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

September 8, 2017

Abstract

The solver DICOPT is based on an 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, there are certain problems that are difficult to solve with this algorithm. One of these problems is when the nonlinear constraints are so restrictive that the nonlinear subproblems produced 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 for DICOPT. Computational comparisons with previous versions of DICOPT and other MINLP solvers on a set of convex MINLPs demonstrate the effectiveness of the proposed algorithm in terms of solution quality and solving time.

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

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

1 Introduction

The capabilities of the algorithms designed to solve mathematical programming problems are continuously increasing. 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 terms (like discrete choices in superstructures or networks) with nonlinear terms (for example blending equations or concave cost functions) [13, 14, 17, 20]. The general form of an MINLP is

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

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}$ is the objective function and at least one of the constraints $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^m$ or the objective function itself is nonlinear. MINLP models are generally nonconvex due to the discrete nature of y and possible nonconvexity of f and g. Models in which f and g_i, $i = 1, \ldots, m$,

*Corresponding author. Email: grossmann@cmu.edu

are convex, are denoted as convex MINLP problems. We denote by (x^*, y^*) an optimal solution of the MINLP if it exists.

DICOPT (short for 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 NLP subproblem 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. 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, the stopping criterion is that the bound defined by the objective of the last MIP master problem exceeds the objective value of the best found solution [23].

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 is valid for convex MINLPs, this issue persists in some problems. This behavior yields 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] 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 a NLP. The MIP problem yields solutions that satisfy integrality requirements ($y \in \mathbb{Z}^{n_y}$) but may violate nonlinear constraints, while the NLP problems satisfy the constraints $g(x, y) \leq 0$ but may violate integrality requirements. In difference 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, it is expected that a solution be obtained that is feasible for both relaxations, and thus for the MINLP itself. The feasibility pump can also be used as a standalone solver for convex MINLP problems including a bound (cutoff-value) to the objective function, and iteratively applying the method. The bound is obtained by the objective value of the best known solution and a desired ϵ-improvement. This will result in finding an ϵ-global optimum of a convex MINLP [6]. The drawback of this algorithm is that it may require many iterations, since only an ϵ-improvement of the objective function is enforced at each iteration.

In this work, the feasibility pump is developed and applied in DICOPT before the outer-approximation method is activated. 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 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. The section also provides a brief introduction to hybrid algorithms that use parts of the two algorithms described there. Section 3 describes the algorithms proposed in this work. This algorithm uses the feasibility pump as initialization for DICOPT. An illustrative example and computational results are presented in Section 4.

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

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:

(A1) The set of constraints $g(x, y) \leq 0$ includes lower and upper bounds for every integer variable.

(A2) The constraint functions $g(x, y)$ and the objective function $f(x, y)$ are continuously differentiable and convex with respect to variable bounds.

(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,

$$\begin{aligned} \min_{x,y} \quad & f(x, y) \\ \text{s.t.} \quad & g(x, y) \leq 0 \\ & x \in \mathbb{R}^{n_x}, y \in \mathbb{R}^{n_y}. \end{aligned} \tag{rMINLP}$$

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 means of linearizing the nonlinear functions in $g(x, y)$ by first-order Taylor series approximations, 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:

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

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 also 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 $\hat{y}^i$:

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

Let $\bar{x}^i$ be a solution to (NLP^i), if feasible, and let $\bar{y}^i := \hat{y}^i$. 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

$$\begin{aligned} \min_{x,s} \quad & \sum_{j=1}^{m} s_j \\ \text{s.t.} \quad & g(x, \hat{y}^i) - s \leq 0 \\ & x \in \mathbb{R}^{n_x}, s \in \mathbb{R}^m_+. \end{aligned} \tag{NLP-feas^i}$$

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, 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 $\mathbb{R}^{n_x} \times \{\hat{y}^i\}$. Therefore, if only those iterations are included into C^i for which (NLP^i) is infeasible, then then 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 the optimal solution of a convex MINLP in a finite number of iterations [11].

Algorithm 1 Outer-approximation algorithm.

Set $Z^U = \infty$, $Z^L = -\infty$, $i = 0$ ▷ Initialization
Define gap tolerance $\epsilon \geq 0$
Solve (rMINLP) ▷ Solve initial relaxation
if (rMINLP) is infeasible **then**
    Set $Z^L = \infty$ ▷ (MINLP) is infeasible
else
    Let $(\bar{x}^0, \bar{y}^0)$ be an optimal solution of (rMINLP)
    Set $Z^L = f(\bar{x}^0, \bar{y}^0)$
    Set $L^0 = \{1, \ldots, m\}$, $C^0 = \emptyset$
    if $\bar{y}^0 \in \mathbb{Z}^{n_y}$ **then**
        Set $Z^U = f(\bar{x}^0, \bar{y}^0)$ and $\hat{y}^0 = \bar{y}^0$
while $Z^U - Z^L > \epsilon$ **do**
    Set $i = i + 1$
    Solve (MIP^i) ▷ Solve master problem
    if (MIP^i) is infeasible **then**
        Set $Z^L = \infty$ ▷ (MINLP) is infeasible
    else
        Let $(\hat{\alpha}^i, \hat{x}^i, \hat{y}^i)$ be an optimal solution of (MIP^i)
        Set $Z^L = \hat{\alpha}^i$
        Solve (NLP^i) ▷ Solve nonlinear subproblem
        if (NLP^i) is infeasible **then**
            Solve (NLP-feas^i)
            Let $(\bar{x}^i, \bar{s}^i)$ be an optimal solution of (NLP-feas^i)
            Set $C^{i+1} = C^i \cup \{i\}$
        else
            Let $\bar{x}^i$ be an optimal solution of (NLP^i)
            Set $C^{i+1} = C^i$
            Set $Z^U = \min(Z^U, f(\bar{x}^i, \hat{y}^i))$
        Set $L^i = \{j \in \{1, \ldots, m\} \,:\, g_j(\bar{x}^i, \hat{y}^i) \geq 0 \text{ and } g_j \text{ is nonlinear}\}$
$(\bar{x}^i, \hat{y}^i)$ is an optimal solution of (MINLP), if $Z^U < \infty$, otherwise (MINLP) is infeasible

2.2 Outer-approximation in DICOPT

Outer-approximation is the main algorithm behind the solver DICOPT [15, 16, 23], which has been developed in the late 1980's 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 specialized to these problem types.

As DICOPT is also intended as a heuristic for nonconvex MINLPs, the implementation of the outer-approximation algorithm deviates slightly from Algorithm 1. The main differences are:

- 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. However, for convex MINLP, Z^L and Z^U yield valid lower and upper bounds on the optimal value of (MINLP) and (NLP^i) is typically 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.

- 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 (MIP^i), but does not add the corresponding linearizations of nonlinear functions, i.e., $L^i = \emptyset$ if (NLP^i) is infeasible in Line 29 of Algorithm 1. This option is sufficient to avoid visiting the same solution point again and improves the success rate on nonconvex MINLPs (where linearizations may not yield supporting hyperplanes). However, it also yields 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`.

- Also 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 of 1000 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.

- Finally, DICOPT relaxes nonlinear equality constraints to inequalities and adds corresponding linearizations to (MIP^i). The dual multipliers in the solution of (NLP^i) are used to decide 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]. Subsequently, several authors have proposed extensions to nonconvex MINLPs [4, 9], 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 outer-approximation, 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:

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

where $L^k \subseteq \{1, \ldots, m\}$ is chosen as in the outer-approximation algorithm, see Section 2.1. 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:

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

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.

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):

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

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 improving solution.

The iterative feasibility pump algorithm is summarized in Algorithm 2. If (MINLP) is feasible, the algorithm finds a δ-optimal solution.

3 Proposed Algorithm

While the main focus of the outer-approximation algorithm is to find a best possible 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

Algorithm 2 Feasibility pump algorithm.

Set $Z^U = \infty$, $i = 0$ ▷ Initialization
Define zero tolerance $\epsilon \geq 0$
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 $L^0 = \{1, \ldots, m\}$
if $\bar{y}^0 \in \mathbb{Z}^{n_y}$ **then**
    Set $Z^U = f(\bar{x}^0, \bar{y}^0)$ ▷ Optimal solution found
else
    Set $i = 1$
    Solve (FP-OA^i) ▷ Solve feasibility OA problem
    while (FP-OA^i) is feasible **do**
        Let $(\hat{x}^i, \hat{y}^i)$ be an optimal 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 $\|\bar{y}^i - \hat{y}^i\| < \epsilon$ **then**
            Set $Z^U = f(\bar{x}^i, \bar{y}^i)$ ▷ New incumbent solution
        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
$(\bar{x}^i, \hat{y}^i)$ is an optimal solution of (MINLP), if $Z^U < \infty$, otherwise (MINLP) is infeasible

constraints. Therefore, the outer-approximation algorithm may be inefficient on problems where feasible solutions are difficult to find, while the (iterative) feasibility pump algorithm may take long to find a (proven) optimal solution on problems with many feasible points. To alleviate and explore the differences of these algorithms, hybrid algorithms have been designed, the first one being in [6]. In this variation of the outer-approximation algorithm, the feasibility pump algorithm is called when the NLP subproblem (NLP^i) is found to be infeasible. As the feasibility pump is expected to quickly find (improved) feasible solutions, a two minutes and five iterations limit was set for each call of the feasibility pump.

For DICOPT, we have 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 have slightly modified the feasibility pump algorithm as stated in Section 2.3 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 having several feasible solutions with the same values in the integer variables. To avoid repeated evaluation of the same values for y, we also add to (FP-OA^i) an integer cut for every fixed y for which (NLP^i) has been solved. Finally, we add the constraint $f(x, y) \leq Z^U - \hat{\delta}|Z^U|$ to (FP-NLP^i) in order to avoid non-improving solutions and replace an absolute cutoff δ by a relative cutoff $\hat{\delta}$ in (FP-OA^i).

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 and integer cuts that have been added to (FP-OA^i). Thus, if the resulting MIP relaxation (MIP^i)

is found to be infeasible by DICOPT, then this proves optimality of the solution found by the feasibility pump.

Algorithm 3 Proposed algorithm.

Set $Z^U = \infty$, $i = 0$ ▷ Initialization
Define zero tolerance $\epsilon \geq 0$
Define cutoff decrease $\hat{\delta} \geq 0$
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 $Z^U = f(\bar{x}^0, \bar{y}^0)$
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 **do**
    Let $(\hat{x}^i, \hat{y}^i)$ be an optimal 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 $\|\bar{y}^i - \hat{y}^i\| < \epsilon$ **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 $C^{i+1} = C^i \cup \{i\}$
    else
        Set $C^{i+1} = C^i$
    Set $L^i = \{j \in \{1, \ldots, m\} \ : \ g_j(\bar{x}^i, \bar{y}^i) \geq 0 \text{ and } g_j \text{ is nonlinear}\}$
    Set $i = i + 1$
    Solve (FP-OA^i) ▷ Solve feasibility OA problem
Solve (MINLP) using DICOPT, initialized with incumbent solution $(\bar{x}^i, \hat{y}^i)$, if $Z^U < \infty$, and linearizations given by L^i and the integer cuts given by C^i in the relaxation (MIP^i).

A general outline of the proposed algorithm is given in Algorithm 3. It has been implemented as part of the solver DICOPT and is available in GAMS since version 24.5. To enable and adjust the algorithm, a number of options have been added, which are summarized in Table 1. As DICOPT is often used for nonconvex MINLPs, see also the discussion in Section 2.2, the default values for options `convex` and `feaspump` are 0.

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 2 (rev. 342, as of 14.2.2017)[2] [21]. First, we selected all instances that are marked as convex, have at least one binary or general integer variable, no semicontinuous or semiinteger variables, and no special-ordered-sets. This gives a set of 326 instances. Second, we run DICOPT with `convex` option enabled and feasibility pump disabled and removed all instances for which DICOPT terminated in less than one second. In this remaining set, there was still strong dominance of some subsets of instances that were clearly derived from the

[2] `http://www.gamsworld.org/minlp/minlplib2/html/index.html`

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 `step` 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 in the feasibility pump	∞
`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	5
`fp_cutoffdecr`	Relative decrement of cutoff value for the objective variable ($\hat{\delta}$)	0.1
`fp_acttol`	Tolerance on when a constraint is found active	10^{-6}
`fp_projzerotol`	Tolerance on when to consider the optimal value of (FP-NLP^i) as zero	10^{-6}
`fp_transfercuts`	Whether to transfer cuts from the feasibility pump MIP to the DICOPT MIP (all except from the iteration in which (FP-OA^i) became infeasible)	1

same model (strong similarity in name). Therefore, we reduced these subsets to the four largest instances. This leaves a final set of 68 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. Appendix B provides this list of instances.

For all the experiments, we have set a time limit of 1800 seconds and set the GAMS gap tolerance (`optcr`) to 0. We used PAVER 2 [7] to help in the comparisons of various test runs and GAMS/Examiner2 to check primal feasibility of the solutions returned by the solvers. All runs were performed on a cluster of Dell PowerEdge M630 blades with 128 GB RAM, Intel Xeon E5-2690v4 CPUs running at 2.60 GHz, and Linux 4.4.0 (64bit). GAMS 24.8.4 was used for all experiments. DICOPT 2 uses CPLEX 12.7.1.0 for solving MIPs, and CONOPT 3.17C for solving NLPs.

4.1 Illustrative example

Before evaluating the performance of the new feasibility pump on the complete testset, we discuss its behavior for a single instance. This instance's application is the block layout design problem with unequal areas. The original problem was proposed by Meller et al. [18] and was reformulated by Castillo et al. [8] as a convex MINLP. This sort 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, 14 of them nonlinear, specifically signomial, and 114 variables, 42 of them binary. This instance can be found in MINLPLib 2 under the name `o7_2`[3]. The original 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 (MIP^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 criteria

[3] `http://www.gamsworld.org/minlp/minlplib2/html/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	6	8	2	2
feasible solutions found	0	1	5	5
FP iterations	0	0	10	10
FP time [s]	0	0	99.45	98.49
infeasible NLP	5	6	0	0
time to optimal sol. [s]	–	547.77	184.42	182.85
solution time [s]	55.41*	839.97	427.25	425.26
final objective value	–	116.94	116.94	116.94

*Time when the solver terminated due to an error.

for all the different options was the crossover between the objective values of the master MIP problem and the NLP subproblem. The default setting for option `infeasder` requires that if the nonlinear subproblem (NLP^i) is infeasible, only a corresponding integer cut is added to (MIP^i). This approach, although rigourous for convex and non-convex MINLPs, is not very efficient, particularly for this sort 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 linearization cuts if the nonlinear subproblem is infeasible, using the solution of (NLP-feas^i) as reference point. This can be enabled by using the option `infeasder`. 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. In fact, the solver terminated with an error when the optimal value of the master MIP problem (MIP^i) that was solved unexpectedly decreased (we are minimizing) after adding an integer cut. This issue was encountered after 55 seconds of execution and was probably caused by numerical errors in the MIP subsolver[4]. During this time the solver performed 6 major iterations. That is, at each iteration it solved a master MIP problem (MIP^i) and an NLP subproblem (NLP^i). All NLP subproblems were infeasible.

Enabling the `infeasder` option, the problem could be solved in 840 seconds. During this time, 5 out of the 7 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 major iteration to prove its optimality. Notice that even after finding an optimal solution, the next major iteration resulted in an infeasible NLP subproblem again, something that did not happen while using the feasibility pump. The use of the feasibility pump allowed the solver to find 4 feasible solutions in the first ≈ 100 seconds. After that, a single major iteration was required to find an optimal solution, which required 183 and 184 seconds with and without the `infeasder` option, respectively. 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 66% reduction in the time needed to find an optimal solution to the problem, and a 49% 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 1363 seconds, with BONMIN 646 seconds, and with SCIP 946 seconds (see Table 10).

[4]With GAMS 24.8.3, using CPLEX 12.7.0.0, this failure did not occur. Instead, DICOPT terminated after 30 minutes without finding any feasible solution.

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

setting	timeout	optimal	good sol.	feasible
default	0	5	38	56
stall10	2	5	52	63
findopt	35	45	55	64

4.2 Feasibility pump alone

In the following, we consider the full test set of 68 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 decrement of $\hat{\delta} = 0.1$. In setting "stall10", we increased the stall limit to 10. The setting "findopt" targets on finding optimal solutions of the MINLP; that is, we disable the stall limit and set the cutoff decrement to 0.

Figure 1 plots the primal gap of all runs where a feasible solution has been found and Table 3 summarizes the results. Detailed results are given in Table 7. 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 5 instances, good solutions ($< 10\%$ primal gap) for another 33 instances, and some feasible solutions ($\geq 10\%$ primal gap) for another 18 instances. Increasing the stall limit helps on many of the instances where previously no or only bad solutions were found. On instances where good solutions were already found in default settings, increasing the stall limit has little effect, likely because the cutoff decrement $\hat{\delta}$ cuts off solutions that are only slightly better or optimal. 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. However, there are also some instances where the feasibility pump terminates with a solution worse than in default settings. This is because by disabling the cutoff decrement, the feasibility pump searches less aggressively for improving solutions.

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 hit. In the "findopt" setting, however, 35 instances terminated when the time limit of 1800 seconds was hit. Thus, the feasibility pump is not suited to prove optimality of found solutions. This justifies the choice of the stall limit as stopping criterion.

4.3 DICOPT with feasibility pump

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

Table 4 summarizes on how many instances out of 68 each setting yield an incorrect result (claiming optimality for a non-optimal solution or returning a solution that is not feasible), on how many instances it hit the time limit, for how many instances an optimal solution was found, for how many instances optimality was proven, and for how many instances a good solution was found (primal gap $\leq 10\%$). Detailed results are given in Tables 8 and 9. The numbers show that adding the feasibility pump to DICOPT leads to finding an optimal solution to two more instances than

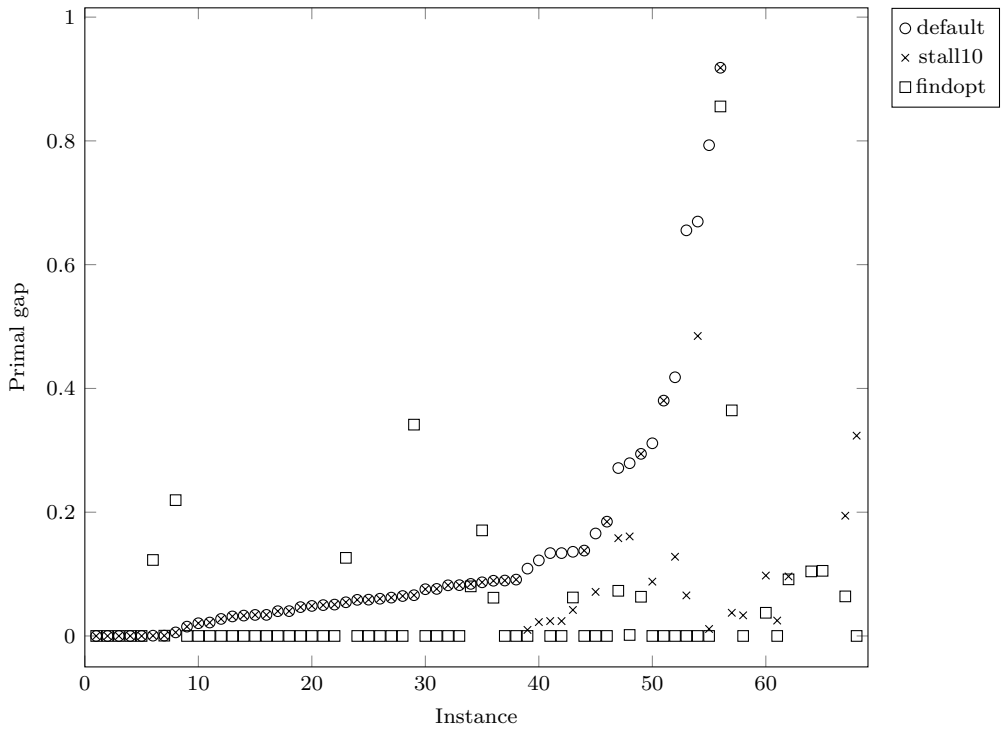

Figure 1: Primal gap of solutions found by feasibility pump (with different settings) for all instances in test set, sorted by primal gap of "default" setting.

Table 4: Results of running DICOPT with different settings.

setting	fail	timeout	optimal	optimal w/ proof	good sol.
DICOPT w/ FP	7	21	42	40	49
DICOPT w/o FP	10	21	40	37	44
FP only	1	35	45	32	55
DICOPT w/ FP w/o OA init	10	21	39	37	44

before and proving optimality for three instances more than before. Running the feasibility pump alone increases the number of found optimal solutions by another 3 and found good solutions by 6, but considerably decreases the number of instances on which optimality is proven. Figure 2 shows performance profiles [10] comparing DICOPT with and without feasibility pump and the feasibility pump alone. We see that enabling the feasibility pump leads to a small performance loss with respect to solution time, but improvements in finding optimal solutions and proving optimality. To conclude, using DICOPT with feasibility pump shows to be a good compromise between finding good or optimal solutions and proving optimality.

Figure 3 compares the runs of DICOPT with enabled or disabled initialization of the outer-approximation MIP (MIP^i) by cuts from the feasibility pump. Even though for the same number of instances optimality is proved, the computing time is considerably decreased by reusing the cuts from the feasibility pump.

4.4 Comparison with other solvers

In the following, we compare the performance of DICOPT with some of the other solvers for convex MINLP available in GAMS. We have run AlphaECP 2.10.06 [24] with option `ECPmaster` set to 2 and `TOLepsg` set to 10^{-6}. AlphaECP uses CPLEX 12.7.1.0 for MIP solves, and CONOPT 3.17C

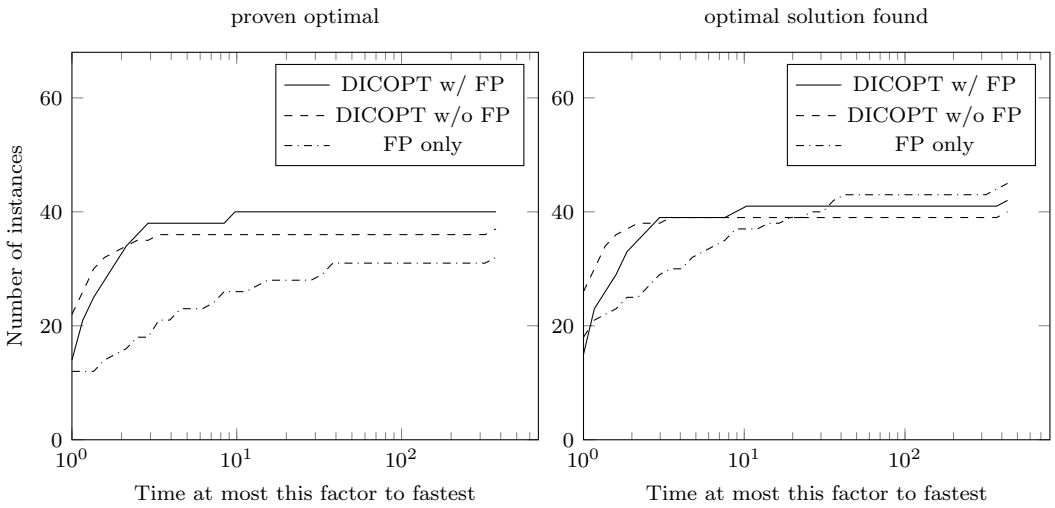

Figure 2: Performance profile showing the number of instances solved to proven optimality (left) and where an optimal solution has been found (right), respectively, with respect to solution time for various DICOPT settings.

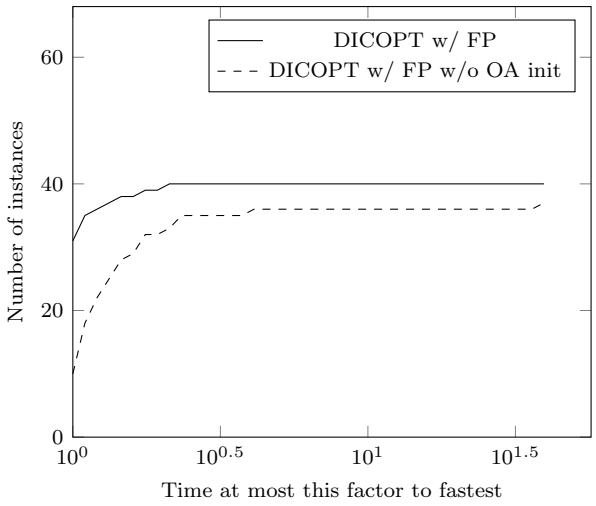

Figure 3: Performance profile showing the number of instances solved to proven optimality with respect to solving time, with and without the initialization of (MIP^i) with the cuts from the feasibility pump.

Table 5: Results from running various solvers.

setting	fail	timeout	optimal	optimal w/ proof	good sol.
AlphaECP	2	26	50	40	58
BONMIN	9	24	38	34	53
DICOPT w/o FP	10	21	40	37	44
DICOPT w/ FP	7	21	42	40	49
SCIP	2	22	55	44	60

for NLP solves. It implements a variant of the outer-approximation algorithm that requires less solutions of NLPs. Further, we have run the B-Hyb algorithm of BONMIN 1.8 [5], which is variant of the branch-and-cut algorithm by Quesada and Grossmann [19]. BONMIN uses Ipopt 3.12 (with MA27 as linear solvers) for NLP solves and CPLEX 12.7.1.0 for MIP solves. Finally, we have run SCIP 3.2 [22] with option `constraints/nonlinear/assumeconvex` enabled. This is a branch-and-cut algorithm based solely on LP relaxations. SCIP uses CPLEX 12.7.1.0 for LP solves and Ipopt 3.12 (with MA27 as linear solver) for NLP solves.

Table 5 reports on the number of instances each solver failed, run into the time limit, find an optimal solution, prove optimality, and found a good solution (primal gap $\leq 10\%$). Detailed results are given in Table 10. We see that on this testset, SCIP solves most instances, followed by AlphaECP and DICOPT with feasibility pump. Further, especially AlphaECP and SCIP succeed in finding good or optimal solutions to many instances. DICOPT without feasibility pump finds good solutions on considerably fewer instances than other solvers. Adding the feasibility pump reduces the distance, but having only one primal heuristic does not seem sufficient.

Figure 4 shows a profile that compares the solvers performance with respect to proving optimality. None of the solvers is fastest on a majority of instances, but SCIP is most efficient, followed by the two variants of DICOPT.

5 Conclusions and perspectives

This paper has addressed the solution of convex MINLPs using the commercial solver DICOPT. 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 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. A key issue to determine was when to stop the feasibility pump and start using DICOPT. As seen in the results from Section 4.2, the feasibility pump is not efficient in proving optimality of feasible solutions found, which validates the use of a stall limit as stopping criterion.

As seen in Figure 2, the use of the feasibility pump allows DICOPT to find better solutions to convex MINLP problems. Taking into consideration that another algorithm is run before passing the problem to DICOPT, the performance loss is marginal while the improvement in the solution quality is considerable. The hypothesis that the outer-approximation cuts generated while using the feasibility pump were useful for DICOPT was confirmed, as seen in Figure 3. The proposed algorithm has shown better performance than the original DICOPT without feasibility pump regarding solution quality, and has similar performance regarding performance and stability. The proposed implementation has been compared with the MINLP solvers AlphaECP, BONMIN, and SCIP. The latter showed the best overall performance.

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. It may be worth

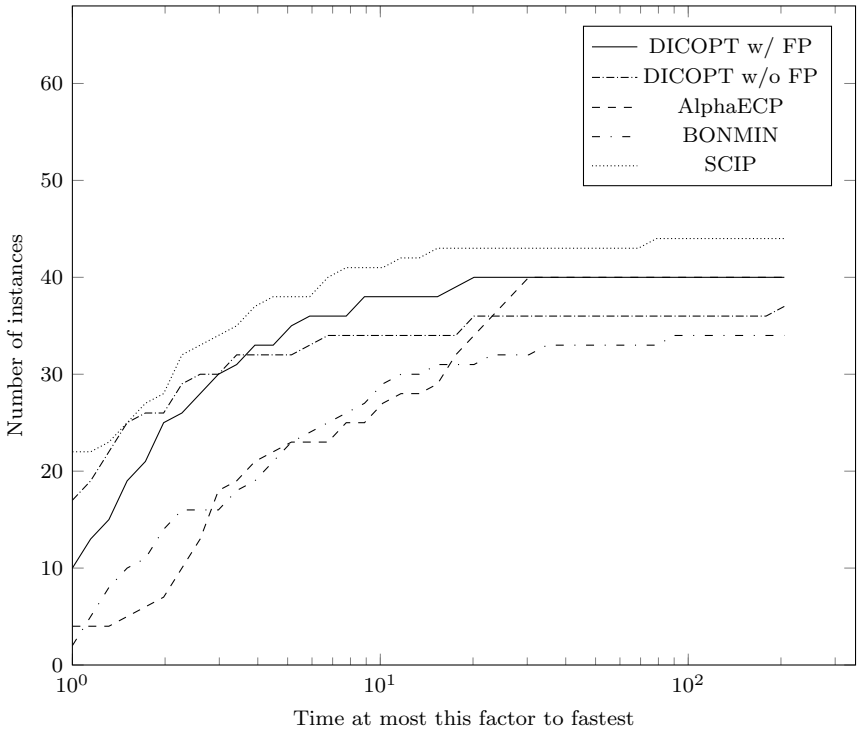

Figure 4: Performance profile showing the number of instances solved to proven optimality with respect to solving time for different solvers.

investigating a more extensive integration of the feasibility pump into DICOPT, e.g., allowing it to be used also when infeasible NLP suproblems 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 G. Jeroslow. Strengthening cuts for mixed integer programs. *European Journal of Operational Research*, 4(4):224–234, 1980. doi:10.1016/0377-2217(80)90106-X.

[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, dec 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] 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.

[14] 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`.

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

[16] 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.

[17] 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.

[18] 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.

[19] Ignacio Quesada and Ignacio E. Grossmann. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Computers & Chemical Engineering*, 16(10-11): 937–947, 1992. doi:10.1016/0098-1354(92)80028-8.

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

[21] 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.

[22] Stefan Vigerske and Ambros Gleixner. SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. ZIB Report 16-24, Zuse Institute Berlin, 2016. urn:nbn:de:0297-zib-59377.

[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.

[24] Tapio Westerlund and Ray Pörn. Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optimization and Engineering*, 3(3):253–280, 2002. doi:10.1023/A:1021091110342.

A Linearization of integer cut

For given bounds $y^L, y^U \in \mathbb{Z}^{n_y}$, $y^L \leq y^U$, on the integer variables y and a point $\bar{y} \in \mathbb{Z}^{n_y}$, $y^L \leq \bar{y} \leq y^U$, consider the integer cut

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

A linear formulation of (1) is easily found if $\bar{y}_j \in \{y^L_j, y^U_j\}$ for every $j \in J := \{1, \ldots, n_y\}$, since the absolute difference $|y_j - \bar{y}_j|$ gets reduced to $y_j - y^L_j$, if $\bar{y}_j = y^L_j$, and $y^U_j - y_j$ otherwise. Thus, for the specific case of binary variables only, i.e., $y^L_j = 0, y^U_j = 1, j \in J$, an (1) 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

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

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 (1) can be written as

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

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 disjunction:

$$\begin{bmatrix} v_j = 0 \\ y_j \leq \bar{y}_j \\ w_j = \bar{y}_j - y_j \end{bmatrix} \vee \begin{bmatrix} v_j = 1 \\ y_j \geq \bar{y}_j \\ w_j = y_j - \bar{y}_j \end{bmatrix},$$

This disjunction can be reformulated into a mixed-integer linear form, which yields the following reformulation of (1):

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

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

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

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

B Test Set

Table 6 details the selected set of instances. For each instance, we show the number of variables $(n_x + n_y)$, the number of binary variables (n_y^{bin}), the number of general integer variables (n_y^{int}), the number of constraint functions (m, excluding bound constraints), and the number of nonlinear functions in objective and constraints (m_{nl}).

C Detailed Results

In the following, we present detailed results for the computations from Section 4 above. Tables 7, 8, 9, and 10 report detailed data about the comparison of the feasibility pump settings, the runs of DICOPT with and without feasibility pump, the evaluation of initializing (MIP^i), and the comparison with other solvers, respectively. For each instance, we provide the primal gap and gap at termination and the solvers running time. For the feasibility pump comparison in Table 7, the gap is omitted, as it is not available.

Table 6: List of instances in convex MINLP test set.

Instance	$n_x + n_y$	n_y^{bin}	n_y^{int}	m	m_{nl}
batch	46	24	0	73	2
batchs101006m	278	129	0	1019	2
batchs121208m	406	203	0	1511	2
batchs151208m	445	203	0	1781	2
batchs201210m	558	251	0	2327	2
clay0205m	80	50	0	135	40
clay0303m	33	21	0	66	36
clay0304m	56	36	0	106	48
clay0305m	85	55	0	155	60
du-opt	20	0	13	9	1
flay05h	382	40	0	465	5
flay05m	62	40	0	65	5
flay06h	566	60	0	693	6
flay06m	86	60	0	93	6
fo7	114	42	0	211	14
fo7_2	114	42	0	211	14
fo8	146	56	0	273	16
fo9	182	72	0	343	18
fo9_ar2_1	180	0	72	435	18
fo9_ar3_1	180	0	72	435	18
fo9_ar4_1	180	0	72	435	18
fo9_ar5_1	180	0	72	435	18
gams01	145	110	0	1268	111
ibs2	3010	1500	0	1821	10
m7_ar2_1	112	0	42	269	14
m7_ar3_1	112	0	42	269	14
m7_ar5_1	112	0	42	269	14
netmod_dol1	1998	462	0	3137	1
netmod_dol2	1998	462	0	3080	1
netmod_kar1	456	136	0	666	1
netmod_kar2	456	136	0	666	1
no7_ar2_1	112	0	42	269	14
no7_ar3_1	112	0	42	269	14
no7_ar4_1	112	0	42	269	14
no7_ar5_1	112	0	42	269	14
o7	114	42	0	211	14
o7_2	114	42	0	211	14
o7_ar25_1	112	0	42	269	14
o7_ar5_1	112	0	42	269	14
o8_ar4_1	144	0	56	347	16
o9_ar4_1	180	0	72	435	18
portfol_classical050_1	150	50	0	103	1
portfol_classical200_2	600	200	0	403	1
rsyn0830m04h	2344	496	0	4236	80
rsyn0830m04m	1240	496	0	3192	80
rsyn0840m04h	2720	576	0	4980	112
rsyn0840m04m	1440	576	0	3728	112
slay09h	810	144	0	1044	1
slay09m	234	144	0	324	1
slay10h	1010	180	0	1305	1
slay10m	290	180	0	405	1
squfl020-150	3020	20	0	3150	1
squfl030-100	3030	30	0	3100	1
squfl030-150	4530	30	0	4650	1
squfl040-080	3240	40	0	3280	1
sssd18-08	200	168	0	82	24
sssd20-08	216	184	0	84	24
sssd22-08	232	200	0	86	24
sssd25-08	256	224	0	89	24
stockcycle	480	432	0	97	1
syn40m03h	1146	240	0	1998	84
syn40m04h	1528	320	0	2904	112
tls12	812	656	12	384	12
tls5	161	131	5	90	5
tls6	215	173	6	120	6
tls7	345	289	7	154	7
watercontamination0303	107222	14	0	108217	1
watercontamination0303r	384	14	0	556	1

Table 7: Detailed solving data for the comparison from Section 4.2.

Instance	default		findopt		stall10	
	prim.gap	time	prim.gap	time	prim.gap	time
batch	6.5	0.2	0.0	0.3	6.5	0.2
batchs101006m	5.8	1.3	0.0	6.1	5.8	1.2
batchs121208m	9.1	2.4	0.0	16.1	9.1	2.4
batchs151208m	3.3	3.1	0.0	19.2	3.3	3.2
batchs201210m	9.0	0.8	0.0	34.2	9.0	0.7
clay0205m	65.5	0.4	0.0	7.5	6.6	1.5
clay0303m	∞	0.2	0.0	1.0	3.3	0.6
clay0304m	∞	0.2	0.0	4.1	32.4	0.9
clay0305m	79.3	0.4	0.0	10.5	1.1	1.1
du-opt	1.5	0.3	0.0	50.9	1.5	0.3
flay05h	5.0	1.9	0.0	458.9	5.0	25.7
flay05m	0.0	0.3	0.0	358.5	0.0	2.6
flay06h	3.2	49.3	0.0	1800.1	3.2	397.0
flay06m	0.0	0.6	0.0	1800.0	0.0	43.8
fo7	5.1	10.1	0.0	67.9	5.1	10.0
fo7_2	0.0	8.9	0.0	23.3	0.0	8.9
fo8	31.1	21.5	0.0	1800.1	8.8	52.3
fo9	∞	1.7	36.5	1800.2	3.7	75.0
fo9_ar2_1	0.6	75.1	21.9	1800.4	0.6	75.5
fo9_ar3_1	6.6	17.7	34.1	1800.4	6.6	17.6
fo9_ar4_1	∞	1.7	3.7	1800.1	9.8	18.8
fo9_ar5_1	5.4	83.0	12.6	1800.8	5.4	83.5
gams01	4.9	7.6	0.0	1800.5	4.9	22.1
ibs2	∞	42.1	∞	1811.6	∞	46.5
m7_ar2_1	6.2	4.8	0.0	11.4	6.2	4.8
m7_ar3_1	0.0	8.4	0.0	1800.7	0.0	8.4
m7_ar5_1	0.0	1.3	0.0	510.9	0.0	1.3
netmod_dol1	∞	19.0	0.0	1800.1	2.5	113.1
netmod_dol2	6.0	28.2	0.0	162.9	6.0	53.4
netmod_kar1	13.4	3.7	0.0	47.5	2.4	8.4
netmod_kar2	13.4	3.7	0.0	47.5	2.4	8.4
no7_ar2_1	0.0	20.1	12.3	1800.3	0.0	20.4
no7_ar3_1	0.1	15.9	0.0	1800.3	0.1	16.0
no7_ar4_1	8.7	149.1	17.1	1800.5	8.7	151.6
no7_ar5_1	8.4	176.5	8.0	1800.4	8.4	176.5
o7	12.2	177.9	–	–	2.2	459.6
o7_2	3.4	98.6	0.0	1229.6	3.4	98.7
o7_ar25_1	4.0	151.6	0.0	1800.2	4.0	151.5
o7_ar5_1	8.9	183.3	6.2	1800.4	8.9	183.8
o8_ar4_1	13.6	644.7	6.2	1800.3	4.2	1710.0
o9_ar4_1	∞	1.6	9.2	1800.4	9.6	1800.4
portfol_classical050_1	7.6	1.3	0.0	1800.0	7.6	1.8
portfol_classical200_2	∞	3.2	6.4	1800.2	19.4	10.9
rsyn0830m04h	2.2	10.4	0.0	31.0	2.2	10.3
rsyn0830m04m	7.6	81.1	0.0	367.5	7.6	81.2
rsyn0840m04h	2.8	11.1	0.0	55.2	2.8	11.4
rsyn0840m04m	5.9	25.8	0.0	1090.8	5.9	25.8
slay09h	16.6	11.3	0.0	1800.2	7.1	14.1
slay09m	10.9	2.7	0.0	847.8	1.0	3.4
slay10h	41.8	4.9	0.0	1800.5	12.8	27.7
slay10m	13.8	2.1	0.0	1450.5	13.8	2.8
squfl020-150	18.5	15.2	0.0	1800.8	18.5	23.8
squfl030-100	29.4	16.1	6.3	1800.5	29.4	24.0
squfl030-150	27.1	52.0	7.3	1801.4	15.8	88.9
squfl040-080	38.0	13.2	0.0	1800.6	38.0	20.2
sssd18-08	3.4	1.4	0.0	1800.8	3.4	1.4
sssd20-08	4.0	27.7	0.0	508.2	4.0	27.6
sssd22-08	4.7	100.5	0.0	244.3	4.7	99.8
sssd25-08	2.1	2.6	0.0	530.4	2.1	2.5
stockcycle	27.9	3.4	0.2	1800.1	16.1	5.3
syn40m03h	8.2	3.6	0.0	12.4	8.2	3.7
syn40m04h	8.2	2.3	0.0	14.3	8.2	2.3
tls12	∞	251.5	∞	1800.4	∞	1800.4
tls5	∞	2.0	10.4	1800.1	∞	9.3
tls6	∞	12.0	10.5	1800.1	∞	51.8
tls7	∞	14.4	∞	1800.1	∞	35.2
watercontamination0303	67.0	137.8	0.0	421.9	48.5	271.3
watercontamination0303r	91.8	3.0	85.6	1800.4	91.8	3.6

Table 8: Detailed solving data for the comparison of DICOPT with and without feasibility pump from Section 4.3.

Instance	DICOPT w/ FP			DICOPT w/o FP			FP only		
	p.gap	gap	time	p.gap	gap	time	p.gap	gap	time
batch	0.0	0.0	0.3	0.0	0.0	0.1	0.0	0.0	0.3
batchs101006m	0.0	0.0	3.4	0.0	0.0	4.7	0.0	0.0	6.1
batchs121208m	0.0	0.0	5.7	0.0	0.0	3.9	0.0	0.0	16.1
batchs151208m	0.0	0.0	10.4	0.0	0.0	8.3	0.0	0.0	19.2
batchs201210m	0.0	0.0	8.2	0.0	0.0	7.9	0.0	0.0	34.2
clay0205m	–	–	–	–	–	–	0.0	0.0	7.5
clay0303m	0.0	0.0	9.5	0.0	0.0	374.2	0.0	0.0	1.0
clay0304m	0.0	68.6	1800.1	31.9	85.9	1800.0	0.0	0.0	4.1
clay0305m	–	–	–	–	–	–	0.0	0.0	10.5
du-opt	0.0	0.0	7.5	0.0	0.0	9.0	0.0	0.0	50.9
flay05h	0.0	0.0	608.4	0.0	0.0	520.7	0.0	0.0	458.9
flay05m	0.0	0.0	192.8	0.0	0.0	212.2	0.0	0.0	358.5
flay06h	0.0	2.8	1800.0	0.0	2.1	1800.0	0.0	∞	1800.1
flay06m	0.0	1.0	1800.0	0.0	0.8	1800.0	0.0	∞	1800.0
fo7	0.0	0.0	50.0	0.0	0.0	51.2	0.0	0.0	67.9
fo7_2	0.0	0.0	14.6	0.0	0.0	7.9	0.0	0.0	23.3
fo8	0.0	0.0	110.8	0.0	0.0	98.8	0.0	∞	1800.1
fo9	0.0	0.0	356.2	0.0	0.0	209.3	36.5	∞	1800.2
fo9_ar2_1	0.0	0.0	958.3	0.0	0.0	724.9	21.9	∞	1800.4
fo9_ar3_1	0.0	0.0	38.7	0.0	0.0	48.6	34.1	∞	1800.4
fo9_ar4_1	0.0	0.0	47.4	0.0	0.0	69.7	3.7	∞	1800.1
fo9_ar5_1	0.0	0.0	175.7	0.0	0.0	584.2	12.6	∞	1800.8
gams01	4.9	∞	1800.0	∞	∞	1800.0	0.0	∞	1800.5
ibs2	77.1	77.2	1800.0	73.2	73.2	1800.0	∞	∞	1811.6
m7_ar2_1	0.0	0.0	6.7	0.0	0.0	4.6	0.0	0.0	11.4
m7_ar3_1	0.0	0.0	9.5	0.0	0.0	4.8	0.0	∞	1800.7
m7_ar5_1	0.0	0.0	2.9	0.0	0.0	1.5	0.0	0.0	510.9
netmod_dol1	8.5	21.5	1800.0	8.5	21.5	1800.0	0.0	∞	1800.1
netmod_dol2	0.0	0.0	167.0	–	–	–	0.0	0.0	162.9
netmod_kar1	0.0	0.0	53.2	0.0	0.0	60.5	0.0	0.0	47.5
netmod_kar2	0.0	0.0	52.6	0.0	0.0	60.4	0.0	0.0	47.5
no7_ar2_1	–	–	–	–	–	–	12.3	∞	1800.3
no7_ar3_1	–	–	–	–	–	–	0.0	∞	1800.3
no7_ar4_1	0.0	0.0	215.5	0.0	0.0	198.8	17.1	∞	1800.5
no7_ar5_1	0.0	0.0	231.7	0.0	0.0	140.5	8.0	∞	1800.4
o7	0.0	0.5	1800.0	0.0	0.4	1800.0	–	–	–
o7_2	0.0	0.0	420.6	0.0	0.0	834.3	0.0	0.0	1229.6
o7_ar25_1	0.0	0.0	526.1	0.0	0.0	345.7	0.0	∞	1800.2
o7_ar5_1	0.0	0.0	384.0	0.0	0.0	636.1	6.2	∞	1800.4
o8_ar4_1	13.6	∞	1800.0	∞	∞	1800.0	6.2	∞	1800.3
o9_ar4_1	∞	∞	1800.0	∞	∞	1800.0	9.2	∞	1800.4
portfol_classical050_1	0.5	3.4	1800.0	0.2	3.3	1800.0	0.0	∞	1800.0
portfol_classical200_2	8.6	19.3	1800.0	7.1	17.9	1800.1	6.4	∞	1800.2
rsyn0830m04h	0.0	0.0	11.3	0.0	0.0	4.0	0.0	0.0	31.0
rsyn0830m04m	0.0	0.0	89.0	0.0	0.0	10.5	0.0	0.0	367.5
rsyn0840m04h	0.0	0.0	11.5	0.0	0.0	4.1	0.0	0.0	55.2
rsyn0840m04m	0.0	0.0	29.7	–	–	–	0.0	0.0	1090.8
slay09h	0.0	0.0	157.3	0.0	0.0	69.3	0.0	∞	1800.2
slay09m	0.0	0.0	27.3	0.0	0.0	27.9	0.0	0.0	847.8
slay10h	0.0	0.7	1800.0	0.0	0.6	1800.0	0.0	∞	1800.5
slay10m	0.0	0.0	1283.8	0.0	0.0	961.8	0.0	0.0	1450.5
squfl020-150	18.5	64.8	1800.0	20.9	65.8	1800.0	0.0	∞	1800.8
squfl030-100	22.7	70.5	1800.0	22.7	70.5	1800.0	6.3	∞	1800.5
squfl030-150	27.1	72.5	1800.1	35.1	75.6	1800.1	7.3	∞	1801.4
squfl040-080	26.1	71.8	1800.0	17.7	68.6	1800.0	0.0	∞	1800.6
sssd18-08	0.0	0.0	297.2	–	–	–	0.0	∞	1800.8
sssd20-08	–	–	–	–	–	–	0.0	0.0	508.2
sssd22-08	–	–	–	–	–	–	0.0	0.0	244.3
sssd25-08	0.0	0.0	40.5	0.0	0.0	91.7	0.0	0.0	530.4
stockcycle	15.6	17.1	1800.0	42.6	43.6	1800.0	0.2	∞	1800.1
syn40m03h	0.0	0.0	3.8	0.0	0.0	1.6	0.0	0.0	12.4
syn40m04h	–	–	–	–	–	–	0.0	0.0	14.3
tls12	∞	∞	1802.2	∞	∞	1815.0	∞	∞	1800.4
tls5	∞	∞	1800.0	∞	∞	1800.0	10.4	∞	1800.1
tls6	∞	∞	1800.0	∞	∞	1800.0	10.5	∞	1800.1
tls7	∞	∞	1801.0	∞	∞	1800.0	∞	∞	1800.1
watercontamination0303	0.0	0.0	224.5	0.0	0.0	126.9	0.0	0.0	421.9
watercontamination0303r	0.0	0.0	29.2	0.0	0.0	25.1	85.6	∞	1800.4

Table 9: Detailed solving data for the evaluation of initializing (MIP^i) in Section 4.3.

Instance	DICOPT w/ FP			DICOPT w/ FP w/o OA init		
	p.gap	gap	time	p.gap	gap	time
batch	0.0	0.0	0.3	0.0	0.0	0.3
batchs101006m	0.0	0.0	3.4	0.0	0.0	5.9
batchs121208m	0.0	0.0	5.7	0.0	0.0	6.1
batchs151208m	0.0	0.0	10.4	0.0	0.0	11.3
batchs201210m	0.0	0.0	8.2	0.0	0.0	8.4
clay0205m	–	–	–	–	–	–
clay0303m	0.0	0.0	9.5	0.0	0.0	374.5
clay0304m	0.0	68.6	1800.1	31.9	85.9	1800.0
clay0305m	–	–	–	–	–	–
du-opt	0.0	0.0	7.5	0.0	0.0	9.3
flay05h	0.0	0.0	608.4	0.0	0.0	522.3
flay05m	0.0	0.0	192.8	0.0	0.0	210.9
flay06h	0.0	2.8	1800.0	0.0	3.5	1800.0
flay06m	0.0	1.0	1800.0	0.0	0.8	1800.0
fo7	0.0	0.0	50.0	0.0	0.0	61.3
fo7_2	0.0	0.0	14.6	0.0	0.0	16.8
fo8	0.0	0.0	110.8	0.0	0.0	120.7
fo9	0.0	0.0	356.2	0.0	0.0	212.0
fo9_ar2_1	0.0	0.0	958.3	0.0	0.0	793.3
fo9_ar3_1	0.0	0.0	38.7	0.0	0.0	65.8
fo9_ar4_1	0.0	0.0	47.4	0.0	0.0	71.6
fo9_ar5_1	0.0	0.0	175.7	0.0	0.0	665.3
gams01	4.9	∞	1800.0	4.9	∞	1800.0
ibs2	77.1	77.2	1800.0	73.2	73.2	1800.0
m7_ar2_1	0.0	0.0	6.7	0.0	0.0	9.3
m7_ar3_1	0.0	0.0	9.5	0.0	0.0	13.1
m7_ar5_1	0.0	0.0	2.9	0.0	0.0	2.8
netmod_dol1	8.5	21.5	1800.0	8.5	21.5	1800.0
netmod_dol2	0.0	0.0	167.0	–	–	–
netmod_kar1	0.0	0.0	53.2	0.0	0.0	64.0
netmod_kar2	0.0	0.0	52.6	0.0	0.0	64.0
no7_ar2_1	–	–	–	–	–	–
no7_ar3_1	–	–	–	–	–	–
no7_ar4_1	0.0	0.0	215.5	0.0	0.0	346.8
no7_ar5_1	0.0	0.0	231.7	0.0	0.0	316.8
o7	0.0	0.5	1800.0	12.2	12.8	1800.0
o7_2	0.0	0.0	420.6	0.0	0.0	936.8
o7_ar25_1	0.0	0.0	526.1	0.0	0.0	496.9
o7_ar5_1	0.0	0.0	384.0	0.0	0.0	814.1
o8_ar4_1	13.6	∞	1800.0	13.6	19.5	1800.0
o9_ar4_1	∞	∞	1800.0	∞	∞	1800.0
portfol_classical050_1	0.5	3.4	1800.0	0.2	3.3	1800.0
portfol_classical200_2	8.6	19.3	1800.0	7.1	17.9	1800.1
rsyn0830m04h	0.0	0.0	11.3	0.0	0.0	11.5
rsyn0830m04m	0.0	0.0	89.0	0.0	0.0	91.3
rsyn0840m04h	0.0	0.0	11.5	0.0	0.0	11.8
rsyn0840m04m	0.0	0.0	29.7	–	–	–
slay09h	0.0	0.0	157.3	0.0	0.0	80.4
slay09m	0.0	0.0	27.3	0.0	0.0	30.3
slay10h	0.0	0.7	1800.0	0.0	0.6	1800.0
slay10m	0.0	0.0	1283.8	0.0	0.0	959.0
squfl020-150	18.5	64.8	1800.0	18.5	64.7	1800.0
squfl030-100	22.7	70.5	1800.0	22.7	70.5	1800.0
squfl030-150	27.1	72.5	1800.1	27.1	72.7	1800.1
squfl040-080	26.1	71.8	1800.0	26.7	72.1	1800.0
sssd18-08	0.0	0.0	297.2	–	–	–
sssd20-08	–	–	–	–	–	–
sssd22-08	–	–	–	–	–	–
sssd25-08	0.0	0.0	40.5	0.0	0.0	93.8
stockcycle	15.6	17.1	1800.0	27.9	29.1	1800.0
syn40m03h	0.0	0.0	3.8	0.0	0.0	4.4
syn40m04h	–	–	–	–	–	–
tls12	∞	∞	1802.2	∞	∞	1810.9
tls5	∞	∞	1800.0	∞	∞	1800.0
tls6	∞	∞	1800.0	∞	∞	1800.0
tls7	∞	∞	1801.0	∞	∞	1800.0
watercontamination0303	0.0	0.0	224.5	0.0	0.0	223.2
watercontamination0303r	0.0	0.0	29.2	0.0	0.0	27.8

Table 10: Detailed solving data for the comparison of the different MINLP solvers in Section 4.4.

Instance	AlphaECP			BONMIN			DICOPT w/ FP			DICOPT w/o FP			SCIP		
	p.gap	gap	time	p.gap	gap	time	p.gap	gap	time	p.gap	gap	time	p.gap	gap	time
batch	0.0	0.0	0.8	0.0	0.0	0.1	0.0	0.0	0.3	0.0	0.0	0.1	0.0	0.0	0.2
batchs101006m	0.0	0.0	15.7	0.0	0.0	15.7	0.0	0.0	3.4	0.0	0.0	4.7	0.0	0.0	7.5
batchs121208m	0.0	0.0	78.6	0.0	0.0	24.9	0.0	0.0	5.7	0.0	0.0	3.9	0.0	0.0	13.8
batchs151208m	–	–	–	0.0	0.0	33.5	0.0	0.0	10.4	0.0	0.0	8.3	0.0	0.0	16.0
batchs201210m	0.0	0.0	122.9	0.0	0.0	66.0	0.0	0.0	8.2	0.0	0.0	7.9	0.0	0.0	18.0
clay0205m	0.0	0.0	20.1	0.0	0.0	14.2	–	–	–	–	–	–	0.0	0.0	7.6
clay0303m	0.0	0.0	4.1	0.0	0.0	6.2	0.0	0.0	9.5	0.0	0.0	374.2	0.0	0.0	1.8
clay0304m	0.0	0.0	13.5	0.0	0.0	156.5	0.0	68.6	1800.1	31.9	85.9	1800.0	0.0	0.0	5.0
clay0305m	0.0	0.0	31.6	0.0	0.0	19.0	–	–	–	–	–	–	0.0	0.0	15.7
du-opt	0.0	0.0	44.1	–	–	–	0.0	0.0	7.5	0.0	0.0	9.0	0.0	0.0	1.5
flay05h	0.0	10.8	1802.0	0.0	0.0	374.6	0.0	0.0	608.4	0.0	0.0	520.7	0.0	0.0	212.2
flay05m	0.0	2.9	1802.0	0.0	0.0	99.0	0.0	0.0	192.8	0.0	0.0	212.2	0.0	0.0	70.3
flay06h	0.0	3.8	1802.2	0.0	6.4	1840.8	0.0	2.8	1800.0	0.0	2.1	1800.0	0.0	10.1	1800.0
flay06m	0.0	4.3	1802.0	0.0	9.3	1841.8	0.0	1.0	1800.0	0.0	0.8	1800.0	–	–	–
fo7	0.0	0.0	198.3	0.0	0.0	102.2	0.0	0.0	50.0	0.0	0.0	51.2	0.0	0.0	104.7
fo7_2	0.0	0.0	18.3	0.0	0.0	70.2	0.0	0.0	14.6	0.0	0.0	7.9	0.0	0.0	31.8
fo8	0.0	0.0	691.7	0.0	0.0	181.1	0.0	0.0	110.8	0.0	0.0	98.8	0.0	0.0	300.0
fo9	0.0	0.0	725.3	0.0	0.0	826.7	0.0	0.0	356.2	0.0	0.0	209.3	0.0	29.0	1800.0
fo9_ar2_1	0.0	0.0	1620.1	0.0	9.7	1813.0	0.0	0.0	958.3	0.0	0.0	724.9	0.0	0.0	1057.2
fo9_ar3_1	0.0	0.0	749.3	0.0	0.0	851.7	0.0	0.0	38.7	0.0	0.0	48.6	0.0	0.0	64.9
fo9_ar4_1	0.0	0.0	457.9	0.0	0.0	652.5	0.0	0.0	47.4	0.0	0.0	69.7	0.0	0.0	717.8
fo9_ar5_1	0.0	0.0	1696.8	0.0	0.0	574.6	0.0	0.0	175.7	0.0	0.0	584.2	0.0	0.0	1249.3
gams01	24.8	∞	1802.5	6.1	93.8	1801.7	4.9	∞	1800.0	∞	∞	1800.0	0.6	93.7	1800.0
ibs2	74.6	∞	1802.0	∞	∞	nan	77.1	77.2	1800.0	73.2	73.2	1800.0	0.3	33.3	1800.0
m7_ar2_1	0.0	0.0	12.9	0.0	0.0	53.2	0.0	0.0	6.7	0.0	0.0	4.6	0.0	0.0	6.6
m7_ar3_1	0.0	0.0	6.6	0.0	0.0	43.8	0.0	0.0	9.5	0.0	0.0	4.8	0.0	0.0	10.3
m7_ar5_1	0.0	0.0	3.8	0.0	0.0	136.4	0.0	0.0	2.9	0.0	0.0	1.5	0.0	0.0	9.8
netmod_dol1	8.5	27.2	1802.0	–	–	–	8.5	21.5	1800.0	8.5	21.5	1800.0	0.0	9.8	1800.0
netmod_dol2	0.0	0.0	279.5	–	–	–	0.0	0.0	167.0	–	–	–	0.0	0.0	36.9
netmod_kar1	0.0	0.0	71.5	–	–	–	0.0	0.0	53.2	0.0	0.0	60.5	0.0	0.0	3.0
netmod_kar2	0.0	0.0	71.7	–	–	–	0.0	0.0	52.6	0.0	0.0	60.4	0.0	0.0	3.0
no7_ar2_1	0.0	0.0	88.0	0.0	0.0	220.7	–	–	–	–	–	–	0.0	0.0	46.8
no7_ar3_1	0.0	0.0	346.5	0.0	0.0	224.9	–	–	–	–	–	–	0.0	0.0	221.6
no7_ar4_1	0.0	0.0	404.3	0.0	0.0	172.0	0.0	0.0	215.5	0.0	0.0	198.8	0.0	0.0	150.2
no7_ar5_1	0.0	0.0	234.1	0.0	0.0	96.7	0.0	0.0	231.7	0.0	0.0	140.5	0.0	0.0	65.8
o7	0.0	11.4	1802.1	0.0	0.0	1785.4	0.0	0.5	1800.0	0.0	0.4	1800.0	0.0	9.7	1800.0
o7_2	0.0	0.0	1362.9	0.0	0.0	645.6	0.0	0.0	420.6	0.0	0.0	834.3	0.0	0.0	945.6
o7_ar25_1	0.0	0.0	1015.4	0.0	0.0	722.3	0.0	0.0	526.1	0.0	0.0	345.7	0.0	0.0	423.7
o7_ar5_1	0.0	0.0	891.5	0.0	0.0	489.4	0.0	0.0	384.0	0.0	0.0	636.1	0.0	0.0	580.8
o8_ar4_1	10.5	19.1	1802.4	0.0	10.5	1812.7	13.6	∞	1800.0	∞	∞	1800.0	0.0	23.6	1800.0
o9_ar4_1	12.7	51.6	1802.1	10.6	34.8	1811.3	∞	∞	1800.0	∞	∞	1800.0	0.0	25.5	1800.0
portfol_classical050_1	0.0	5.1	1802.0	1.4	4.5	1828.2	0.5	3.4	1800.0	0.2	3.3	1800.0	0.0	0.0	127.2
portfol_classical200_2	44.5	63.2	1802.0	7.0	17.8	1807.8	8.6	19.3	1800.0	7.1	17.9	1800.1	3.6	17.5	1800.0
rsyn0830m04h	0.0	0.0	54.8	0.0	0.0	23.3	0.0	0.0	11.3	0.0	0.0	4.0	0.0	0.0	3.4
rsyn0830m04m	0.0	0.0	302.0	0.8	9.5	1848.2	0.0	0.0	89.0	0.0	0.0	10.5	0.4	19.1	1800.0
rsyn0840m04h	0.0	0.0	59.1	0.0	0.0	19.2	0.0	0.0	11.5	0.0	0.0	4.1	0.0	0.0	3.3
rsyn0840m04m	0.0	0.0	446.2	1.3	19.7	1845.0	0.0	0.0	29.7	–	–	–	–	–	–
slay09h	0.0	3.4	1802.0	1.4	4.5	1838.3	0.0	0.0	157.3	0.0	0.0	69.3	0.0	0.0	248.7
slay09m	–	–	–	0.0	0.0	31.3	0.0	0.0	27.3	0.0	0.0	27.9	0.0	0.0	26.5
slay10h	0.0	4.6	1802.0	3.3	10.6	1840.3	0.0	0.7	1800.0	0.0	0.6	1800.0	0.0	5.4	1800.0
slay10m	0.0	2.4	1802.0	0.0	0.0	548.7	0.0	0.0	1283.8	0.0	0.0	961.8	0.0	0.0	149.4
squfl020-150	0.8	∞	1802.1	5.7	49.8	1804.5	18.5	64.8	1800.0	20.9	65.8	1800.0	30.1	99.2	1800.1
squfl030-100	20.1	∞	1802.1	5.5	57.4	1805.0	22.7	70.5	1800.0	22.7	70.5	1800.0	78.1	93.8	1800.2
squfl030-150	9.3	∞	1802.1	10.2	64.7	1801.8	27.1	72.5	1800.1	35.1	75.6	1800.1	24.5	∞	1800.0
squfl040-080	26.2	∞	1802.0	9.6	64.4	1807.7	26.1	71.8	1800.0	17.7	68.6	1800.0	78.8	94.1	1800.2
sssd18-08	0.0	0.0	152.2	2.0	24.0	1840.9	0.0	0.0	297.2	–	–	–	0.0	0.0	967.8
sssd20-08	0.0	0.0	378.7	0.5	15.7	1836.5	–	–	–	–	–	–	0.0	0.0	1800.0
sssd22-08	0.0	0.0	308.2	0.8	17.1	1839.1	–	–	–	–	–	–	0.0	0.0	815.6
sssd25-08	0.0	0.0	901.0	0.9	14.7	1844.5	0.0	0.0	40.5	0.0	0.0	91.7	0.0	0.0	432.1
stockcycle	2.2	10.3	1802.0	6.9	8.5	1830.8	15.6	17.1	1800.0	42.6	43.6	1800.0	0.0	0.0	197.6
syn40m03h	0.0	0.0	17.7	–	–	–	0.0	0.0	3.8	0.0	0.0	1.6	0.0	0.0	1.2
syn40m04h	0.0	0.0	28.4	–	–	–	–	–	–	–	–	–	0.0	0.0	1.7
tls12	∞	∞	1802.2	∞	∞	1812.5	∞	∞	1802.2	∞	∞	1815.0	∞	∞	1800.3
tls5	24.3	60.3	1802.0	23.7	68.3	1812.4	∞	∞	1800.0	∞	∞	1800.0	0.0	31.1	1800.3
tls6	59.8	80.8	1802.1	–	–	–	∞	∞	1800.0	∞	∞	1800.0	0.0	44.5	1800.6
tls7	69.9	93.0	1802.2	∞	∞	1802.6	∞	∞	1801.0	∞	∞	1800.0	6.8	69.6	1800.4
watercontamination0303	0.0	∞	1800.6	–	–	–	0.0	0.0	224.5	0.0	0.0	126.9	74.8	∞	1800.1
watercontamination0303r	0.0	∞	1800.1	0.0	0.0	25.6	0.0	0.0	29.2	0.0	0.0	25.1	0.0	0.0	1800.0