(x, z) \\
\text{s.t.} & \quad g(x, z) \geq 0, \\
& \quad x \in \mathbb{R}^n, \\
& \quad 0 \leq z_i \perp 1 - z_i \geq 0 \quad \text{for all } i = 1, \ldots, m,
\end{align*}
\]
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 $\Omega C$ of (3) coincides with the feasible set $\Omega$ of the original MINLP. Constraint (3d) is a so-called complementarity constraint, i.e., Model (3) is an MPCC, or an, in the more general notion, 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 [24] by Luo, Pang, and Ralph 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 Ye and Zhu [35]. 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.

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, Kanzow, and Schwartz [18]. These schemes solve regularized problems $R(\tau)$ 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 $\tau_0$ they solve the problem $R(\tau_0)$, compute a new parameter $\tau_1 < \tau_0$, solve $R(\tau_1)$ (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 $R(\tau_k)$ 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 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 [18]. 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 Scholtes [30]. Applied to our situation the regularization discussed by Scholtes

---

**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 $\tau_{\text{min}} > 0$, and a constant $\sigma \in (0, 1)$.
2. Set $k = 0$.
3. while $\tau_k > \tau_{\text{min}}$ do
   4. Solve the regularized problem $R(\tau_k)$ 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 $k \leftarrow k + 1$.
4. end while
5. return last iterate $(x^k, z^k)$. 

---
reads
\[
\begin{align*}
\min_{x,z} & \quad f(x,z) \\
\text{s.t.} & \quad g(x,z) \geq 0, \\
& \quad x \in \mathbb{R}^n, \ z \in \mathbb{R}^m, \\
& \quad z_i(1 - z_i) \leq \tau, \ z_i \in [0,1] \quad \text{for all } i = 1, \ldots, m.
\end{align*}
\]

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 $\Omega_C$ 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 Hoheisel, Kanzow, and Schwartz [18]: 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., Baumrucker, Renfro, and Biegler [1].

Another frequently used strategy for replacing the complementarity constraints (3d) is using so-called NCP functions $\phi : \mathbb{R}^2 \to \mathbb{R}$ that satisfy $\phi(a,b) = 0$ iff $a,b \geq 0$ and $ab = 0$; see, e.g., Sun and Qi [32]. 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 [15]. 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
\[
\begin{align*}
\min_{x,z} & \quad f(x,z) \\
\text{s.t.} & \quad g(x,z) \geq 0, \\
& \quad x \in \mathbb{R}^n, \ z \in [0,1]^m, \\
& \quad \sqrt{z_i^2 + (1 - z_i)^2 + \tau} \geq 1 \quad \text{for all } i = 1, \ldots, m.
\end{align*}
\]

This regularization has also been applied for reformulating discrete-continuous optimization problems in Kraemer and Marquardt [22].

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

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

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

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

This is somehow the strongest regularization of the discussed ones since it obviously holds \(\Omega_P \subset \Omega_C\) for the feasible set \(\Omega_P\) of (6). This approach has been proposed in Hu and Ralph [19] 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 robust 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), and (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, a practical implementation that only relies on the techniques discussed in Section 2 is not as successful as desired. Some reasons for this can be handled within the solution procedure. The corresponding techniques are discussed in this section that ends with a description of the entire algorithm in Section 3.3.

3.1. Re-Initialization and Variable Fixing. An iterative tightening of the regularization parameter within the constraint-based regularization schemes somewhat yields disjoint feasible regions for every binary variable. 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 Kraemer and Marquardt [22] for the Fischer–Burmeister regularization of MINLPs. We follow their ideas resolving the problem in this section: Whenever an NLP \(R(\tau_k)\) 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 \(R(\tau_k)\) again but with \(z_i, i \in I\), initialized to \(\hat{z}_i\) as follows:

\[
\hat{z}_i = \begin{cases} 
0, & \text{if } z_i^k \geq 0.5, \\
1, & \text{otherwise}
\end{cases}
\]
If \( R(\tau_k) \) 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 \( z_j \) by imposing the temporary fixations
\[
    z_j = \begin{cases} 
        0, & \text{if } z_j^k \geq 0.5, \\
        1, & \text{otherwise}.
    \end{cases}
\]

If this variable fixing strategy or the above mentioned re-initialization strategy yields a feasible solution of \( R(\tau_k) \) 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 \([5]\) needs some additional explanations. Since the penalty regularized problems typically are never infeasible 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 \( R(\tau_k) \) 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 \( R(\tau_k) \) with the goal to obtain an approximate solution of \( R(0) \). It is known that such kind of methods can fail in practice if the parameter \( \tau \) 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 \( R(\tau_k) \) a reason for this failure may also be a too aggressive update from \( \tau_k - 1 \) to \( \tau_k \). Thus, we enter a backtracking phase in which we backtrack the regularization parameter via
\[
    \tau_{k,\ell} = \frac{\tau_k}{\sigma^{1/\ell^2}}, \quad \tau_k^0 = \tau_k, \quad \ell = 1, \ldots, \ell_{\text{max}},
\]
and solve the corresponding problem \( R(\tau_{k,\ell}) \) until we obtain a feasible solution or the (user-specified) backtracking iteration limit \( \ell_{\text{max}} \) is reached. In case that we find a feasible solution of \( R(\tau_{k,\ell}) \) for some \( \ell \) we resume the overall algorithm for \( \tau = \tau_k \) and solve \( R(\tau_k) \) initialized with the solution of \( R(\tau_{k,\ell}) \).

### 3.3. The Final Algorithm

We now collect all enhanced solution techniques and combine them with the generic Algorithm 1 of Section 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 \( R(\tau_k) \) 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 \( R(\tau_k) \). In this state, re-initialization candidates are determined and re-initialized as described in Section 3.1

**Fix:** The NLP solver failed to solve the re-initialized problem of state Re-Init. The variable fixing strategy of Section 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 Section 3.2 until a feasible solution has been found or an backtracking iteration limit is reached.

The overall solution procedure is illustrated in Figure 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 \( R(\tau) \) 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.

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 [12, 14], the SQP code SNOPT [17], and the interior-point solver Ipopt [34].
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 Section 3, i.e., re-initialization, variable fixing, and regularization parameter backtracking.

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 [9]. The code has been compiled with gcc 4.8.4. All computations have been executed on a 48-core machine.
Table 1. Number of feasible solutions found for different combinations of constraint-based MPEC reformulations and NLP solvers in different states of the algorithm

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

with AMD Opteron™ 6176 SE processors running at 2300 MHz with a maximum of 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 variables, 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 Dolan and Moré [11] to compare running times and solution quality. Following [20], the latter is measured by the primal gap that is defined by

\[ \text{gap} = \frac{b_p - b_d}{\inf\{|z| : z \in [b_d, b_p]\}}, \]  

where \( b_p \) is the primal and \( b_d \) is best known solution according to the MINLPLib2 website, respectively. Obviously, (7) is only valid for minimization problems but can be easily adapted for maximization problems. Two special cases merit particular mention: It holds that \( \text{gap} = +\infty \) whenever \( b_d < 0 \leq b_p \), and we set \( \text{gap} = 0 \) if \( b_d = b_p = 0 \).

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 starting point. In these cases we used the provided point. The regularization parameter \( \tau \) is always initialized to 100. The time limit is set to 900 s and the integrality tolerance is chosen to be \( 10^{-5} \).

We now turn to the presentation and discussion of the numerical results. In Table 1 we see the number of instances that are solved to feasibility for all tested combinations of constraint-based regularization 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 Section 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 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 than 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. 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 Figure 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 $\tau_0$ and assuming that the algorithm stays in the Basic state, the update rule for $\tau$ leads to a regularization parameter $\tau_5$ such that (both for the reformulation scheme of Scholtes and the Fischer–Burmeister scheme) the 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 is a significant amount of instances for which the NLP solution of the initial MPEC regularization 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 fewest 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 such 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 Figure 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 $k \geq 8$.

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

---

1For CONOPT4, the failed instances are crudeoil_lee1_08, crudeoil_lee2_05, crudeoil_lee2_06, crudeoil_lee3_05, crudeoil_lee3_06, crudeoil_lee3_10, gasprod_sarawak81, nuclear25a, sepasequ_convent, 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_lee6_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 falls on faclay75 for the Fischer–Burmeister reformulation and on faclay33 and for the penalty-based reformulation.
Table 2. Number of feasible solutions found for the penalty-based reformulation and different NLP solvers in different states of the algorithm

<table>
<thead>
<tr>
<th>Regularization</th>
<th>Solver</th>
<th>Basic</th>
<th>Re-Init</th>
<th>Fix</th>
<th>Backtrack</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Penalty</td>
<td>CONOPT4</td>
<td>146</td>
<td>316</td>
<td>5</td>
<td>7</td>
<td>474</td>
</tr>
<tr>
<td>Penalty</td>
<td>Ipopt</td>
<td>129</td>
<td>335</td>
<td>3</td>
<td>—</td>
<td>467</td>
</tr>
<tr>
<td>Penalty</td>
<td>SNOPT</td>
<td>133</td>
<td>342</td>
<td>1</td>
<td>—</td>
<td>476</td>
</tr>
</tbody>
</table>

Figure 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).

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 Section 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
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 $10^{-5}$)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>CONOPT4</th>
<th>Ipopt</th>
<th>SNOPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scholtes</td>
<td>47</td>
<td>61</td>
<td>37</td>
</tr>
<tr>
<td>Fischer-Burmeister</td>
<td>47</td>
<td>64</td>
<td>42</td>
</tr>
<tr>
<td>Penalty</td>
<td>64</td>
<td>81</td>
<td>52</td>
</tr>
</tbody>
</table>

schemes. Moreover, one clearly sees that the additional states, i.e., the Fix and the Backtrack state, does not give much more feasible instances. The bar plots of Figure 3 approve these observations. After finding always 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 Figure 2 to Iteration 20. 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 Figure 3. 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 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 robustness strongly depends on the formulation: While the constraint-based formulations solved with SNOPT lead to the most worse 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 Figure 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 Figure 6 we see the performance profiles of all tested methods using the primal gap plot as defined in Equation (7) used as performance measure. 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 the feasible points with the best objective values: For 25% of the 662 instances Ipopt applied to the penalty regularization yields a feasible point that is not 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%.

5. Conclusion

In this paper we described how feasible points for 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, one may also consider the embedding of our method in a global solver in order to measure its impact as a primal heuristic. In this context, further techniques from 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 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

Figure 6. Log-scaled performance profiles of the primal gap for all 9 combinations of MPEC reformulation schemes and NLP solvers


REFERENCES


---

1Lars Schewe, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Discrete Optimization, Cauerstr. 11, 91058 Erlangen, Germany, 2Martin Schmidt, (a) Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Discrete Optimization, Cauerstr. 11, 91058 Erlangen, Germany; (b) Energie Campus Nürnberg, Fürther Str. 250, 90429 Nürnberg, Germany

E-mail address: 1lars.scheewe@fau.de

E-mail address: 2mar.schmidt@fau.de