Improving the performance of DICOPT in convex MINLP problems using a feasibility pump

David E. Bernal[a], Stefan Vigerske[b], Francisco Trespalacios[c], and Ignacio E. Grossmann[a*]

[a]Department of Chemical Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA;
[b]GAMS Software GmbH, c/o Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany;
[c]ExxonMobil Research and Engineering, 1545 Route 22 East, Annandale NJ 08801, USA

August 12, 2019

Abstract

The solver DICOPT is based on the outer-approximation algorithm used for solving mixed-integer nonlinear programming (MINLP) problems. This algorithm is very effective for solving some types of convex MINLPs. However, it has been observed that DICOPT has difficulties solving instances in which some of the nonlinear constraints are so restrictive that nonlinear subproblems generated by the algorithm are infeasible. This problem is addressed in this paper with a feasibility pump algorithm, which modifies the objective function in order to efficiently find feasible solutions. It has been implemented as a preprocessing algorithm, which is used to initialize both the incumbent and the mixed-integer linear relaxation of the outer-approximation algorithm. Computational comparisons with previous versions of DICOPT on a set of convex MINLPs demonstrate the effectiveness of the proposed algorithm in terms of solution quality and solution time.

Keywords: feasibility pump; mixed-integer nonlinear programming; primal heuristics

2010 Mathematics Subject Classification: 90C11, 90C25, 90C30.

1 Introduction

The capabilities of algorithms designed to solve mathematical programming problems are continuously improving. This allows solving increasingly larger and more complex problems. Efficient solutions of mixed-integer linear programs (MIP) and nonlinear programs (NLP) enable the solution of mixed-integer nonlinear programs (MINLP). These problems are of great interest in chemical engineering and many other areas as they combine integer variables (like discrete choices in superstructures or networks) with nonlinear constraints (for example posynomial equations in resource allocation for scheduling and convex reformulations of horizon time constraints in the design of multiproduct batch processes) [5, 15, 16, 19, 21]. The general form of an MINLP is,

$$\begin{aligned} \min_{x,y} \quad & f(x,y), \\ \text{s.t.} \quad & g(x,y) \leq 0, \\ & x \in X,\ y \in Y \cap \mathbb{Z}^{n_y}, \end{aligned} \tag{MINLP}$$

where $X \subseteq \mathbb{R}^{n_x}$, $Y \subseteq \mathbb{R}^{n_y}$, $f : X \times Y \to \mathbb{R}$ is the objective function and at least one of the constraints $g : X \times Y \to \mathbb{R}^m$ or the objective function itself is nonlinear. MINLP models are

*Corresponding author. Email: grossmann@cmu.edu

generally nonconvex due to the discrete nature of y and possible nonconvexity of f and g. Models in which X and Y are convex and f and g_i, $i = 1, \ldots, m$, are convex on $X \times Y$, are denoted as convex MINLP problems. In the following, we will assume that X and Y describe possible variable bounds, that is, $X = [x^L, x^U]$ and $Y = [y^L, y^U]$ for $x^L, x^U \in (\mathbb{R} \cup \{\pm\infty\})^{n_x}$ and $y^L, y^U \in (\mathbb{Z} \cup \{\pm\infty\})^{n_y}$.

DICOPT (Discrete Continuous Optimizer) is an MINLP solver that has been developed in 1988. It combines the outer-approximation method [11] with equality relaxation and augmented penalty [23]. The algorithm decomposes the MINLP into an NLP subproblem defined by fixing the discrete variables in the MINLP and a MIP approximation defined by linearizations of the nonlinear functions in the MINLP. The MIP and the NLP are solved alternately, whereby the MIP approximation provides values for fixing the discrete variables in the NLP, and the convex NLP subproblem, if feasible, provides feasible solutions to the MINLP and cutting planes to improve the MIP approximation. If the MINLP is convex, then this MIP approximation is a relaxation of the MINLP, thus providing a lower bound to its optimal value, and the NLP subproblems can be solved to global optimality yielding an upper bound. By adding additional inequalities to the MIP approximation, one can further ensure that any fixed values for the discrete variables are evaluated by an NLP at most once. Therefore, for a convex MINLP, a possible stopping criterion is that the bound defined by the last MIP approximation is within a tolerance of the objective value of the best solution found.

For some problems, DICOPT has difficulty in finding a feasible solution. The main reason for this is that by default, and to address nonconvex problems, DICOPT does not include linearizations of nonlinearities from infeasible NLPs into the MIP. Instead, it only excludes the infeasible fixed integer variables in the MIP and resolves it. Furthermore, even if linearizations are included for infeasible NLPs, which are valid for convex MINLPs, DICOPT still has difficulties in finding a feasible solution for some problems, which results in slow progress compared to the case where feasible MINLP solutions are found early in the search.

In order to quickly find initial feasible solutions for convex MINLPs, an implementation of a feasibility pump [6, 12] has been incorporated into DICOPT as described in this paper. The feasibility pump is similar to the outer-approximation algorithm, but its focus is on finding feasible solutions. As with outer-approximation, the main idea of the feasibility pump is to decompose the original MINLP problem into a MIP and an NLP. The MIP problem yields solutions that satisfy integrality requirements ($y \in \mathbb{Z}^{n_y}$) but may violate nonlinear constraints, while the solutions of the NLP problems satisfy the constraints $g(x, y) \leq 0$ but may violate integrality requirements. Contrary to outer-approximation, both MIP and NLP are defined over relaxations of the feasible area of the original MINLP. By alternately projecting onto the MIP and NLP relaxations, a solution is obtained that is feasible for both relaxations, and thus for the MINLP itself, if certain constraint qualifications are satisfied [6]. The feasibility pump can also be used as a standalone solver for convex MINLPs by including a bound (cutoff-value) to the objective function, which is set to the objective value of the best-known solution, reduced by a desired δ-improvement. Applying this modified feasibility pump repeatedly until the MIP becomes infeasible, yields a δ-optimal solution of a convex MINLP [6]. The drawback of this algorithm is that it may require many iterations, since only a δ-improvement of the objective function is enforced at each iteration.

In this work, a feasibility pump is added to DICOPT and is used as an initialization of the outer-approximation method. In the feasibility pump, improvements in the objective function are enforced at each iteration. After the method finishes, the cuts that define the MIP relaxation of the feasibility pump and the best-found solution are passed on to the outer-approximation method to find better solutions, if any, and prove optimality. The described extension of DICOPT has been available in GAMS[1] since version 24.5. We present computational results of the new method on a set of convex MINLP problems, and show that it outperforms the previous version of DICOPT.

This paper is organized as follows. Section 2 provides an overview of the outer-approximation and the feasibility pump algorithms for convex MINLP problems. Section 3 describes the algorithm proposed in this work, which uses the feasibility pump as initialization for the outer-approximation algorithm. An illustrative example and computational results are presented in Section 4.

[1] `http://www.gams.com/latest`

2 Background

In the following, we summarize the outer-approximation algorithm [11], the feasibility pump algorithm [6], and a hybrid of both algorithms. These algorithms are intended to solve problems of the form (MINLP). The following assumptions are made [11, 13]:

(A1) The integer variables are bounded, that is, Y is a bounded set.

(A2) The constraint functions $g(x, y)$ and the objective function $f(x, y)$ are continuously differentiable and convex on $X \times Y$.

(A3) The continuous relaxation of the (MINLP) obtained by removing the integrality requirement $y \in \mathbb{Z}^{n_y}$ is bounded.

2.1 Outer-approximation algorithm

The outer-approximation algorithm was proposed by Duran and Grossmann in 1986 [11]. In the original version of the algorithm, the starting point was given by some fixed values for the binary variables y. Viswanathan and Grossmann [23] proposed to solve the continuous relaxation of the MINLP in the first iteration, which is obtained by relaxing the integrality requirement on y,

$$\begin{aligned} \min_{x,y} \quad & f(x, y), \\ \text{s.t.} \quad & g(x, y) \leq 0, \\ & x \in X,\ y \in Y. \end{aligned} \tag{rMINLP}$$

If (rMINLP) is infeasible, then (MINLP) is also infeasible. Otherwise, let $(\bar{x}^0, \bar{y}^0)$ be a solution to (rMINLP). If $\bar{y}^0$ is integral, $(\bar{x}^0, \bar{y}^0)$ is an optimal solution to (MINLP) and the algorithm stops.

If $\bar{y}^0$ is not integral, a MIP relaxation of (MINLP) is constructed by linearizing the nonlinear functions in $g(x, y)$ and $f(x, y)$ by first-order Taylor series approximations at $(\bar{x}^0, \bar{y}^0)$, which, in the case of convex functions, provide supporting hyperplanes [23]. Given a set of solutions $\bar{x}^k$, $k = 1, \ldots, i-1$, the i-th MIP problem generated by the outer-approximation algorithm is as follows:

$$\begin{aligned} \min_{x,y,\alpha} \quad & \alpha, \\ \text{s.t.} \quad & f(\bar{x}^k, \bar{y}^k) + \nabla f(\bar{x}^k, \bar{y}^k)^\top \begin{pmatrix} x - \bar{x}^k \\ y - \bar{y}^k \end{pmatrix} \leq \alpha, && k = 0, \ldots, i-1, \\ & g_l(\bar{x}^k, \bar{y}^k) + \nabla g_l(\bar{x}^k, \bar{y}^k)^\top \begin{pmatrix} x - \bar{x}^k \\ y - \bar{y}^k \end{pmatrix} \leq 0, && l \in L^k, k = 0, \ldots, i-1, \\ & \|y - \bar{y}^k\|_1 \geq 1, && k \in C^i, \\ & x \in X,\ y \in Y \cap \mathbb{Z}^{n_y},\ \alpha \in \mathbb{R}, \end{aligned} \tag{MIP^i}$$

where $L^k \subseteq \{1, \ldots, m\}$ is a subset of constraints for which linearizations are included ($L^0 = \{1, \ldots, m\}$, typically), and $C^i \subseteq \{1, \ldots, i-1\}$ is a subset of iterations in which the so-called *integer cut* $\|y - \bar{y}^k\|_1$ is added [2] (discussed below). Note, that due to assumption (A1), the equation $\|y - \bar{y}^k\|_1 \geq 1$ can be written in an equivalent linear form, see Appendix A. (MIP^i) is called the *master problem*. We denote by $(\hat{\alpha}^i, \hat{x}^i, \hat{y}^i)$ a solution for (MIP^i), if feasible. Due to assumption (A2), the optimal value of (MIP^i) yields a lower bound to the optimal value of (MINLP), if $C^i = \emptyset$ (for now).

The solution of (MIP^i) is used to define the following NLP subproblem of MINLP, obtained by fixing the integer variables to $\hat{y}^i$:

$$\begin{aligned} \min_{x} \quad & f(x, \hat{y}^i), \\ \text{s.t.} \quad & g(x, \hat{y}^i) \leq 0, \\ & x \in X. \end{aligned} \tag{NLP^i}$$

Let $\bar{y}^i := \hat{y}^i$ and let $\bar{x}^i$ be a solution to (NLP^i), if feasible. Then $(\bar{x}^i, \bar{y}^i)$ is a feasible point to (MINLP) and provides an upper bound on its optimal value. If (NLP^i) is not feasible, then let $(\bar{x}^i, \bar{s}^i)$ be a minimal infeasible solution to (NLP^i), that is, a solution to the NLP

$$\begin{aligned} \min_{x,s} \quad & \sum_{j=1}^{m} s_j, \\ \text{s.t.} \quad & g(x, \hat{y}^i) \leq s, \\ & x \in X,\ s \in \mathbb{R}^m_+. \end{aligned} \tag{NLP-feas^i}$$

Note that adding linearization of $g_j(x, y)$ in $(\bar{x}^i, \bar{y}^i)$ for those $j \in \{1, \ldots, m\}$ with $g_j(\bar{x}^i, \bar{y}^i) > 0$ to (MIP^i) will eliminate $(\bar{x}^i, \bar{y}^i)$ from its feasible set. However, for the case that (NLP^i) is feasible and the corresponding linearization are added there may exist some other values of x for which $(x, \bar{y}^i)$ is still feasible for (MIP^i). Therefore, one may, additionally or alternatively, add the integer-cut $\|y - \hat{y}^i\|_1 \geq 1$ to (MIP^i) to cut off any point in $\mathbb{R}^{n_x} \times \{\hat{y}^i\}$. Hence, if only those iterations are included into C^i for which (NLP^i) is infeasible, then the optimal value of (MIP^i) provides a lower bound to the optimal value of (MINLP).

The outer-approximation algorithm is summarized in Algorithm 1. The NLP and MIP problems are solved alternately until the gap between the bounds given by (NLP^i) and (MIP^i) is less than the specified tolerance. It has been proved that this algorithm finds an ϵ-optimal solution of a convex MINLP or proves that none exist in a finite number of iterations if the solution of every (NLP^i) and (NLP-feas^i) satisfies the linear independence constraint qualification [5, 11, 13], that is, the gradients of the constraints satisfied with equality are linearly independent.

2.2 Outer-approximation in DICOPT

Outer-approximation is the main algorithm behind the solver DICOPT [17, 18, 23], which has been developed in the late 1980s by the research group of I.E. Grossmann at the Engineering Research Design Center at Carnegie Mellon University. Since then, it has been available in the commercial algebraic modeling system GAMS. DICOPT solves NLP and MIP problems by means of other solvers that are available in GAMS and are specialized in these problem types.

Although the outer approximation algorithm has a guaranteed convergence for convex MINLP problems [11], DICOPT implements the methods of equality relaxation and augmented penalty to make it a better heuristic for solving nonconvex MINLP problems. The implementation of the outer-approximation algorithm does therefore slightly deviate from Algorithm 1:

- For convex MINLP, Z^L and Z^U yield valid lower and upper bounds on the optimal value of (MINLP) given that (NLP^i) can be solved to global optimality. Therefore, closing the gap between these bounds is a stopping criterion that ensures finding a global optimal solution in a finite number of iterations. This can be enabled in DICOPT by setting the option `stop` to 1. For nonconvex MINLPs, valid lower bounds and solving (NLP^i) to global optimality are not ensured. Therefore, by default DICOPT stops as soon as the upper bound Z^U stops improving. Although it is a heuristic, this stopping criterion has shown that in many cases it yields optimal or near optimal integer solutions. For the computational experiments in this paper, we have set option `stop` to 1.

- If the NLP subproblem (NLP^i) is feasible, linearizations of nonlinear functions are not added in their original form to (MIP^i). Instead, they are added as soft-constraints, that is, violation of these constraints is allowed but penalized in the objective function (by default, a weight ρ of 1000 times the constraint marginal is used) [23]. Also in the context of convex MINLP, the penalty relaxation of linearizations is applied. Note, that the optimal value of the modified master problem still provides a valid lower bound on the optimal value of (MINLP) if the contribution of the penalty term is removed and termination is still ensured due to the finite number of integer points y to be enumerated. This modification leads to the following MIP

Algorithm 1 Outer-approximation algorithm.

1: Set $Z^U = \infty$, $Z^L = -\infty$, $i = 0$ ▷ Initialization
2: Define gap tolerance $\epsilon \geq 0$
3: Solve (rMINLP) ▷ Solve initial relaxation
4: **if** (rMINLP) is infeasible **then**
5: Set $Z^L = \infty$ ▷ (MINLP) is infeasible
6: **else**
7: Let $(\bar{x}^0, \bar{y}^0)$ be an optimal solution of (rMINLP)
8: Set $Z^L = f(\bar{x}^0, \bar{y}^0)$
9: Set $L^0 = \{1, \ldots, m\}$, $C^0 = \emptyset$
10: **if** $\bar{y}^0 \in \mathbb{Z}^{n_y}$ **then**
11: Set $Z^U = f(\bar{x}^0, \bar{y}^0)$ and $\hat{y}^0 = \bar{y}^0$
12: **while** $Z^U - Z^L > \epsilon$ **do**
13: Set $i = i + 1$
14: Solve (MIP^i) ▷ Solve master problem
15: **if** (MIP^i) is infeasible **then**
16: Set $Z^L = \infty$ ▷ (MINLP) is infeasible
17: **else**
18: **if** (MIP^i) is unbounded **then**
19: Let $(\hat{\alpha}^i, \hat{x}^i, \hat{y}^i)$ be a feasible solution of (MIP^i)
20: **else**
21: Let $(\hat{\alpha}^i, \hat{x}^i, \hat{y}^i)$ be an optimal solution of (MIP^i)
22: Set $Z^L = \hat{\alpha}^i$
23: Set $\bar{y}^i = \hat{y}^i$ and solve (NLP^i) ▷ Solve nonlinear subproblem
24: **if** (NLP^i) is infeasible **then**
25: Solve (NLP-feas^i)
26: Let $(\bar{x}^i, \bar{s}^i)$ be an optimal solution of (NLP-feas^i)
27: Set $C^{i+1} = C^i \cup \{i\}$
28: **else**
29: Let $\bar{x}^i$ be an optimal solution of (NLP^i)
30: Set $C^{i+1} = C^i$
31: **if** $Z^U < f(\bar{x}^i, \hat{y}^i)$ **then**
32: Set $Z^U = f(\bar{x}^i, \hat{y}^i)$, $x^* = \bar{x}^i$ and $y^* = \hat{y}^i$
33: Set $L^i = \{j \in \{1, \ldots, m\} \,:\, g_j(\bar{x}^i, \hat{y}^i) \geq 0 \text{ and } g_j \text{ is nonlinear}\}$
34: (x^*, y^*) is an optimal solution of (MINLP), if $Z^U < \infty$, otherwise (MINLP) is infeasible

master problem:

$$
\begin{aligned}
\min_{x,y,\alpha,s} \quad & \alpha + \sum_{l \in L_k} \rho_l s_l, \\
\text{s.t.} \quad & f(\bar{x}^k, \bar{y}^k) + \nabla f(\bar{x}^k, \bar{y}^k)^\top \begin{pmatrix} x - \bar{x}^k \\ y - \bar{y}^k \end{pmatrix} \leq \alpha, && k = 0, \ldots, i-1, \\
& g_l(\bar{x}^k, \bar{y}^k) + \nabla g_l(\bar{x}^k, \bar{y}^k)^\top \begin{pmatrix} x - \bar{x}^k \\ y - \bar{y}^k \end{pmatrix} \leq s_l, && l \in L^k, k = 0, \ldots, i-1, \\
& \|y - \bar{y}^k\|_1 \geq 1, && k \in C^i, \\
& x \in X,\ y \in Y \cap \mathbb{Z}^{n_y},\ \alpha \in \mathbb{R}, s \in \mathbb{R}_+^{\sum_{k=0}^{i-1} |L_k|}.
\end{aligned}
\tag{rMIP^i}
$$

- If the NLP subproblem (NLP^i) is infeasible, DICOPT by default adds only an integer cut to eliminate the current fixing $y = \hat{y}^i$ from (rMIP^i), but does not add the corresponding linearizations of nonlinear functions, i.e., $L^i = \emptyset$ if (NLP^i) is infeasible in Line 33 of

Algorithm 1. This option is sufficient to avoid visiting the same solution point again while avoiding adding linearizations which are not supporting hyperplanes for nonconvex MINLPs. However, it also exhibits slower progress as less information is made available to the master problem. Thus, when solving a convex MINLP, these valid linearizations should be added. This can be enabled in DICOPT by setting the option `infeasder` to 1. We do so in our computational experiments.

- Finally, DICOPT relaxes nonlinear equality constraints to inequalities and adds corresponding linearizations to (rMIP^i). The dual multipliers in the solution of (NLP^i) are used to decide in which direction to relax the inequalities [23]. For a convex MINLP, such constraints do not appear.

2.3 Feasibility pump

The feasibility pump algorithm is a primal heuristic developed by Fischetti, Glover, and Lodi to quickly find feasible solutions for MIPs where all integer variables are binaries [12]. Extensions and variations of the algorithm have been proposed, including an extension to general integer variables [3]. Nowadays, many state-of-the-art commercial and non-commercial MIP solvers feature implementations of the feasibility pump [3]. The first extension of the feasibility pump algorithm to convex MINLP problems was introduced by Bonami, Cornuéjols, Lodi, and Margot [6]. Contrary to the original feasibility pump for MIP [12], the feasibility pump for convex MINLP is guaranteed to converge to a feasible solution, if any. Subsequently, several authors have proposed extensions to nonconvex MINLPs [4, 9, 14], where the handling of the nonconvex nonlinear constraints poses an additional challenge. The MINLP solvers BONMIN and Couenne have implemented feasibility pump algorithms as primal heuristics [4, 6].

The main idea of this algorithm is to decompose the original mixed-integer problem into two parts: integer feasibility and constraint feasibility. For convex MINLPs, a MIP is solved to obtain a solution, which satisfies the integrality constraints on y, but may violate some of the nonlinear constraints; next, by solving an NLP, a solution is computed that satisfies the constraints ($g(x, y) \leq 0$) but might again violate the integrality constraints on y. By minimizing iteratively the distance between these two types of solutions, a solution that is both constraint- and integer-feasible can be expected. The first iteration of the algorithm proposed in [6] is the same as in Algorithm 1, where the continuous relaxation (rMINLP) of the original MINLP problem is solved. Following this, the next iteration builds a MIP master problem with the outer-approximation linearization of the nonlinear constraints and a modified objective function called the Feasibility Outer-Approximation:

$$\begin{aligned} \min_{x,y} \quad & \|y - \bar{y}^{i-1}\|_1, \\ \text{s.t.} \quad & g_l(\bar{x}^k, \bar{y}^k) + \nabla g_l(\bar{x}^k, \bar{y}^k)^\top \begin{pmatrix} x - \bar{x}^k \\ y - \bar{y}^k \end{pmatrix} \leq 0, \quad l \in L^k, k = 0, \ldots, i-1, \\ & x \in X,\ y \in Y \cap \mathbb{Z}^{n_y}, \end{aligned} \tag{FOA^i}$$

where $L^k \subseteq \{1, \ldots, m\}$ is chosen as in Algorithm 1. The solution to this problem is denoted as $(\hat{x}^i, \hat{y}^i)$. In (FOA^i), the original objective function has been replaced by the L_1-distance of y to $\bar{y}^{i-1}$. In the first iteration, $\bar{y}^0$ corresponds to the solution of the continuous relaxation (rMINLP) of (MINLP). However, in the following iterations, $\bar{y}^{i-1}$ is given by the solution of the following nonlinear program for the feasibility pump:

$$\begin{aligned} \min_{x,y} \quad & \|y - \hat{y}^{i-1}\|_2^2, \\ \text{s.t.} \quad & g(x, y) \leq 0, \\ & x \in X,\ y \in Y. \end{aligned} \tag{FP-NLP^i}$$

The solution of this problem is denoted as $(\bar{x}^i, \bar{y}^i)$. If $\bar{y}^i \in \mathbb{Z}^{n_y}$, a feasible solution for (MINLP) has been found.

Bonami et al. [6] have shown on an example that this basic algorithm can cycle ($(\bar{x}^{i-1}, \bar{y}^{i-1}) = (\bar{x}^i, \bar{y}^i)$) if the linear independence constraint qualification is not satisfied. A possibility to avoid this cycling is to add the cut

$$(\bar{y}^i - \hat{y}^{i-1})^\top (y - \bar{y}^i) \geq 0 \tag{1}$$

to (FOA^i). Since (FP-NLP^i) projects the solution $\hat{y}^{i-1}$ onto the convex set $\{y \in Y : \exists x \in X : g(x, y) \leq 0\}$, the cut (1) outer-approximates the feasible region of (MINLP) and is violated by $\hat{y}^{i-1}$ (unless $\bar{y}^i = \hat{y}^{i-1}$, in which case $(\bar{x}^i, \bar{y}^i)$ is a feasible solution for (MINLP)). Thus, adding it to (FOA^i) avoids revisiting $\hat{y}^{i-1}$. This algorithm is denoted as *enhanced Feasibility Pump* in [6] and has been shown to find a feasible solution to (MINLP) or prove that none exist in a finite number of iterations, if assumptions (A1) and (A2) are satisfied.

To find further (and better) feasible solutions, the feasibility pump can be applied iteratively, thereby excluding solutions for which the (linearized) objective function has a worse value than the best known value. This is achieved by the following modification to (FOA^i):

$$\begin{aligned}
\min_{x,y} \quad & \|y - \bar{y}^{i-1}\|_1, \\
\text{s.t.} \quad & f(\bar{x}^k, \bar{y}^k) + \nabla f(\bar{x}^k, \bar{y}^k)^\top \begin{pmatrix} x - \bar{x}^k \\ y - \bar{y}^k \end{pmatrix} \leq \alpha, \qquad k = 0, \ldots, i-1, \\
& g_l(\bar{x}^k, \bar{y}^k) + \nabla g_l(\bar{x}^k, \bar{y}^k)^\top \begin{pmatrix} x - \bar{x}^k \\ y - \bar{y}^k \end{pmatrix} \leq 0, \quad l \in L^k, k = 0, \ldots, i-1, \\
& \alpha \leq Z^U - \delta, \\
& x \in X,\ y \in Y \cap \mathbb{Z}^{n_y},\ \alpha \in \mathbb{R}.
\end{aligned}$$

The variable α is initially unbounded ($Z^U = \infty$). When a new incumbent is found, Z^U is updated to the value of the original objective function in the incumbent. The small positive constant δ ensures that the incumbent becomes infeasible in (FP-OA^i) and enforces the search for an improved solution. If (MINLP) is feasible, (A2) and (A3) are satisfied, and the linear independence constraint qualificiation holds for (FP-NLP^i) at $(\bar{x}^i, \bar{y}^i)$, then this iterative algorithm finds a δ-optimal solution [6].

3 Proposed Algorithm

While the main focus of the outer-approximation algorithm is to find an optimal solution and proving its optimality, the feasibility pump algorithm mostly disregards the original objective function and focuses primarily on simultaneously minimizing violation of integrality and nonlinear constraints. Therefore, the outer-approximation algorithm may take longer to find feasible solutions on problems where feasible solutions are difficult to find, while the (iterative) feasibility pump algorithm may take longer to find a (proven) optimal solution on problems with many feasible points. To alleviate and explore the differences between these algorithms, hybrid algorithms have been designed, the first one being in [6]. In [6], the feasibility pump algorithm is called when the NLP subproblem (NLP^i) is found to be infeasible.

For DICOPT, we implemented a variation of this hybrid algorithm. Instead of starting the feasibility pump for one or several times within the outer-approximation algorithm, we run the iterative feasibility pump once before the main outer-approximation loop starts. Furthermore, we slightly modified the feasibility pump algorithm in the following way.

A drawback of neglecting the original objective function in the feasibility pump algorithm as stated in Section 2.3 is that although it may be successful in finding feasible solutions, the quality of solutions in terms of the objective function value can be poor [1]. Therefore, as in [6], after finding a feasible solution by means of solving (FP-NLP^i), we try to improve it further by solving the NLP subproblem obtained from fixing all integer variables in (MINLP) to the values in the solution of (FP-NLP^i) (that is, we solve (NLP^i) with $\hat{y}^i$ replaced by $\bar{y}^i$).

Another problem arises from the possibility of repeating the same values in the integer variables ($\hat{y}^{i-1} = \hat{y}^i$), either due to cycling or when several feasible solutions with the same values in the

integer variables exist. The former is avoided by adding the cut (1), if $\bar{y}^i \neq \hat{y}^{i-1}$, as proposed by [6]. If, however, $\bar{y}^i = \hat{y}^{i-1}$, then a feasible solution for (MINLP) has been found and we can add the integer cut $\|y - \hat{y}^i\| \geq 1$. With $\delta > 0$, this would not be necessary to ensure progress in the search for an improving solution. However, we observe that in certain instances adding this cut accelerates the search. Since the linearization of integer cuts may require additional variables if general integer variables are present (see Appendix A), integer cuts are by default only added for mixed-binary problems ($Y = [0,1]^{n_y}$). To summarize, the MIP projection problem that we solve is the following:

$$
\begin{aligned}
\min_{x,y} \quad & \|y - \bar{y}^{i-1}\|_1, \\
\text{s.t.} \quad & f(\bar{x}^k, \bar{y}^k) + \nabla f(\bar{x}^k, \bar{y}^k)^\top \begin{pmatrix} x - \bar{x}^k \\ y - \bar{y}^k \end{pmatrix} \leq \alpha, && k = 0, \ldots, i-1, \\
& g_l(\bar{x}^k, \bar{y}^k) + \nabla g_l(\bar{x}^k, \bar{y}^k)^\top \begin{pmatrix} x - \bar{x}^k \\ y - \bar{y}^k \end{pmatrix} \leq 0, && l \in L^k, k = 0, \ldots, i-1, \\
& \|y - \bar{y}^k\|_1 \geq 1, && k \in C^i, \\
& (\bar{y}^k - \hat{y}^k)^\top (y^k - \bar{y}^k) \geq 0, && k = 1, \ldots, i-1, \\
& \alpha \leq Z^U - \delta \max(|Z^U|, 1), \\
& x \in X,\ y \in Y \cap \mathbb{Z}^{n_y},\ \alpha \in \mathbb{R}.
\end{aligned}
\tag{FP-OA^i}
$$

Finally, similar to (FP-OA^i), we add the constraint $f(x,y) \leq Z^U - \delta \max(|Z^U|, 1)$ to (FP-NLP^i) in order to avoid non-improving solutions.

When the feasibility pump terminates, the outer-approximation algorithm is initialized not only by the best solution that the feasibility pump may have found, but also with the linearizations, integer cuts, and cuts (1) that have been added to (FP-OA^i). However, regarding the cuts (1), only those generated before the last incumbent solution has been found can be used to initialize the outer-approximation algorithm, since subsequent cuts were generated with respect to the additional constraint $f(x,y) \leq Z^U - \delta \max(|Z^U|, 1)$, which may cut off an optimal solution (x^*, y^*) if $Z^U - \delta \max(|Z^U|, 1) < f(x^*, y^*) < Z^U$.

A general outline of the proposed algorithm is given in Algorithm 2. Up until Line 27 of Algorithm 2, the algorithm is similar to the Iterated Feasibility Pump (IFP) for MINLP proposed in [6]. The main differences with the IFP are the optional inclusion of the integer cuts and the solution of problem (NLP^i) in Line 19. Therefore, we can argue that executing lines 1–27 of Algorithm 2 with $i_{\max} = \infty$ finds a δ-optimal ($\delta > 0$) solution to (MINLP), or proves that none exist if assumptions (A2) and (A3) are satisfied (see Theorem 2 in [6]). To find an optimal solution ($\delta = 0$), if any exists, integer cuts need to be generated also if general integer variables are present. To be able to add these in linear form, also Assumption (A1) needs to be satisfied.

Contrary to the Enhanced Outer Approximation method presented by Bonami et al. [6], which runs the feasibility pump both as starting procedure and when the NLP subproblem (NLP^i) is infeasible, we are employing the feasibility pump only once and before the outer-approximation algorithm implemented in DICOPT. This is motivated by the fact that the MIP (FP-OA^i), which is built by the feasibility pump, provides a valid relaxation for the convex MINLP. Therefore, the feasibility pump does not only provide an initial feasible solution if successful, but also an initialization of the MIP relaxation (rMIP^i) in any case.

Algorithm 2 has been implemented as part of the solver DICOPT and is available in GAMS since version 25.1 (an earlier version without cuts (1) is available since GAMS 24.5). To enable and adjust the algorithm, a number of options were added, which are summarized in Table 1. As DICOPT is often used for nonconvex MINLPs, as commented in Section 2.2, the default values for options `convex` and `feaspump` are 0.

Algorithm 2 Proposed algorithm.

Set $Z^U = \infty$, $i = 0$ ▷ Initialization
Define cutoff decrease $\delta \geq 0$
Solve (rMINLP) ▷ Solve initial relaxation
if (rMINLP) is infeasible **then**
    Stop ▷ (MINLP) is infeasible
Let $(\bar{x}^0, \bar{y}^0)$ be an optimal solution of (rMINLP)
Set $L^0 = \{1, \ldots, m\}$, $C^0 = \emptyset$
Set $Z^L = f(\bar{x}^0, \bar{y}^0)$
if $\bar{y}^0 \in \mathbb{Z}^{n_y}$ **then**
    Set $Z^U = f(\bar{x}^0, \bar{y}^0)$ ▷ Optimal solution found
    Stop
Set $i = 1$
Solve (FP-OA^i) ▷ Solve feasibility OA problem
while (FP-OA^i) is feasible and $i \leq i_{\max}$ **do**
    Let $(\hat{x}^i, \hat{y}^i)$ be an optimal (or feasible if unbounded) solution of (FP-OA^i)
    Solve (FP-NLP^i) ▷ Solve nonlinear feasibility problem
    Let $(\bar{x}^i, \bar{y}^i)$ be an optimal solution of (FP-NLP^i)
    if $\|\bar{y}^i - \hat{y}^i\| = 0$ **then**
        Solve (NLP^i) ▷ Solve nonlinear subproblem
        Let $\bar{x}^i$ be an optimal solution of (NLP^i)
        Set $Z^U = \min(Z^U, f(\bar{x}^i, \bar{y}^i))$ ▷ New incumbent solution
        Set $C^{i+1} = C^i \cup \{i\}$ (if $y \in \{0,1\}^{n_y}$ in (MINLP))
    else
        Set $C^{i+1} = C^i$
    Set $L^i = \{j \in \{1, \ldots, m\} \ : \ g_j(\bar{x}^i, \bar{y}^i) \geq 0$ and g_j is nonlinear$\}$
    Set $i = i + 1$
    Solve (FP-OA^i) ▷ Solve feasibility OA problem
Solve (MINLP) using DICOPT's modification of Alg. 1, initialized with incumbent solution $(\bar{x}^i, \bar{y}^i)$, if $Z^U < \infty$, and linearizations given by $L^0, \cdots, L^i$, integer cuts given by C^i, and cuts (1) in the relaxation (rMIP^i).

4 Computational results

In the following, we evaluate the benefits of adding the feasibility pump to DICOPT on a set of convex MINLPs selected from MINLPLib (version `c0f77612`, as of 13.3.2018)[2] [22]. First, we selected all instances that are marked as convex, are not proven to be infeasible, have at least one binary or general integer variable, no semi-continuous or semi-integer variables, and no special-ordered-sets. This gives a set of 359 instances. Second, we run DICOPT with the `convex` option enabled, and the feasibility pump disabled, and removed all instances for which DICOPT terminated in less than one second. In this remaining set, a strong dominance of some subsets of instances that were clearly derived from the same model (strong similarity in name) was observed. Therefore, we reduced these subsets to the four largest instances. This leaves a final set of 80 instances, which have their origin in a wide variety of applications, ranging from process synthesis flowsheets, facilities layout problems, batch design with storage, water treatment models, and investment portfolios. Table 1 in the Supplemental material provides this list of instances.

For all the experiments, we used a time limit of 1800 seconds and set the GAMS gap tolerance `optcr` (relative distance of Z^L and Z^U) to 10^{-5}. GAMS 25.1.1 was run on a cluster of Dell PowerEdge M620 blades with 64 GB RAM, Intel Xeon E5-2680 CPUs running at 2.70 GHz, and Linux 4.4.0 (64bit). With this GAMS version, DICOPT uses CPLEX 12.8.0.0 for solving the MIPs,

[2] `http://www.minlplib.org`

Table 1: Feasibility pump options in DICOPT

Option	Description	Default
`convex`	If enabled, then the default values for the following options are changed to be more appropriate for convex MINLPs, see also Section 2.2: option `stop` is set to 1, option `infeasder` is set to 1, and option `feaspump` is set to 1	0
`feaspump`	Whether to run the feasibility pump	0
`fp_iterlimit`	Major iteration limit ($i_{\max}$) in the feasibility pump	20
`fp_timelimit`	Time limit in the feasibility pump	∞
`fp_sollimit`	Limit on number of (improving) solutions found by the feasibility pump	∞
`fp_stalllimit`	Limit on the number of consecutive iterations where no improving solution is found. Only applies after a first solution has been found.	5
`fp_cutoffdecr`	Relative decrease of cutoff value for objective variable (δ)	0.1
`fp_acttol`	Tolerance on when a constraint is found active	10^{-6}
`fp_projzerotol`	Tolerance on when to consider the difference $\|\bar{y}^i - \hat{y}^i\|$ as zero	10^{-4}
`fp_mipgap`	Optimality tolerance (relative gap) when solving (FP-OA^i)	0.01
`fp_transfercuts`	Whether to transfer cuts from the feasibility pump MIP to the DICOPT MIP	1
`fp_integercuts`	Whether to add integer cuts to (FP-OA^i) when finding a new feasible solution	1
`fp_projcuts`	Whether to add cuts (1) to (FP-OA^i) after solving (FP-NLP^i)	1

and CONOPT 3.17I for solving the NLPs. We used PAVER 2 [7] to help in the evaluation.

4.1 Illustrative example

Before evaluating the performance of the new feasibility pump on the complete test set, we discuss its behavior for a single instance. This instance corresponds to the block layout design problem with unequal areas. The original problem was proposed by Meller et al. [20] and was reformulated by Castillo et al. [8] as a convex MINLP. This type of problems may be applied in piping design problems and in process plants layouts. The complete formulation of this model is reported in [8]. The test case selected was the block layout design problem of 7 departments and with an aspect ratio (the maximum permissible ratio between its longest and shortest dimensions) of 5. The problem involves 211 constraints, of which 14 are nonlinear, specifically signomial, and 114 variables, of which 42 are binary. This instance can be found in MINLPLib under the name `o7_2`[3]. The authors of the model used several MINLP solvers to find the optimal solution to this problem, among them DICOPT. DICOPT performed very poorly because the linearizations in the initial outer-approximation (rMIP^i) were not helpful and many nonlinear subproblems (NLP^i) are infeasible [8].

The given instance was tested using different options for DICOPT. The stopping criterion for all the different options was closing the gap between the objective values of the MIP master problem and the incumbent solution. The default setting for option `infeasder` requires that if the nonlinear subproblem (NLP^i) is infeasible, only a corresponding integer cut is added to (rMIP^i). This approach, although rigorous for convex and non-convex MINLPs, is not very efficient, particularly for this type of problems where "a significant amount of integer cuts may be required before a feasible solution is obtained" [8]. For convex MINLPs another rigorous approach is to add

[3] `http://www.minlplib.org/o7_2.html`

Table 2: Results of the solution of the illustrative example o7_2 for each setting of DICOPT.

DICOPT options	w/o FP w/o infeasder	w/o FP w/ infeasder	w/ FP w/o infeasder	w/ FP w/ infeasder
major iterations	113	7	2	2
feasible solutions found	0	1	5	5
FP iterations	0	0	12	12
FP time [s]	0	0	199.4	200.1
infeasible NLP	112	5	0	0
time to optimal sol. [s]	–	676.6	417.3	418.4
solution time [s]	> 1800*	915.5	620.9	621.2
final objective value	–	116.95	116.95	116.95

*Time limit reached.

linearization cuts if the nonlinear subproblem is infeasible, using the solution of (NLP-feas^i) as a reference point. This can be enabled by using the option `infeasder`, see Section 2.2. Note that the setting of the `infeasder` option does not influence the handling of infeasible NLPs within the feasibility pump. A comparison of DICOPT on instance `o7_2` with the feasibility pump and the `infeasder` option enabled and disabled is given in Table 2.

We notice that DICOPT without feasibility pump and with `infeasder` disabled cannot find a feasible solution within 30 minutes. During this time the solver performed 113 major iterations. That is, at each iteration, it solved a MIP master problem (rMIP^i) and an NLP subproblem (NLP^i). All NLP subproblems were infeasible. Enabling the `infeasder` option, the problem could be solved in 912 seconds. During this time, 5 out of the 6 solved NLP subproblems were infeasible. The only feasible solution, found in the 7th iteration, was also an optimal solution to the problem. It required another solution of the MIP to prove its optimality.

The use of the feasibility pump allowed the solver to find 4 feasible solutions in the first ≈ 200 seconds. After that, a single major iteration was required to find an optimal solution, which required 217 seconds in both cases with and without the `infeasder` option. In that same major iteration, optimality of the solution was proven. The results when using the feasibility pump with and without the `infeasder` option are the same (except for variations in time measurement) since none of the NLP subproblems (NLP^i) in the outer-approximation algorithm were infeasible.

These results highlight that first, enabling the `infeasder` option can be essential to solving a problem or just finding a feasible solution. Second, the feasibility pump can further improve the performance by finding feasible solutions early. That is, we obtained a 38% reduction in the time needed to find an optimal solution to the problem and a 32% reduction in the complete solution time by enabling the feasibility pump. It is also interesting to note that when this problem is solved with AlphaECP it required 897 seconds, with BONMIN 756 seconds, and with SCIP 768 seconds. When running only the iterative feasibility pump algorithm of BONMIN [6], no feasible solution is obtained before hitting the time limit of 30 minutes.

4.2 Feasibility pump alone

In the following, we consider the full test set of 80 instances. First, we run only our (iterative) feasibility pump implementation with various settings, that is, without continuing with the outer-approximation algorithm of DICOPT. In setting "default", the feasibility pump is run in its default settings, see Table 1, that is, a stall limit of 5 and a cutoff decrease of $\delta = 0.1$, except that the iteration limit has been disabled ($i_{\max} = \infty$). A stall limit of k iterations stops the feasibility pump if, after a first solution has been found, no improving solution is found within the next k iterations. In setting "stall10", we increased the stall limit to 10. In setting "no cuts (1)", we disabled the addition of cuts (1) to (FP-OA^i) after having solved the NLP projection problem (FP-NLP^i). Setting "no integer cuts" completely disables the addition of integer cuts when a new

Table 3: Results of running feasibility pump alone with different settings. For each setting, we show the number of instances in which the feasibility pump reaches the time limit, found a δ-optimal solution (without necessarily proving optimality), found a solution with primal gap $\leq 10\%$, found any feasible solution, and the time used, respectively.

setting	timeout	optimal	good sol.	feasible	mean time [s]
default	4	12	57	77	9.1
stall10	4	14	64	77	11.4
no cuts (1)	5	13	55	76	9.6
no integer cuts	4	13	60	77	9.8
findopt	25	67	74	77	119.0
findopt w/o integer cuts	42	44	52	77	238.1
findopt w/ all integer cuts	29	69	74	77	197.1

feasible solution has been found ($C^{i+1} = \emptyset$ in Line 22 of Algorithm 2). Finally, we evaluated three settings that target to find δ-optimal solutions of the MINLP. For this, setting "findopt" disables the stall limit and sets the cutoff decrease δ to 10^{-5}. Setting "findopt w/o integer cuts" additionally disables the addition of integer cuts when a new solution is found, while setting "findopt w/ all integer cuts" enables the addition of integer cuts also for problems with general integer variables.

Table 3 summarizes the results for all settings and detailed results are given in Tables B2 and B3 in the Supplemental material. The mean time in Table 3 reports the shifted geometric mean of the runtimes $(t_1, \ldots, t_{80})$ of the feasibility pump on all instances, computed as $\prod_{i=1}^{80}(t_i + 1)^{\frac{1}{80}} - 1$. Figure 1 plots the primal gap of all runs for settings "default", "stall10", and "findopt". As primal gap, we compute the relative distance between the objective function value of the best solution found by the algorithm and the objective function value of the best known solution reported in MINLPLib. We can observe that the feasibility pump in default settings finds an optimal solution for 12 instances, good solutions ($< 10\%$ primal gap) for another 45 instances, and some feasible solutions ($\geq 10\%$ primal gap) for another 20 instances. Increasing the stall limit helps on seven of the instances where previously only bad solutions were found. On instances where good solutions were already found in default settings, increasing the stall limit has an effect on one instance only, likely since the cutoff decrease δ cuts off solutions that are only slightly better or optimal. However, increasing the stall limit also increases the mean running time by 25%. By using the "findopt" setting, however, the feasibility pump is able to find optimal solutions for many instances where previously a small gap was remaining.

For the runs with stall limit ("default" and "stall10"), the feasibility pump usually terminates either when the MIP approximation (FP-OA^i) becomes infeasible or the stall limit is reached. In the "findopt" setting, however, 25 instances terminated when the time limit of 1800 seconds was reached. Thus, the feasibility pump is not suited to prove optimality of the found solutions. This justifies the choice of the stall limit as a stopping criterion.

Disabling cuts (1) has a small negative impact on performance. Without this cut, the feasibility pump fails to find a solution for instance `tls7` within the allowed time, which then also leads to an increase in the mean time. As noted by [6], cut (1) was not necessary in practice, though adding it is unlikely to have a negative effect. Also disabling the integer cuts has little impact on the performance. The number of instances with optimal and good solution increases slightly, but the mean running time also increases slightly. However, disabling integer cuts in the "findopt" setting has a severe impact on the performance, since, without these cuts, only the cutoff decrease δ, which is only 10^{-5} in this setting, is responsible to force the feasibility pump to look for better feasible solutions. On instances with general integer variables, integer cuts are already disabled by default, which is the reason why there is one instance (second instance in Figure 1) where the "findopt" setting produces a worse solution than "default". Hence, enabling integer cuts also for instances with general integer variables improves solution quality, at the cost of a considerably increased running time.

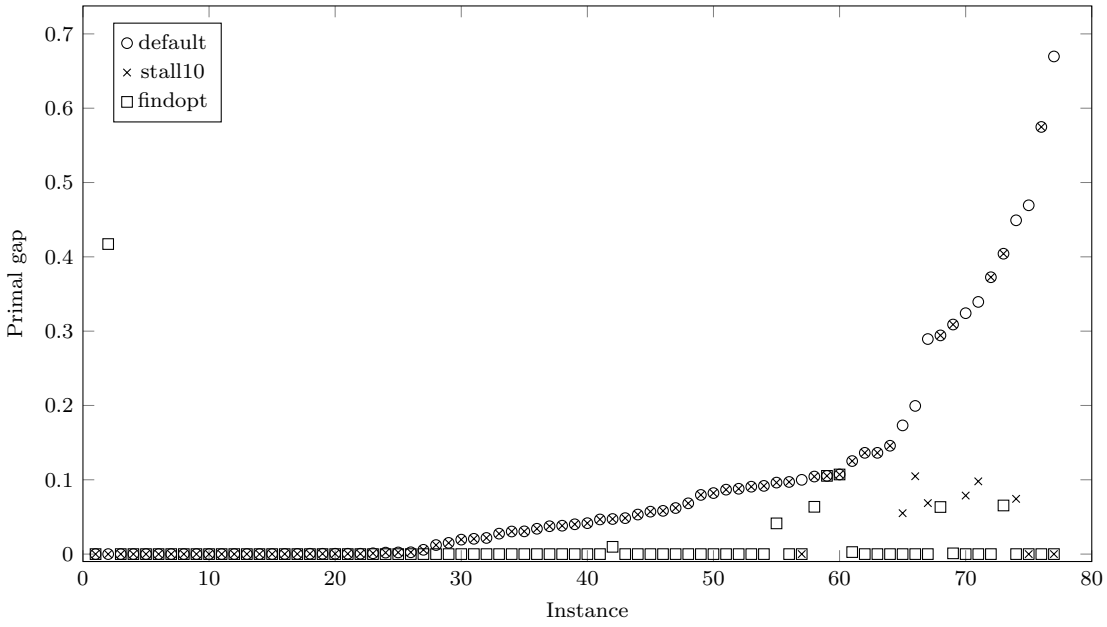

Figure 1: Primal gap of solutions found by feasibility pump (with different settings) for all instances in test set, sorted by primal gap of "default" setting.

4.3 DICOPT with feasibility pump

We used DICOPT with the following settings: In the "DICOPT w/ FP" setting, the option `convex` was enabled, which also enables the feasibility pump. In the "DICOPT w/o FP" setting, the option `convex` was also enabled, but the feasibility pump disabled. In the "DICOPT w/ FP w/o OA init" setting, option `convex` was again enabled, but the transfer of cuts from the feasibility pump MIP (FP-OA^i) to the outer-approximation MIP (rMIP^i) has been disabled. Additionally, with "FP only" we consider the results from running only the feasibility pump without stall limit and cutoff decrease $\delta = 10^{-5}$ ("findopt" setting in Section 4.2).

Table 4 summarizes for each setting the number of instances for which the time limit was reached, a δ-optimal solution was found, optimality was proven, a good solution was found (primal gap $\leq 10\%$), and mean running time. Detailed results are given in Tables B3 and B4 in the Supplemental material. Performance profiles are shown in Figures 2 and 3. The numbers show that adding the feasibility pump to DICOPT leads to finding optimal solutions to three more instances than before and slightly reducing the mean running time, but does not affect the number of instances for which optimality is proven. Running the feasibility pump alone increases the number of instances where optimal or good solutions are found and even decreases the mean running time, but decreases the number of instances where optimality is proven. That is, the best performance in terms of finding proven optimal solutions can only be expected when combining the feasibility pump as primal heuristic and outer-approximation to prove optimality. We also note that the outer-approximation algorithm in DICOPT is targeted for general MINLPs, which results in applying linearizations of nonlinear functions, also if convex, as soft-constraints only, cf. Section 2.2. That is, tuning the implementation of the outer-approximation in DICOPT to work better in case of a convex MINLP might lead to achieving the best performance of DICOPT (with feasibility pump) also with regard to running time or finding good solutions.

One of the original motivations to add the feasibility pump to DICOPT was to use cuts from the MIP projection problem (FP-OA^i) to warm-start the outer-approximation MIP (rMIP^i). As seen from Table 4, even though the solution quality does not decrease when disabling the initialization of (rMIP^i), one can observe that the additional time that is spent for running the feasibility pump pays off only if also the cuts from (FP-OA^i) are transferred to (rMIP^i).

Table 4: Results of running DICOPT with different settings.

setting	timeout	optimal	optimal w/ proof	good sol.	mean time
DICOPT w/ FP	31	58	48	67	137.9
DICOPT w/o FP	31	55	48	62	140.8
FP only	25	67	30	74	119.0
DICOPT w/ FP w/o OA init	31	59	49	68	170.5

5 Conclusions and perspectives

This paper has addressed the solution of convex MINLPs using the commercial solver DICOPT. Based on the work of Bonami, Cornuéjols, Lodi, and Margot [6], a modified iterative feasibility pump algorithm as a preprocessing for DICOPT has been proposed and implemented. As seen in the illustrative example, DICOPT in default settings has shown to perform poorly when many of the nonlinear subproblems are infeasible. Solving the illustrative example using DICOPT with the feasibility pump, better performance in solution time and solution quality could be achieved. As seen in the results from Section 4.2, the feasibility pump is not efficient in proving optimality, which validates the use of a stall limit as the criterion when to switch from the feasibility pump to the outer-approximation algorithm.

If the feasibility pump is used as primal heuristic only, the quality of the found solutions is improved, but the running time of DICOPT is increased considerably. Only when additionally the cuts from the MIP projection problem of the feasibility pump are used to initialize the MIP of the outer-approximation algorithm, improvements can be achieved with respect to DICOPT without feasibility pump in terms of both solution quality and running time.

Further work to improve the feasibility pump implementation in DICOPT is motivated by the following observations. Achterberg and Bethold [1] proposed a modification to the original algorithm that includes some information about the original objective function in the objective function of the feasibility pump problems to mitigate the issue of finding poor feasible solutions in terms of the original objective. Further, currently, the feasibility pump is only run at the beginning of DICOPT before the main loop of the outer-approximation algorithm. Executing it only once has been sufficient to find good feasible solutions for many instances in our test set. However, for some instances, it may be worth investigating a more extensive integration of the feasibility pump into DICOPT, e.g., allowing it to be used also when infeasible NLP subproblems are encountered in a similar manner as proposed by Bonami et al. [5]. Finally, the feasibility pump implementation should be generalized to nonconvex MINLP problems. Several authors have proposed such extensions [4, 9]. DICOPT itself already has heuristics to deal with nonconvex MINLPs, see Section 2.2, which could be carried over to the feasibility pump implementation.

Funding

The first and fourth authors would like to acknowledge financial support from the Center for Advanced Process Decision-making (CAPD). The second author was supported by the Research Campus MODAL *Mathematical Optimization and Data Analysis Laboratories* funded by the German Federal Ministry of Education and Research (BMBF Grant 05M14ZAM).

References

[1] Tobias Achterberg and Timo Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, 2007. doi:10.1016/j.disopt.2006.10.004.

[2] Egon Balas and Robert Jeroslow. Canonical cuts on the unit hypercube. *SIAM Journal on Applied Mathematics*, 23(1):61–69, 1972. ISSN 0036-1399. doi:10.1137/0123007.

[3] Livio Bertacco, Matteo Fischetti, and Andrea Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4(1):63–76, 2007. doi:10.1016/j.disopt.2006.10.001.

[4] Timo Berthold. *Heuristic algorithms in global MINLP solvers*. PhD thesis, TU Berlin, 2014.

[5] Pierre Bonami, Lorenz T. Biegler, Andrew R. Conn, Gérard Cornuéjols, Ignacio E. Grossmann, Carl D. Laird, Jon Lee, Andrea Lodi, François Margot, Nicolas Sawaya, and Andreas Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008. doi:10.1016/j.disopt.2006.10.011.

[6] Pierre Bonami, Gérard Cornuéjols, Andrea Lodi, and François Margot. A Feasibility Pump for mixed integer nonlinear programs. *Mathematical Programming*, 119(2):331–352, 2009. doi:10.1007/s10107-008-0212-2.

[7] Michael R. Bussieck, Steven P. Dirkse, and Stefan Vigerske. PAVER 2.0: an open source environment for automated performance analysis of benchmarking data. *Journal of Global Optimization*, 59(2-3):259–275, jul 2014. doi:10.1007/s10898-013-0131-5.

[8] Ignacio Castillo, Joakim Westerlund, Stefan Emet, and Tapio Westerlund. Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods. *Computers & Chemical Engineering*, 30(1):54–69, 2005. doi:10.1016/j.compchemeng.2005.07.012.

[9] Claudia D'Ambrosio, Antonio Frangioni, Leo Liberti, and Andrea Lodi. A storm of feasibility pumps for nonconvex MINLP. *Mathematical Programming*, 136(2):375–402, 2012. doi:10.1007/s10107-012-0608-x.

[10] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. doi:10.1007/s101070100263.

[11] Marco A. Duran and Ignacio E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3):307–339, 1986. doi:10.1007/BF02592064.

[12] Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005. doi:10.1007/s10107-004-0570-3.

[13] Roger Fletcher and Sven Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66(1-3):327–349, 1994. doi:10.1007/BF01581153.

[14] Björn Geißler, Antonio Morsi, Lars Schewe, and Martin Schmidt. Penalty alternating direction methods for mixed-integer optimization: A new view on feasibility pumps. *SIAM Journal on Optimization*, 27(3):1611–1636, 2017. doi:10.1137/16M1069687.

[15] Ignacio E. Grossmann and Zdravko Kravanja. Mixed-Integer Nonlinear Programming: A survey of algorithms and applications. In Lorenz T. Biegler, Thomas F. Coleman, Andrew R. Conn, and Fadil N. Santosa, editors, *Large-Scale Optimization with Applications: Part II: Optimal Design and Control*, pages 73–100. Springer New York, 1997. doi:10.1007/978-1-4612-1960-6_5.

[16] Ignacio E. Grossmann and Jon Lee. Cyberinfrastructure for mixed-integer nonlinear programming. *SIAG/OPT Views-and-News*, 22(1):8–12, 2011. URL `http://www.minlp.org`.

[17] Ignacio E Grossmann, Jagadisan Viswanathan, Aldo Vecchietti, Ramesh Raman, and Erwin Kalvelagen. GAMS/DICOPT: A Discrete Continuous Optimization Package, 2002.

[18] G. R. Kocis and I. E. Grossmann. Computational experience with DICOPT solving MINLP problems in process systems engineering. *Computers & Chemical Engineering*, 13(3):307–315, 1989. doi:10.1016/0098-1354(89)85008-2.

[19] Jon Lee and Sven Leyffer, editors. *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*. Springer, 2012. doi:10.1007/978-1-4614-1927-3.

[20] Russell D Meller and Kai-Yin Gau. The facility layout problem: Recent and emerging trends and perspectives. *Journal of Manufacturing Systems*, 15(5):351–366, 1996. doi:10.1016/0278-6125(96)84198-7.

[21] Francisco Trespalacios and Ignacio E. Grossmann. Review of mixed-integer nonlinear and generalized disjunctive programming methods. *Chemie Ingenieur Technik*, 86(7):991–1012, 2014. doi:10.1002/cite.201400037.

[22] Stefan Vigerske. MINLPLib 2. In L. G. Casado, I. García, and E. M. T. Hendrix, editors, *Proceedings of the XII global optimization workshop MAGO 2014*, pages 137–140, 2014.

[23] J. Viswanathan and I. E. Grossmann. A combined penalty function and outer-approximation method for MINLP optimization. *Computers & Chemical Engineering*, 14(7):769–782, 1990. doi:10.1016/0098-1354(90)87085-4.

A Linearization of integer cut

For a point $\bar{y} \in Y$, consider the integer cut

$$\|y - \bar{y}\|_1 \geq 1. \tag{2}$$

Recall that $Y = [y^L, y^U]$ with $y^L, y^U \in \mathbb{Z}^{n_y}$ (due to (A1)). A linear formulation of (2) is easily found if $\bar{y}_j \in \{y_j^L, y_j^U\}$ for every $j \in J := \{1, \ldots, n_y\}$, since the absolute difference $|y_j - \bar{y}_j|$ is reduced to $y_j - y_j^L$, if $\bar{y}_j = y_j^L$, and $y_j^U - y_j$ otherwise. Thus, for the specific case of binary variables only, i.e., $y_j^L = 0, y_j^U = 1, j \in J$, (2) simplifies to

$$\sum_{j \in J^L} y_j - \sum_{j \in J^U} (1 - y_j) \geq 1.$$

In the general case, we partition the set J into

$$\begin{aligned} J^L &= \{j \in J \ : \ \bar{y}_j = y_j^L\}, \\ J^U &= \{j \in J \ : \ \bar{y}_j = y_j^U\}, \\ J^M &= J \setminus (J^L \cup J^U). \end{aligned}$$

Using this set partition, the absolute difference of the variables to a given solution can be expressed as a sum of three terms. Thus, the integer cut (2) can be written as

$$\sum_{j \in J^L} (y_j - y_j^L) + \sum_{j \in J^U} (y_j^U - y_j) + \sum_{j \in J^M} |y_j - \bar{y}_j| \geq 1.$$

For every $j \in J^M$, we introduce a binary variable v_j, which determines whether the variable y_j is greater than or less than $\bar{y}_j$, and a positive continuous variable w_j to represent the value $|y_j - \bar{y}_j|$. This can be expressed using the following disjunctions:

$$\begin{bmatrix} v_j = 0 \\ y_j \leq \bar{y}_j \\ w_j = \bar{y}_j - y_j \end{bmatrix} \vee \begin{bmatrix} v_j = 1 \\ y_j \geq \bar{y}_j \\ w_j = y_j - \bar{y}_j \end{bmatrix}, \quad j \in J^M.$$

This disjunction can be reformulated into mixed-integer linear form, which yields the following reformulation of (2):

$$
\begin{aligned}
&\sum_{j \in J^L} (y_j - y_j^L) + \sum_{j \in J^U} (y_j^U - y_j) + \sum_{j \in J^M} w_j \geq 1, && \\
&-w_j \leq y_j - \bar{y}_j \leq w_j, && j \in J^M, \\
&w_j \leq y_j - \bar{y}_j + M_j^1(1 - v_j), && j \in J^M, \\
&w_j \leq \bar{y}_j - y_j + M_j^2 v_j, && j \in J^M, \\
&w_j \geq 0, && j \in J^M, \\
&v_j \in \{0, 1\}, && j \in J^M.
\end{aligned}
$$

To avoid weak relaxations, the big-M constants M_j^1 and M_j^2 should be chosen as small as possible and such that

$$
\begin{aligned}
&y_j - \bar{y}_j + M_j^1 \geq w_j = \bar{y}_j - y_j \quad \forall y_j \in [y_j^L, \bar{y}_j] && \text{(case } v_j = 0 \rightarrow y_j \leq \bar{y}_j\text{)}, \\
&\bar{y}_j - y_j + M_j^2 \geq w_j = y_j - \bar{y}_j \quad \forall y_j \in [\bar{y}_j, y_j^U] && \text{(case } v_j = 1 \rightarrow y_j \geq \bar{y}_j\text{)}.
\end{aligned}
$$

Thus, $M_j^1 = 2(\bar{y}_j - y_j^L)$ and $M_j^2 = 2(y_j^U - \bar{y}_j)$.

B Performance Profiles

Figure 2 shows performance profiles [10] comparing DICOPT with and without feasibility pump and the feasibility pump alone. Figure 3 shows a performance profile that illustrates the effect of disabling the initialization of (rMIP^i) with the cuts from (FP-OA^i).

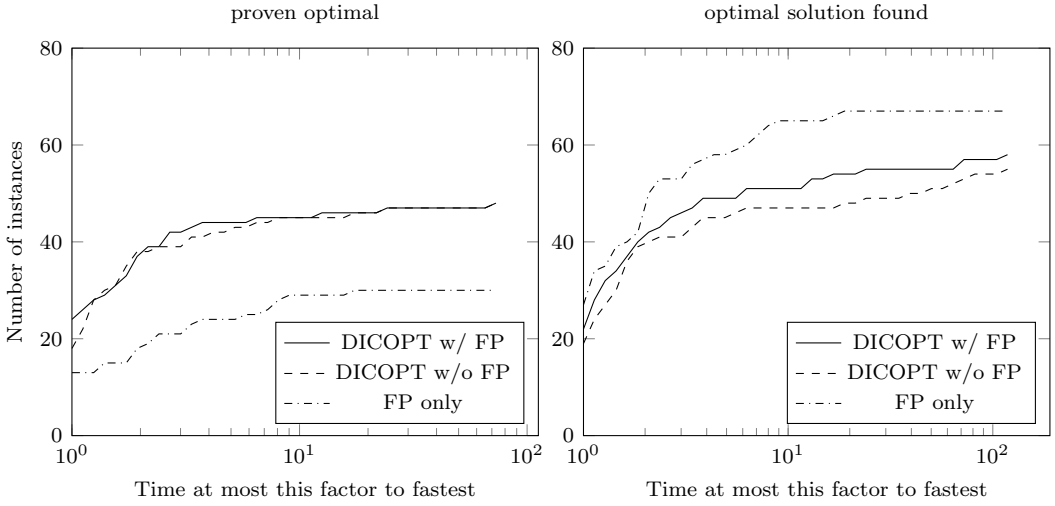

Figure 2: Performance profile showing the number of instances solved to proven optimality (left) and where an optimal solution has been found (right), respectively, with respect to solution time for various DICOPT settings.

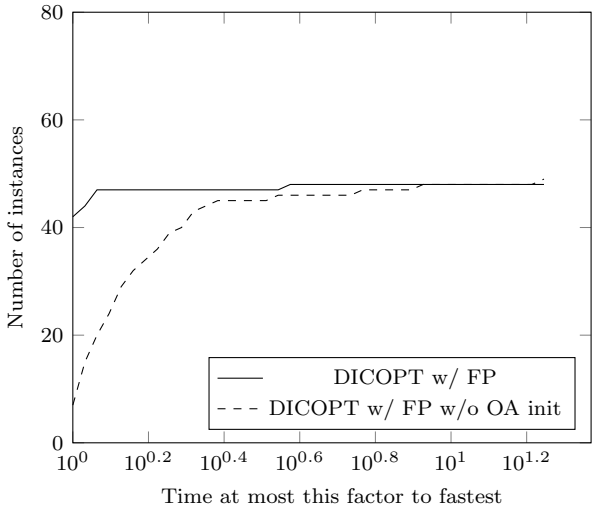

Figure 3: Performance profile showing the number of instances solved to proven optimality with respect to solving time, with and without the initialization of (rMIP^i) with the cuts from the feasibility pump.