Noname manuscript No.
(will be inserted by the editor)

A Sequential Benders-based Mixed-Integer Quadratic Programming Algorithm

Andrea Ghezzi*, Wim Van Roy*,
Sebastian Sager, Moritz Diehl

Received: xxxx / Accepted: xxxx

Abstract For continuous decision spaces, nonlinear programs (NLPs) can be efficiently solved via sequential quadratic programming (SQP) and, more generally, sequential convex programming (SCP). These algorithms linearize only the nonlinear equality constraints and keep the outer convex structure of the problem intact, such as (conic) inequality constraints or convex objective terms.The aim of the presented sequential mixed-integer quadratic programming (MIQP) algorithm for mixed-integer nonlinear problems (MINLPs) is to extend the SQP/SCP methodology to MINLPs and leverage the availability of efficient MIQP solvers. The algorithm employs a three-step method in each iterate: First, the MINLP is linearized at a given iterate ("linearization point"). Second, an MIQP with its feasible set restricted to a specific region around the current linearization point is formulated and solved. Third, the integer variables obtained from the MIQP solution are fixed, and only an NLP in the continuous variables is solved. The outcome of the third step is compared to previous iterates, and the best iterate so far is used as a linearization point in the next iterate. Crucially, the objective values and derivatives from all previous iterates are used to formulate the polyhedral region in the second step. The linear inequalities that define the region build on concepts from generalized Benders' decomposition (GBD) for MINLPs. Although the presented MINLP algorithm is a heuristic method without any global optimality guarantee, it converges to the exact integer solution when applied to convex MINLP with a linear outer

* The authors contributed equally

A. Ghezzi
Department of Microsystems Engineering (IMTEK) University of Freiburg, Germany
E-mail: andrea.ghezzi@imtek.uni-freiburg.de
Corresponding author

W. Van Roy
Department of Mechanical Engineering, KU Leuven, Belgium
E-mail: wim.vanroy@kuleuven.be

S. Sager
Institute of Mathematical Optimization, Otto-von-Guericke Universität Magdeburg, Germany
E-mail: sager@ovgu.de

M. Diehl
Department of Microsystems Engineering (IMTEK) and Department of Mathematics, University of Freiburg, Germany
E-mail: moritz.diehl@imtek.uni-freiburg.de

structure. The conducted numerical experiments demonstrate that the proposed algorithm is competitive with other open-source solvers for MINLP. Finally, we utilize the proposed algorithm to solve two mixed-integer optimal control problems (MIOCPs) transcribed into MINLPs via direct methods. By approximately solving such problems we aim to demonstrate that the presented algorithm can effectively deal with nonlinear equality constraints, a major hurdle for generic MINLP solvers.

Keywords Mixed-integer nonlinear programming (MINLP) · sequential mixed-integer quadratic programming · generalized Benders' decomposition · outer approximation · optimal control

Mathematics Subject Classification (2020) 90C11 · 90C30 · 90C59 · 65K05 · 49M25 · 49-04

1 Introduction

The class of mixed-integer nonlinear programs (MINLPs) comprises problems characterized by continuous and discrete variables coupled with nonlinear relationships in the objective or constraints. Hence, MINLPs represent a powerful optimization paradigm that offers a natural way to formulate a wide range of problems and applications. The intersection of nonlinearity and discrete variables poses unique challenges for the numerical solution of such problems, which are characterized by NP-hard complexity [25, 34]. For unbounded support, they are even undecidable, meaning that not every unbounded MINLP problem can be solved.

Several excellent surveys and books cover algorithms for solving MINLPs [6], [50, §21]. Here, we provide only a high-level picture of the field and focus on the literature directly connected to the algorithm we propose.

We can divide the existing algorithms into two families. The first consists of branch-and-bound-type algorithms, such as nonlinear branch-and-bound [18, 28] and spatial branch-and-bound [51]. The second family relies on the creation of cutting planes to iteratively tighten the integer search space, as in the generalized Benders' decomposition (GBD) [26], outer approximation [21] and its quadratic version [23], and extended supporting cutting planes [55, 35]. A combination of both families includes LP/NLP-based branch-and-bound [45] and branch-and-check [52]. Branch-and-bound-based methods are appealing because they leverage the existence of reliable, efficient, and open-source nonlinear solvers such as IPOPT [54] and WORHP [15]. In contrast, the second family of methods additionally requires efficient mixed-integer linear (quadratic) solvers such as the open-source CBC [24], SCIP [2], Highs [31], as well as the commercial CPLEX [32], Gurobi [29], and Mosek [44]. Moreover, there also exist solvers that directly implement MINLP-specific algorithms, such as the open-source Bonmin [8], Couenne [5], SHOT [40], and the commercial Antigone [43], Baron [49], Knitro [17], and Gurobi. Regarding sequential programming algorithms for mixed integer programs, the literature provides only a few suggestions [22]. Therein, the presented algorithm resembles a standard trust-region sequential quadratic programming (SQP) for continuous decision spaces. The algorithm can be applied to *nonconvex* MINLPs and also to MINLPs where the integer variable cannot be relaxed. However, the authors do not provide a proof of convergence to the global minimizer, even for *convex* MINLPs. The Mittelmann benchmarks webpage [41] gives an up-to-date overview of the current solvers and their performance.

Most generic MINLP solvers focus on solving *convex* MINLPs, often providing only heuristics for *nonconvex* ones. Some solvers, such as those based on outer approximation, may fail, particularly with problems containing numerous nonlinear equality constraints. A practical example where these constraints arise is in the solution of mixed-integer optimal control problems (MIOCPs) via direct methods, such as direct collocation [53] or direct multiple shooting [7]. In such cases, equality constraints enforce continuity of the underlying dynamics equations at every grid node.

For the solution of MIOCPs, an important class of methods known as combinatorial integral approximation (CIA) [48] employs an error-controlled decomposition approach. In practice, the MIOCP is discretized to obtain an MINLP, which is then approximately solved by computing an integer approximation. This approximation minimizes the distance from the relaxed solution of the MINLP using a dedicated norm. An open-source implementation of this approach, named pycombina [12], has been developed and demonstrated too be effective in various engineering applications [13, 47]. This method is capable of handling generic dwell time constraints [56] and provides fast approximate solutions, with the main computational challenge lying in the nonlinear optimization rather than combinatorial aspects. The method relies on the similarity of the relaxed solution with the optimal one. This is a drawback when dealing with coarse discretization grids or long uptimes, as shown in [14, 1]. To address this, [14] proposes a new distance function for the second step, based on a quadratic programming approximation around the relaxed MINLP solution. This approach often enhances the quality of the MIOCP solution in terms of both objective and constraint satisfaction.

Another class of methods for approximately solving MIOCP is switching time optimization (STO). This approach relies on an initial sequence of states that can be applied to the system, with switch times optimized. During the iterations, the sequence is adjusted either by inserting additional elements [37, 4] or by removing unnecessary sequences [1]. Also this approach is a heuristic and relies on the initial sequence provided to the solver.

In this work, we introduce an algorithm for solving MINLPs, which draws inspiration from [14] and classical MINLP methods based on outer approximation. Traditionally, such methods are not well-suited for MINLPs with nonlinear equality constraints. Our algorithm leverages the cuts generated by GBD to construct a Benders-region, which is integrated into the algorithm, similarly to the Voronoi-region constraint proposed in [27]. We demonstrate that the algorithm converges to the global optimal solution for *convex* MINLPs, while providing a valid heuristic for *nonconvex* ones.

1.1 Contribution and outline

In the remainder of this section, we outline some preliminary definitions and notions utilized throughout the rest of the paper. In Section 2, we introduce the new algorithm, which comprises a three-step method with alternating nonlinear program (NLP) and mixed-integer program (MIP) steps, efficiently combining cutting planes based on GBD and outer approximation. Subsequently, we describe in detail all its constituent components. Furthermore, assuming convexity, we prove that the algorithm converges to the global optimum or with a certificate of infeasibility for the given MINLP. Finally, we conclude the section by illustrating the behavior of the proposed algorithm with a simple tutorial example. In Section 3, we propose an extension for treating nonconvex MINLPs by introducing heuristics to modify the generated cutting planes. We demonstrate that these introduced heuristics do not compromise convergence to the global minimizer in the case of *convex* MINLPs. Additionally,

for MINLPs where the integer variables enter affinely, making the relaxed integer feasible set convex, we prove that the proposed algorithm terminates with either a feasible solution or a certificate of infeasibility. In Section 4, we compare the proposed algorithm against SHOT and Bonmin, two established MINLP solvers, using a selected but large subset of generic MINLPs from the MINLPLib [16] containing at least one integer and one continuous variable. This comparison reveals that the presented algorithm converges to a solution with a lower objective value for many instances compared to SHOT and Bonmin. Regarding computation time, the proposed algorithm requires more computation time than SHOT but less than Bonmin. Finally, we present the results obtained with the proposed algorithm in two cases of optimal control for switched systems: a textbook example of a small, nonlinear, and unstable system, and a complex nonlinear energy system for building control. For these examples, a comparison against SHOT was not possible. However, we compare again against Bonmin and the specialized algorithm CIA. In both examples, the proposed algorithm outperforms Bonmin in terms of speed while yielding either the same or a lower objective solution. Compared to CIA, the proposed algorithm also achieves a solution with a lower objective, although, as expected, the computation time is higher. The gap between solution quality and computation time for the complex example is dramatic. While the proposed algorithm achieves significantly better solutions compared to CIA, the computation times that might be impractical for real-world control applications.

Alongside the methods and numerical examples, we provide an open-source implementation[1] in Python based on CasADi [3]. Users have the flexibility to choose from various solvers interfaced by CasADi, to solve both the NLPs and the MIPs, enabling an implementation free from reliance on commercial solvers. In addition to the new algorithm presented in this paper, the software package includes a comprehensive range of other existing algorithms, such as GBD, (quadratic) outer approximation, feasibility pumps, and the ability to use seamlessly MINLP solvers already interfaced by CasADi, such as Bonmin. Further details are available in the `README` file of the repository.

1.2 Notation

We denote with $\mathbb{Z}_{[a,b]}$ the set of integer numbers in the interval $[a,b]$ with $a,b \in \mathbb{Z}$ and $a < b$. For a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ we extend the gradient notation such that $\nabla f(z) \equiv \left(\frac{\partial f}{\partial z}(z)\right)^\top$. We define the linearization of a function f at $(\bar{x},\bar{y})$ as a Taylor series of first order $f_\mathrm{L}(x,y;\bar{x},\bar{y}) := f(\bar{x},\bar{y}) + \frac{\partial f}{\partial (x,y)}(\bar{x},\bar{y}) \begin{pmatrix} x-\bar{x} \\ y-\bar{y} \end{pmatrix}$. The term *convex* MINLP refers to a MINLP that is convex with respect to both the the continuous optimization variables and the relaxed integer variables.

1.3 Mixed-Integer Nonlinear Problem formulations

In this paper, we consider the generic MINLP as

[1] `https://github.com/minlp-toolbox/minlp-algorithms`

$$\mathcal{P}_{\text{MINLP}}: \quad \begin{aligned} \min_{x \in X, y \in Y} \quad & f(x,y) \\ \text{s.t.} \quad & g(x,y) \leq 0, \\ & h(x,y) = 0. \end{aligned} \tag{1}$$

We can express (1) more compactly as:

$$\min_{y \in Y} \quad J(y), \tag{2}$$

where

$$\begin{aligned} J(y) \coloneqq \min_{x \in X} \quad & f(x,y) \\ \text{s.t.} \quad & g(x,y) \leq 0, \\ & h(x,y) = 0. \end{aligned} \tag{3}$$

In the remainder of the work, in slight abuse of notation, we use $J(y)$ to denote either an optimization problem, a function, or a specific objective value. The intended usage should be clear from the context. Additionally, we denote the subgradient of the possibly nonsmooth function $J : Y \to \mathbb{R}$ as ∇J. The computation of the subgradient ∇J is detailed in Section 2.2. The proofs in this work rely on the following assumptions regarding problem (1).

Assumption 1. *We assume that*

1. *$Y = \bar{Y} \cap \mathbb{Z}^{n_y}$, where both $X \subset \mathbb{R}^{n_x}$ and $\bar{Y} \subset \mathbb{R}^{n_y}$ are closed convex polyhedral sets.*
2. *Functions $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}$, $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_g}$ and $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_h}$ are at least once continuously differentiable.*
3. *The integer set Y is finite.*

Assumption 2 (Convexity)**.** *Function h is affine, and functions $g_1, \ldots, g_{n_g}$ and f are convex on $X \times \bar{Y}$.*

Definition 1. *We define the feasible set of the integer variables as*

$$\mathcal{F} \coloneqq \{y \in Y \mid \exists x \in X, g(x,y) \leq 0, h(x,y) = 0\}.$$

Similarly, we define the feasible set of the relaxed integer variables as

$$\bar{\mathcal{F}} \coloneqq \{y \in \bar{Y} \mid \exists x \in X, g(x,y) \leq 0, h(x,y) = 0\}.$$

In the following, we only require Assumption 1 to hold unless specified otherwise.

1.4 The MILP/MIQP approximation

We outline the process of deriving a MILP/MIQP approximation of problem (2), which constitutes a key component of the proposed algorithm. Let us denote the linearizations of functions f, g, h at $(\bar{x}, \bar{y})$ as $f_{\text{L}}(\cdot\,; \bar{x}, \bar{y}), g_{\text{L}}(\cdot\,; \bar{x}, \bar{y})$, and $h_{\text{L}}(\cdot\,; \bar{x}, \bar{y})$, respectively. The resulting quadratic programming based approximation of J is defined as

$$
\begin{aligned}
J_{\mathrm{QP}}(y;\bar{x},\bar{y},B) \coloneqq \min_{x \in X} \quad & f_{\mathrm{L}}(x,y;\bar{x},\bar{y}) + \frac{1}{2}\begin{pmatrix} x-\bar{x} \\ y-\bar{y} \end{pmatrix}^{\top} B \begin{pmatrix} x-\bar{x} \\ y-\bar{y} \end{pmatrix} \\
\text{s.t.} \quad & g_{\mathrm{L}}(x,y;\bar{x},\bar{y}) \leq 0, \\
& h_{\mathrm{L}}(x,y;\bar{x},\bar{y}) = 0,
\end{aligned}
\tag{4}
$$

where B is a positive semidefinite Hessian approximation.

Remark 1 In many discretized optimal control problems, the cost function f in (1) takes the form of a nonlinear least squares function:

$$
f(x,y) \coloneqq \frac{1}{2}\left\|f_1(x,y)\right\|_2^2 + f_2(x,y), \tag{5}
$$

where $f_1 : \mathbb{R}^{n_x+n_y} \to \mathbb{R}^{n_1}$ and $f_2 : \mathbb{R}^{n_x+n_y} \to \mathbb{R}$ are differentiable functions, and $\|\cdot\|_2$ denotes the L_2 norm. In this case, the Gauss-Newton Hessian approximation is commonly used, defined as

$$
B \coloneqq \nabla f_1(\bar{x},\bar{y}) \nabla f_1(\bar{x},\bar{y})^{\top},
$$

and when using this choice of B, problem (4) coincides with the Gauss-Newton MIQP presented in [14].

2 Sequential Benders MIQP algorithm

The algorithm proposed in this work leverages various approximations of the function $J(y)$ based on the first and second derivatives of the problem functions. Initially, we utilize a quadratic programming-based approximation of J, denoted as J_{QP}. This approximation is expected to perform effectively within a defined region $\mathbb{B}$ around the current linearization point, determined by the best-visited point thus far, denoted as (x_b, y_b). Additionally, alongside J_{QP}, we construct a lower bound function J_{LB} that aggregates first-order information from all visited points so far, similar to the GBD [26]. These two functions, J_{QP} and J_{LB}, serve as the foundation for constructing the two master problems responsible for addressing the combinatorial nature of (2). Finally, we construct an auxiliary problem to assess the quality of the approximated solutions for the original MINLP (2). This evaluation is crucial for updating the linearization point and guiding the search for an optimal solution.

Moving forward, we first present the complete algorithm, outlining its main features. Following this overview, we delve into the constituent components, namely the auxiliary problem, the master problems, and the termination condition.

2.1 The S-B-MIQP algorithm

Algorithm 1 presents the Sequential Benders MIQP (S-B-MIQP) algorithm. Here, we consider Assumption 2 about convexity to hold; thus, line 16 is omitted and we assume $\tilde{\mathcal{D}}_k \coloneqq \mathcal{D}_k$. We address the nonconvex case in Section 3.

In Algorithm 1, the letter k, primarily used for subscripts, denotes the iteration index, where $k \in \mathbb{Z}$. Algorithm 1 begins with an integer point $y_0 \in Y$ provided by the user, along with a lower bound, LB, obtained by solving (2) under integer relaxation. Also, the value $\bar{V}$ is user-defined and corresponds to a large number, with $\bar{V} < \mathrm{UB}$. Following initialization, we

enter a cycle that terminates only when the lower bound becomes greater or equal than the upper bound, UB. This termination condition is common for mixed-integer programming algorithms.

The first half of the cycle involves evaluating the quality of the integer solution y_k and solving for the continuous variable x. In each iteration, the upper bound is updated by comparing the current UB with $J(y_k)$, the objective value of the NLP obtained by fixing the integer variable of the MINLP (2) to y_k. This NLP with the fixed integer variable is denoted as $\mathcal{P}_{\text{NLP}}$, and its solution x_k provides the optimized continuous variable. The zero and first-order information of its objective are used to construct the Benders region $\mathbb{B}_k$. Note that $\mathcal{P}_{\text{NLP}}$ might be infeasible for some y_k. Therefore, we anticipate a feasibility problem $\mathcal{P}_{\text{FNLP}}$, whose solution is used to construct specific cutting planes aiming to steer the integer solution back to the feasible set. We refer to $\mathcal{P}_{\text{NLP}}$ and $\mathcal{P}_{\text{FNLP}}$ as auxiliary problems. The index sets, $\mathbb{T}$ and $\mathbb{S}$, store the index of feasible and infeasible iterates, respectively.

The second half of Algorithm 1 focuses on computing new integer solutions and lower bounds. The integer solutions are computed in the master problems $\mathcal{P}_{\text{TR}-\text{MIQP}}$ and $\mathcal{P}_{\text{LB}-\text{MILP}}$, which are an MIQP and an MILP, respectively. The idea is to approach the minimizer of $\mathcal{P}_{\text{MINLP}}$ (2) by solving MIQPs, as done in SQP for continuous problems. Moreover, in the context of mixed-integer programming, solving MIQPs as master problems rather than MILPs has shown to be beneficial for the outer approximation algorithm [23]. Nowadays, we can also leverage the existence of highly efficient MIQP solvers, where the runtime gap between MILP and MIQP has been dramatically reduced.

On the contrary, solving exclusively MIQPs as master problems does not guarantee the convergence to a global minimizer even under convexity Assumption 2. For this reason, we anticipate a fallback strategy based on the solution of a MILP obtained by discarding the Hessian term of $\mathcal{P}_{\text{TR}-\text{MIQP}}$ and reformulating it in an epigraph form. The objective of $\mathcal{P}_{\text{LB}-\text{MILP}}$ provides a lower bound on the objective of (2). In summary, we solve $\mathcal{P}_{\text{TR}-\text{MIQP}}$ to quickly attain integer solutions y_k that produce low objectives of $\mathcal{P}_{\text{NLP}}$. As soon as the solution of $\mathcal{P}_{\text{TR}-\text{MIQP}}$ stagnates, we start solving $\mathcal{P}_{\text{LB}-\text{MILP}}$ to obtain a lower bound for the current best solution and a new integer solution to evaluate in the next iterate.

In the next subsections, we present $\mathcal{P}_{\text{NLP}}$, $\mathcal{P}_{\text{FNLP}}$, $\mathcal{P}_{\text{BR}-\text{MIQP}}$, $\mathcal{P}_{\text{LB}-\text{MILP}}$, $b(k)$ and $\mathcal{D}_k$.

Algorithm 1 Sequential Benders MIQP (S-B-MIQP)

Initialize: $y_0 \in Y$, $\mathbb{B}_0 \leftarrow Y$, $k \leftarrow 0$, $\mathbb{S}_{-1} \leftarrow \emptyset$, $\mathbb{T}_{-1} \leftarrow \emptyset$, UB $= +\infty$, LB $= -\infty$, $V_0 \leftarrow \bar{V}$
LB $\leftarrow \min_{y \in \bar{Y}} J(y)$ ▷ `LB given by the objective of MINLP relaxation`
$b(k) \leftarrow 0$
while UB $>$ LB **do**:
  Solve $\mathcal{P}_{\text{NLP}}(y_k)$
  if $\mathcal{P}_{\text{NLP}}(y_k)$ is feasible **then**
    Store solution $(k, x_k, y_k, J(y_k), \nabla J(y_k))$ in $\mathcal{D}_k$ and $\mathbb{T}_k \leftarrow \mathbb{T}_{k-1} \cup \{k\}$
    if $J(y_k) <$ UB **then**: ▷ `Update the best solution`
      UB $\leftarrow J(y_k)$
    end if
  else
    Solve $\mathcal{P}_{\text{FNLP}}(y_k, y_{b(k)})$ and obtain $\bar{y}_k$
    Store solution $(k, x_k, y_k, \bar{y}_k, J_{\text{f}}(\bar{y}_k; y_k), \nabla J(\bar{y}_k))$ in $\mathcal{D}_k$ and $\mathbb{S}_k \leftarrow \mathbb{S}_{k-1} \cup \{k\}$
  end if
  Compute $b(k)$ according to (15)
  Modify gradients in $\mathcal{D}_k$, obtain $\tilde{\mathcal{D}}_k$
  Compute Benders region $\mathbb{B}_k$ based on $\tilde{\mathcal{D}}_k$ according to (17)
  if $V_k \leq$ UB **then**:
    Solve $\mathcal{P}_{\text{BR-MIQP}}$ with $J_{\text{QP}}(y; x_{b(k)}, y_{b(k)}, B_{b(k)})$ and $\mathbb{B}_k$, store solution $\tilde{y}$
    $V_{k+1} \leftarrow V_{\text{MIQP}}$
  end if
  if $V_k >$ UB or $\mathcal{P}_{\text{BR-MIQP}}$ is infeasible **then**:
    Solve $\mathcal{P}_{\text{LB-MILP}}$ with $J_{\text{LB}}(y; \mathbb{T}_k, \mathbb{S}_k, \tilde{\mathcal{D}}_k)$, obtain solution $\tilde{y}$ and V_{MILP}
    if $\mathcal{P}_{\text{LB-MILP}}$ is infeasible **then**
      LB $\leftarrow +\infty$
    else
      if $V_{\text{MILP}} >$ LB **then**
        LB $\leftarrow V_{\text{MILP}}$
      end if
    end if
  end if
  $y_{k+1} \leftarrow \tilde{y}$
  $b(k+1) \leftarrow b(k)$
  $k \leftarrow k+1$
end while

2.2 Auxiliary problem

The auxiliary problem $\mathcal{P}_{\text{NLP}}$ is an NLP obtained by fixing the integer variables of the original MINLP (1). Hence, given any fixed integer vector $\tilde{y} \in Y$, the auxiliary NLP and its optimal value $J(\tilde{y})$ are

$$\mathcal{P}_{\text{NLP}}(\tilde{y}):\quad J(\tilde{y}) := \min_{x \in X} \; f(x,\tilde{y}) \tag{6a}$$
$$\text{s.t.} \quad g(x,\tilde{y}) \leq 0, \tag{6b}$$
$$h(x,\tilde{y}) = 0. \tag{6c}$$

Note that for some $\tilde{y} \in Y$, $\mathcal{P}_{\text{NLP}}(\tilde{y})$ might be infeasible, in this case we set $J(\tilde{y}) = +\infty$.

Assumption 3. *A constraint qualification holds at the solution $x^\star$ of $\mathcal{P}_{\text{NLP}}(\tilde{y})$ for all $\tilde{y} \in Y$.*

Assuming $x^\star \in X$ to be a feasible and optimal solution of $\mathcal{P}_{\text{NLP}}(\tilde{y})$ and Assumption 3 holds, it is possible to compute the subgradient ∇J at $\tilde{y} \in Y$ from the Karush-Kuhn-Tucker (KKT) conditions of the following problem

$$\begin{aligned} \min_{x \in X, y \in Y} \quad & f(x,y) \\ \text{s.t.} \quad & g(x,y) \leq 0, \\ & h(x,y) = 0, \\ & y - \tilde{y} = 0, \end{aligned} \tag{7}$$

that are given by

$$\begin{cases} \nabla_x f(x^\star,\tilde{y}) + \nabla_x g(x^\star,\tilde{y})\lambda + \nabla_x h(x^\star,\tilde{y})\mu & = 0, \\ \nabla_y f(x^\star,\tilde{y}) + \nabla_y g(x^\star,\tilde{y})\lambda + \nabla_y h(x^\star,\tilde{y})\mu & = -\mu_{\tilde{y}}, \\ g(x^\star,\tilde{y}) & \leq 0, \\ h(x^\star,\tilde{y}) & = 0, \\ \lambda & \geq 0, \\ \lambda_i g_i(x^\star,\tilde{y}) & = 0, \;\; i = 1,\ldots,n_g \end{cases} \tag{8}$$

with $\lambda \in \mathbb{R}^{n_g}, \mu \in \mathbb{R}^{n_h}$ being the Lagrangian multipliers of the constraints associated with functions g, h, respectively, and $\mu_{\tilde{y}} \in \mathbb{R}^{n_y}$ is the multiplier associated to the constraint $y - \tilde{y} = 0$. Finally, the subgradient function J at $\tilde{y}$ with respect to the integer variable $y \in Y$ is given by

$$\nabla J(\tilde{y}) := -\mu_{\tilde{y}}. \tag{9}$$

Remark 2 We use the term "subgradient" because the function J is potentially nonsmooth. Moreover, since we solve (6) with an interior point solver, the computed subgradient $\nabla J(\tilde{y})$ might be an approximation of the true one. In practice, this approximation has not posed significant issues.

2.2.1 Feasibility of the auxiliary problem

For some $\hat{y} \in Y$, the problem $\mathcal{P}_{\text{NLP}}$ which we aim to solve might be infeasible. In this specific situation, we solve a feasibility problem to find the closest point to $\hat{y}$ that lies on the boundary of $\bar{\mathcal{F}}$. The sought point is obtained by solving the following NLP

$$\mathcal{P}_{\text{FNLP}}(\hat{y}, y_b): \quad \begin{aligned} \bar{y}(\hat{y}, y_b) \coloneqq \arg\min_{y \in \bar{Y}} \quad & J_{\text{f}}(y; \hat{y}) \\ \text{s.t.} \quad & \|y_b - y\|_2^2 \leq \|y_b - \hat{y}\|_2^2, \quad \text{if } y_b \text{ given,} \end{aligned} \tag{10}$$

with J_{f} being

$$J_{\text{f}}(y; \hat{y}) \coloneqq \begin{aligned} \min_{x \in X} \quad & \|y - \hat{y}\|_2^2 \\ \text{s.t.} \quad & g(x, y) \leq 0, \\ & h(x, y) = 0. \end{aligned} \tag{11}$$

The constraint in (10) is enforced only if a feasible (best) solution $y_b \in \mathcal{F}$ is available. This constraint narrows the feasible set $\bar{Y}$ by requiring that $\bar{y}$ lies within a ball of radius $\|y_b - \hat{y}\|_2$ around the best point y_b. This requirement is particularly helpful in case of a disconnected feasible set $\bar{\mathcal{F}}$, but it does not introduce further complexity in case of a convex feasible set $\bar{\mathcal{F}}$ (cf. Lemma 1).

Note that the above problems have a bi-level structure but can be written as one minimization problem in the joint space of $X \times \bar{Y}$. Here, we are only interested in the minimizer of problem (10) which is used to construct the following infeasibility cut:

$$(\hat{y} - \bar{y})^\top (y - \bar{y}) \leq 0. \tag{12}$$

This constraint can be interpreted as the hyperplane on a convex set going through $\bar{y}$ with a normal vector $\hat{y} - \bar{y}$. Note that, similarly to J, also J_{f} can denote either an optimization problem, a function, or a specific objective value. The usage should be clear from the context.

Lemma 1 *Under Assumption 1, if $\mathcal{P}_{\text{NLP}}(\hat{y})$ is infeasible, and if $\bar{y}$ solves $\mathcal{P}_{\text{FNLP}}(\hat{y}, y_b)$, then $y = \hat{y}$ is infeasible in the constraint* (12)*.*

Proof. For $y = \hat{y}$, we have $(\hat{y} - \bar{y})^\top (\hat{y} - \bar{y}) = \|\hat{y} - \bar{y}\|_2^2 > 0$, thus $\hat{y}$ violates (12). Conversely, if $\mathcal{P}_{\text{NLP}}(\hat{y})$ is feasible, the solution of $\mathcal{P}_{\text{FNLP}}(\hat{y}, y_b)$ is $\bar{y} = \hat{y}$, and (12) holds with equality. □

Lemma 2 *Under Assumption 2, if $\mathcal{P}_{\text{NLP}}(\hat{y})$ is infeasible, and if $\bar{y}$ solves $\mathcal{P}_{\text{FNLP}}(\hat{y}, y_b)$, then all $y \in \bar{\mathcal{F}}$ are feasible in the constraint* (12)*.*

Proof. The point $\bar{y}$ is a boundary point of $\bar{\mathcal{F}}$, since $\hat{y} \notin \bar{\mathcal{F}}$ and $\bar{y} = \arg\min_{y \in \bar{\mathcal{F}}} \|y - \hat{y}\|_2^2$. Based on the separating hyperplane theorem [9], there exists a separating hyperplane given by

$$\exists\, a, \alpha : a^\top \hat{y} > \alpha \text{ and } a^\top y \leq \alpha, \forall y \in \bar{\mathcal{F}} \tag{13}$$

with a being normal to the set $\bar{\mathcal{F}}$, such that

$$a^\top y \leq a^\top \bar{y}, \quad \forall y \in \bar{\mathcal{F}}. \tag{14}$$

Since by definition $\bar{y}$ is the closest point to $\hat{y}$ and $\bar{y} \in \bar{\mathcal{F}}$, then $a = (\hat{y} - \bar{y})$ defines the normal vector to the set $\bar{\mathcal{F}}$. Substituting this into (14) leads to:

$$(\hat{y} - \bar{y})^\top (y - \bar{y}) \leq 0, \quad \forall y \in \bar{\mathcal{F}}.$$

□

Based on Lemma 2, we can conclude that the constraint (12) is inactive for the best found solution y_b, if the problem is convex.

2.3 The lower bound function

Let us define the lower bound function J_{LB} introduced earlier. Consider a collection of points for which first-order information of J is available in terms of evaluation tuples $(i, x_i, y_i, J(y_i), \nabla J(y_i))$. We collect them in a dataset

$$\mathcal{D}_k \coloneqq ((0, x_0, y_0, J(y_0), \nabla J(y_0)), \ldots, (k, x_k, y_k, J(y_k), \nabla J(y_k))) .$$

If, at the k-th iteration, we have successfully solved $\mathcal{P}_{\text{NLP}}$ with $J(y_k)$, then we store k in the index set $\mathbb{T}_k$, which collects the indices of feasible iterates. Otherwise, if we have solved the feasibility problem $\mathcal{P}_{\text{FNLP}}(y_k, y_{b(k)})$, we store its solution $\bar{y}_k$ by extending the corresponding tuple, and we store k in the index set $\mathbb{S}_k$. The following set expression holds: $\mathbb{T}_k \cup \mathbb{S}_k = \mathbb{Z}_{[0,k]}$. Moreover, at the k-th iteration, we identify the index corresponding to the best solution found as

$$b(k) \coloneqq \begin{cases} \arg\min\limits_{i \in \mathbb{T}_k} J(y_i), & \text{if } \mathbb{T}_k \neq \emptyset, \\ \arg\min\limits_{i \in \mathbb{S}_k} J_{\text{f}}(\bar{y}_i; y_i), & \text{otherwise.} \end{cases} \tag{15}$$

Clearly, $b(k) \leq k$. With the above definition of $b(k)$, we can not only keep track of the feasible point with the minimum objective, but also of the least infeasible point.

The lower bound function J_{LB} is defined on these index sets as follows

$$\begin{aligned} J_{\text{LB}}(y; \mathbb{T}_k, \mathbb{S}_k, \mathcal{D}_k) \coloneqq \min_{\eta} \quad & \eta \\ \text{s.t.} \quad & \eta \geq J(y_i) + \nabla J(y_i)^\top (y - y_i), \quad i \in \mathbb{T}_k, \\ & (y_j - \bar{y}_j)^\top (y - \bar{y}_j) \leq 0, \qquad\quad j \in \mathbb{S}_k. \end{aligned} \tag{16}$$

The term $\nabla J(y_i)$ is a subgradient of the potentially nonsmooth function J, and its computation has been discussed in Section 2.2.

2.4 Benders region computation and MIQP

As stated in Algorithm 1, problem $\mathcal{P}_{\text{BR}-\text{MIQP}}$ depends on the convex polyhedron $\mathbb{B}_k$, referred to "Benders region". It is defined as

$$\mathbb{B}_k \coloneqq \{ y \in \mathbb{R}^{n_y} \mid J_{\text{LB}}(y; \mathbb{T}_k, \mathbb{S}_k, \mathcal{D}_k) \leq \bar{J}_k \}, \tag{17}$$

where $\bar{J}_k$ is determined by

$$\bar{J}_k \coloneqq \alpha J(y_{b(k)}) + (1 - \alpha)\text{LB}, \tag{18}$$

where $\alpha \in [0, 1)$. The interval is open on the right ensuring that the current best point is always excluded from (17). The computation of this reduced right-hand-side term follows an idea from [36]. It has the property to exclude all the visited points including the best point $y_{b(k)}$ from the current region $\mathbb{B}_k$, unless the lower-bound value is equal to the current best point. As the lower bound LB is not updated during each iteration, this constraint might make $\mathcal{P}_{\text{BR}-\text{MIQP}}$ infeasible. When this happens, we trigger the if-condition of line 22 and solve a different master problem, named $\mathcal{P}_{\text{LB}-\text{MILP}}$, which is presented in the next section.

Finally, the Benders-region MIQP $\mathcal{P}_{\text{BR}-\text{MIQP}}$ comprises the quadratic approximation of $\mathcal{P}_{\text{MINLP}}$ as presented in (4), computed around the current best point $(x_{b(k)}, y_{b(k)})$ with the

Hessian approximation $B_{b(k)}$, the additional constraints created by the Benders region (17), and the infeasibility cuts (12). Hence, we state

$\mathcal{P}_{\text{BR-MIQP}}$:

$$\begin{aligned} \min_{x \in X, y \in Y} \quad & f_{\text{L}}(x,y;x_{b(k)},y_{b(k)}) + \frac{1}{2}\begin{pmatrix} x-x_{b(k)} \\ y-y_{b(k)} \end{pmatrix}^{\top} B_{b(k)} \begin{pmatrix} x-x_{b(k)} \\ y-y_{b(k)} \end{pmatrix} \\ \text{s.t.} \quad & g_{\text{L}}(x,y;x_{b(k)},y_{b(k)}) \leq 0, \\ & h_{\text{L}}(x,y;x_{b(k)},y_{b(k)}) = 0, \\ & J(y_i) + \nabla J(y_i)^{\top}(y-y_i) \leq \bar{J}_k, \quad i \in \mathbb{T}_k, \\ & (y_i - \bar{y}_i)^{\top}(y-\bar{y}_i) \leq 0, \qquad\quad i \in \mathbb{S}_k. \end{aligned} \tag{19}$$

We denote the objective value of (19) as V_{MIQP}. Eventually, in the case no feasible solution has been found yet, i.e., $\mathbb{T}_k = \emptyset$, the functions $f_{\text{L}}, g_{\text{L}}, h_{\text{L}}$ are obtained by linearizing at the current least infeasible point $(x_{b(k)}, y_{b(k)})$, where $b(k)$ is computed according to (15).

2.5 Termination certificates

As previously mentioned, problem $\mathcal{P}_{\text{BR-MIQP}}$ might be infeasible since the Benders region might be an empty set. In this case, we do not know whether the current best solution is an optimal solution for $\mathcal{P}_{\text{MINLP}}$, even in the case the latter is convex (under Assumption 2), because the solution of $\mathcal{P}_{\text{BR-MIQP}}$ does not provide a lower bound of $\mathcal{P}_{\text{MINLP}}$. Therefore, we introduce a strategy to verify whether the current best solution is optimal or other solutions exist.

The strategy consists of solving an MILP with a structure similar to the main problem of the GBD algorithm [26]. In our case, we additionally include in the constraints the linear approximation of the original MINLP (1) around the best solution found. This can be seen as a mix between GBD and outer approximation [23]. At the k-th iteration of Algorithm 1, $\mathcal{P}_{\text{LB-MILP}}$ can be formulated as a MILP in the full variable space as follows

$\mathcal{P}_{\text{LB-MILP}}$:

$$\begin{aligned} \min_{\eta \in \mathbb{R},\, x \in X,\, y \in Y} \quad & \eta && && \text{(20a)} \\ \text{s.t.} \quad & \eta \geq f_{\text{L}}(x,y;x_{b(k)},y_{b(k)}), && && \text{(20b)} \\ & 0 \geq g_{\text{L}}(x,y;x_{b(k)},y_{b(k)}), && \text{if } b(k) \in \mathbb{T}_k, && \text{(20c)} \\ & 0 = h_{\text{L}}(x,y;x_{b(k)},y_{b(k)}), && \text{if } b(k) \in \mathbb{T}_k,, && \text{(20d)} \\ & \eta \geq J(y_i) + \nabla J(y_i)^{\top}(y-y_i), && i \in \mathbb{T}_k, && \text{(20e)} \\ & 0 \geq (y_i - \bar{y}_i)^{\top}(y-\bar{y}_i), && i \in \mathbb{S}_k. && \text{(20f)} \end{aligned}$$

Differently from (19) the outer approximation constraints are only imposed for the current best solution, which corresponds to the linearization point, but only if the pair $(x_{b(k)}, y_{b(k)})$ is feasible, i.e., $b(k) \in \mathbb{T}_k$.

Problem (20) entails all the visited integer solutions. Specifically, for each feasible visited solution y_i we construct a linear approximator in the integer space of the nonlinear

function $J : Y \mapsto \mathbb{R}$, cf., (3). Thus, by minimizing the slack variable η, we aim to find the minimum of the epigraph of the function obtained by the intersection of all the epigraphs of linear models.

2.6 Algorithm properties

In the following we state a few properties of Algorithm 1, using the following well-known results.

Lemma 3 *A differentiable function $f : X \to \mathbb{R}$ is convex if and only if X is a convex set and $f(z) \geq f(x) + \nabla f(x)^\top (z - x)$ holds for all $x, z \in X$.*

Proof. Given in [9, §3.1.3]. □

Lemma 4 *If Assumptions 1, 2, and 3 hold, then $\mathcal{P}_{\text{LB}-\text{MILP}}$ has the same minimizer as $\mathcal{P}_{\text{MINLP}}$, given all feasible integer points are visited, $Y = \{y_k \in \mathbb{T}_k \mid k \to \infty\}$.*

Proof. Proved in [38, §13.1] for GBD, here the only difference is that the master problem $\mathcal{P}_{\text{LB}-\text{MILP}}$ includes the linearization of constraints about $(x_{b(k)}, y_{b(k)})$. Since functions f, g, h are differentiable and convex, the first order Taylor series $f_{\text{L}}, g_{\text{L}}, h_{\text{L}}$ provide lower bounds for the corresponding functions (cf. Lemma 3). Hence, the feasible set of $\mathcal{P}_{\text{LB}-\text{MILP}}$ is still an overapproximation of the feasible set of $\mathcal{P}_{\text{MINLP}}$ that contains the global minimum. Note that the feasible set of $\mathcal{P}_{\text{LB}-\text{MILP}}$ is tighter compared to the one of the master problem in GBD. □

Theorem 1 *If Assumptions 1, 2, and 3 hold, Algorithm 1 converges in a finite number of iterations to the global optimal solution of MINLP or with a certificate of infeasibility.*

Proof. First we need to demonstrate that no integer assignment is repeated by Algorithm 1. The finiteness of Algorithm 1 follows from the finiteness of the set Y.

An integer assignment y_k can make $\mathcal{P}_{\text{NLP}}$ with $J(y_k)$ feasible or infeasible. If $\mathcal{P}_{\text{NLP}}(y_k)$ is infeasible, we solve $\mathcal{P}_{\text{FNLP}}(y_k, y_{b(k)})$ and we add the constraint

$$(y_k - \bar{y}_k)^\top (y - \bar{y}_k) \leq 0,$$

in the next master problems $\mathcal{P}_{\text{BR}-\text{MIQP}}$ and $\mathcal{P}_{\text{LB}-\text{MILP}}$, so as to exclude y_k from their feasible region (cf. Lemma 1).

If $\mathcal{P}_{\text{NLP}}$ is feasible and the objective is not equal to LB, y_k will be excluded in the $\mathcal{P}_{\text{BR}-\text{MIQP}}$ by the Benders region constraint. Hence, solving $\mathcal{P}_{\text{BR}-\text{MIQP}}$ would either result in a new integer solution $\bar{y}$ or in a certificate of infeasibility. For the latter case, we would then solve $\mathcal{P}_{\text{LB}-\text{MILP}}$, for which y_k is feasible and provides a new linearization point for a supporting hyperplane that approximates from below the solution of $\mathcal{P}_{\text{MINLP}}$ (cf. Lemma 3). If $\mathcal{P}_{\text{NLP}}$ is feasible and the objective is equal to LB, the optimal value is also reached and Algorithm 1 terminates.

Algorithm 1 always terminates at the global optimal solution of $\mathcal{P}_{\text{MINLP}}$ or with a certificate of infeasibility for $\mathcal{P}_{\text{MINLP}}$.

Let $(x^\star, y^\star)$ be the global optimal solution of $\mathcal{P}_{\text{MINLP}}$ with objective $f^\star$, hence $\mathcal{P}_{\text{NLP}}$ is feasible for $J(y^\star)$ and $\text{UB} = J(y^\star) = f(x^\star, y^\star)$. Assume that when solving $\mathcal{P}_{\text{LB}-\text{MILP}}$, we obtain a solution $(\eta^\star, \tilde{x}, y^\star)$ for which constraints (20c), (20d), (20f) are satisfied. Conversely,

we regard the remaining constraints of $\mathcal{P}_{\text{LB}-\text{MILP}}$

$$\eta \geq f(x^\star,y^\star) + \nabla f(x^\star,y^\star)^\top \begin{pmatrix} \tilde{x}-x^\star \\ y^\star - y^\star \end{pmatrix} \tag{21}$$

$$\eta \geq J(y_i) + \nabla J(y_i)^\top (y^\star - y_i),\ i \in \mathbb{T}_k. \tag{22}$$

From the first equation, we know that no valid descent direction exists along x at $x^\star$ (cf. Assumption 3), therefore

$$\nabla f(x^\star,y^\star)^\top \begin{pmatrix} \tilde{x}-x^\star \\ 0 \end{pmatrix} \geq 0.$$

Additionally, the right hand side of Eq. (22) consists of lower bounds for the solution of $\mathcal{P}_{\text{MINLP}}$ and $J(y_i) > \text{UB}$ for all $i \in \mathbb{T}_k$, it follows

$$\text{LB} = \eta \geq f(x^\star,y^\star) + \nabla f(x^\star,y^\star)^\top \begin{pmatrix} \tilde{x}-x^\star \\ 0 \end{pmatrix} \geq \text{UB},$$

thus Algorithm 1 terminates.

Assume that Algorithm 1 terminates at a point (x',y') which is not the global optimum, hence $\text{UB} = J(y') > J(y^\star)$, and the solution of $\mathcal{P}_{\text{LB}-\text{MILP}}$ is $\eta' = \text{LB}$. To attain termination, we need $\text{LB} \geq \text{UB}$, it follows that $\text{LB} = \eta' \geq \text{UB} > J(y^\star)$, which contradicts Lemma 3. In fact, the problem $\mathcal{P}_{\text{LB}-\text{MILP}}$ is an underapproximator of $\mathcal{P}_{\text{MINLP}}$, thus the optimal solution $(x^\star,y^\star)$ must be feasible for $\mathcal{P}_{\text{LB}-\text{MILP}}$ and yield a $\eta^\star < \eta'$.

Algorithm 1 classifies $\mathcal{P}_{\text{MINLP}}$ as infeasible when every $y \in Y$ has proved to make $\mathcal{P}_{\text{NLP}}$ infeasible. Algorithm 1 never updates UB, then $\text{UB} = +\infty$. Moreover, once every $y \in Y$ has been tested, the infeasibility cutting planes make $\mathcal{P}_{\text{LB}-\text{MILP}}$ infeasible as well. Therefore, Algorithm 1 terminates since we have $\text{LB} = \text{UB} = +\infty$. □

2.7 Tutorial example for S-B-MIQP

We illustrate the behavior of Algorithm 1 for a MINLP where we can have an effective graphical representation. The example is taken from [27]. Consider the following convex MINLP

$$\min_{\substack{x\in\mathbb{R},\\ (y_1,y_2)\in\mathbb{Z}^2}} \quad (y_1-4.1)^2 + (y_2-4.0)^2 + \lambda x \tag{23a}$$

$$\text{s.t.} \quad y_1^2 + y_2^2 - r^2 - x \leq 0, \tag{23b}$$

$$-x \leq 0, \tag{23c}$$

where $r = 3$ and $\lambda = 1000$. The term λx in (23a) can be seen as a penalization of the violation of some quadratic constraint $y_1^2 + y_2^2 - r^2 \leq 0$. The shape of the cost function and the global minimizer of (23) are represented in Figure 1. The global minimizer can be found graphically and corresponds to $(x,y_1,y_2,x) = (0,2,2)$. Note that x is determined by $x = \max(0, y_1^2 + y_2^2 - r^2)$. Given a linearization point $(\bar{x},\bar{y}_1,\bar{y}_2)$, one can compute the MILP/MIQP approximation as explained in Section 1.4. Since the original cost is convex and quadratic, one can choose B as the exact Hessian of (23a). The quadratic constraint (23b) is linearized to fit the MILP/MIQP approximation as follows

$$-\bar{y}_1^2 - \bar{y}_2^2 + 2y_1\bar{y}_1 + 2y_2\bar{y}_2 - x \leq r^2. \tag{24}$$

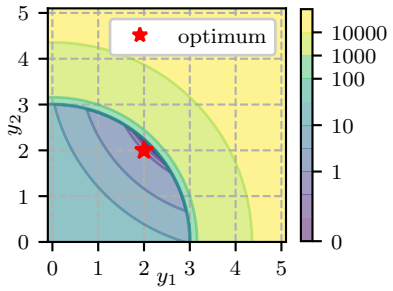

Fig. 1: Level lines of the cost function of problem (23). The red asterisk denotes the global minimizer.

Table 1: Iterations of Algorithm 1 for problem (23)

k	LB	UB	$b(k)$	y_k	$J(y_k)$	V_k
0	7.44	7016.81	0	(0, 4)	7016.81	-
1	7.44	7016.81	0	(4, 3)	16001.01	1.01
2	7.44	4005.21	2	(3, 2)	4005.21	5.21
3	8.41	8.41	3	(2, 2)	8.41	8.41

Also, for the MILP/MIQP approximation x is implicitly defined as $x = \max(0, -\bar{y}_1^2 - \bar{y}_2^2 + 2y_1\bar{y}_1 + 2y_2\bar{y}_2 - r^2)$. For this reason, in the following, we focus only on the values of y_1, y_2. By a slight abuse of notation, we stack y_1, y_2 in the vector y_k as $y_k := (y_1, y_2)$ where the subscript k denotes the iteration number of Algorithm 1.

We apply Algorithm 1 to solve (23), with hyper-parameter $\alpha = 0.9$ for the reduced right-hand-side (18). Table 1 reports the results of each iteration, and Figure 2 gives a graphical interpretation.

The problem is initialized with $y_0 = (0,4)$, and the lower bound computed via MINLP relaxation is LB $= 7.44$. In the first iteration, the initial point y_0 is evaluated by solving $\mathcal{P}_{\text{NLP}}(y_0)$. Since the problem is feasible, its objective value updates the initial upper bound, UB $= 7016.81$. Then, we compute the Benders region $\mathbb{B}_0$ and we solve $\mathcal{P}_{\text{BR-MIQP}}$ with $J_{\text{QP}}(y; x_0, y_0, B_0)$. Since $\mathcal{P}_{\text{BR-MIQP}}$ is feasible, we store its objective in V_1 and its solution in y_1, hence $V_1 = 1.01$ and $y_1 = (4,3)$. The next two iterations of Algorithm 1, i.e., $k = 1,2$ has a similar structure to the first iteration. We move our focus to the fourth iteration where $y_3 = (2,2)$, and the associated $\mathcal{P}_{\text{NLP}}(y_3)$ objective is 8.41, then UB is updated. Also, the Benders region $\mathbb{B}_3$ is computed, and its intersection with the integer feasible set is empty. Therefore, we switch to the solution of the associated $\mathcal{P}_{\text{LB-MILP}}$, whose minimizer is $y_2 = (2,2)$, and the objective is $V_{\text{MILP}} = 8.41$. The lower bound LB is updated and, finally, the termination condition of Algorithm 1 is met since UB $=$ LB.

Note that the sequential nature of Algorithm 1 is meaningful only thanks to the update of the Benders region based on the information gathered at each iteration. Without this update, Algorithm 1 would cycle eternally between the first two integer solutions.

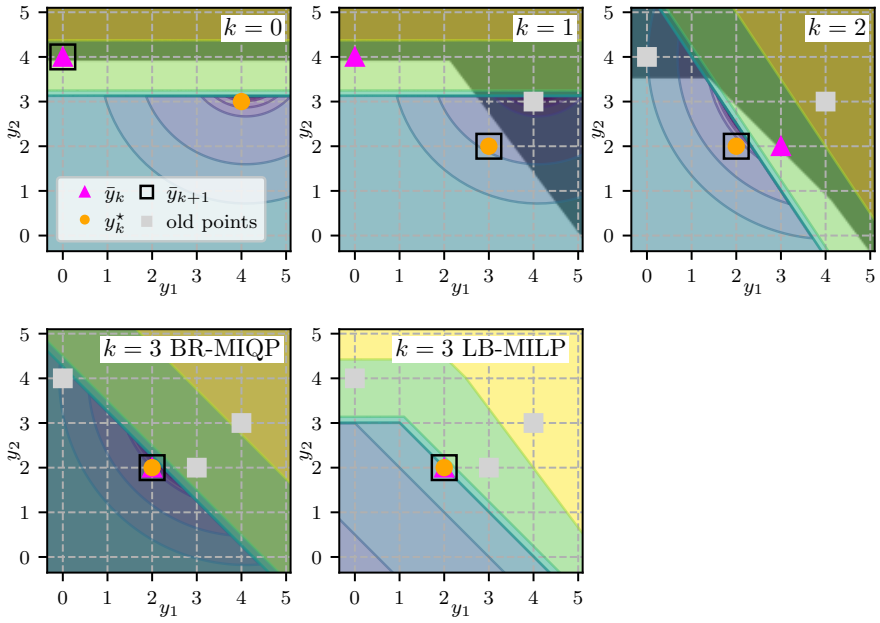

Fig. 2: Representation of algorithm iterations. The level lines correspond to either $\mathcal{P}_{\mathrm{BR-MIQP}}$ or $\mathcal{P}_{\mathrm{LB-MILP}}$, according to the iteration. The darkened areas describe the regions excluded by the Benders regions $\mathbb{B}_k$.

3 Extension to the nonconvex case

Under the convexity Assumption 2, Theorem 1 guarantees final termination of Algorithm 1. However, when we deal with nonconvex MINLPs (1), for which we only require Assumption 1 and 3 to hold, the termination condition of Algorithm 1 might be triggered in wrong situations. Due to nonconvexities, the cutting planes based on J_{LB} are not guaranteed to be lower bounds for the global optimum or for the current best solution.

We delineate two premature termination scenarios. The first scenario occurs when the solution of $\mathcal{P}_{\mathrm{LB-MILP}}$ yields a new point (x_k, y_k) with a value exceeding the current UB. This situation is inherently ambiguous because we halt the algorithm due to a newfound minimizer that we acknowledge to be inferior to the best point found during the algorithm's iterations. The second scenario arises from infeasible cutting planes, causing $\mathcal{P}_{\mathrm{LB-MILP}}$ to become infeasible. Here, we set LB to $+\infty$, triggering Algorithm 1 termination. This second scenario is particularly misleading as it encroaches upon the condition $\mathrm{LB} = +\infty$, typically reserved for detecting infeasibility in the original problem $\mathcal{P}_{\mathrm{MINLP}}$.

We illustrate the first situation of premature termination with the following example. Consider the nonlinear integer program

$$\min_{y \in \mathbb{Z}_{[-4,4]}} \quad \left(y^2 - 5\right)^2 + 4y. \tag{25}$$

For simplicity, we run Algorithm 1 with zero Hessian $B = 0$ in $\mathcal{P}_{\mathrm{BR-MIQP}}$, resulting in an MILP. As shown in Figure 3, at the last iteration, the new Benders region $\mathbb{B}_3$ is empty. Hence, we attempt to solve $\mathcal{P}_{\mathrm{LB-MILP}}$. Note that for the latter problem, the cutting planes

are not lower bounds of the current best solution $y_{b(3)} = -3$. The solution of $\mathcal{P}_{\text{LB-MILP}}$ is $y = 2$, which has a value greater than UB, triggering the termination of Algorithm 1.

The second scenario is illustrated in Figure 4, where $\bar{\mathcal{F}}$ is nonconvex, $y_b \in \bar{\mathcal{F}}$ is the best solution found, and $\hat{y}$ is the infeasible solution that Algorithm 1 has computed lastly. In the attempt to create an infeasibility cut as (12) for $\mathcal{P}_{\text{FNLP}}(\hat{y}, y_b)$, we cut a portion of the feasible set that includes y_b. This makes LB $= \infty$, triggering the termination of Algorithm 1.

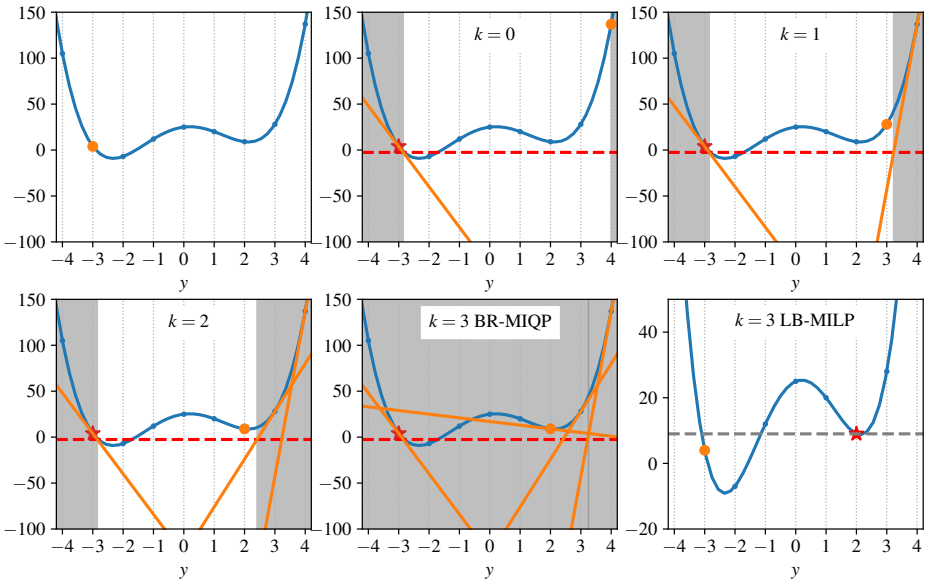

Fig. 3: Iterations of Algorithm 1 for Problem (25) with LB equal to the global minimum and $\alpha = 0.5$, starting from $y_0 = -3$. In blue, the cost function; in orange, the linear models; the red star is the current best point, and the orange dot the solution computed at the given iteration. The shaded dark areas correspond to the area excluded from the feasible set by the given Benders region. The first plot depicts the initial condition of Algorithm 1, and the last plot illustrates its termination, highlighting the first scenario of premature termination.

In what follows, we introduce a strategy to address these situations of early termination and enforce Algorithm 1 to not discard the current best solution. First, we present a procedure to correct the gradient of the linear models, then we show a way to enlarge the Benders region.

3.1 Gradient correction

During the computation of J_{LB} at the k-th iteration of Algorithm 1, it might happen that the cutting planes exclude the current best solution $y_{b(k)}$, eventually resulting in an early termination. To prevent this situation, we correct the gradients used in J_{LB} to ensure that the resulting linearizations are lower bounds for the best point as follows

$$J(y_i) + \nabla J(y_i)^\top (y - y_i) \leq J(y_{b(k)}), \text{ for all } i \in \mathbb{T}_k. \tag{26}$$

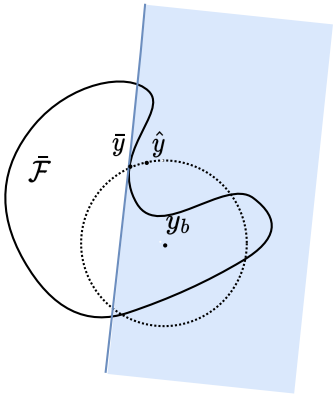

Fig. 4: Termination condition triggered by infeasibility cuts. The relaxed integer feasible set $\bar{\mathcal{F}}$ corresponds to the interior of the solid black line, $y_b \in \mathcal{F}$ is the current best and feasible solution, $\hat{y} \in \mathbb{Z}^{n_y}$ is infeasible, and $\bar{y}$ is the solution of (10). The infeasibility cut in $\bar{y}$ makes y_b infeasible and triggers the termination condition.

But, if for $y = y_{b(k)}$, the inequality does not hold for some index i, we have

$$J(y_i) + \nabla J(y_i)^\top (y_{b(k)} - y_i) > J(y_{b(k)}). \tag{27}$$

To address this issue, we focus on modifying the gradient of the linear model ∇J and define the set of "admissible" gradients $\mathbb{G}_{(i,k)}$ such that

$$\mathbb{G}_{(i,k)} := \{\tilde{g} \mid J(y_i) + \tilde{g}^\top (y_{b(k)} - y_i) \leq J(y_{b(k)})\}. \tag{28}$$

Among the elements of this set, we choose the gradient that allows for the minimal correction in a weighted Euclidean norm, with a symmetric positive definite weight matrix W. Thus, the corrected gradient is defined as

$$g^{\text{corr}}_{(i,k)} := \underset{g \in \mathbb{G}_{(i,k)}}{\arg\min} \quad \frac{1}{2}\|g - \nabla J(y_i)\|^2_W \tag{29}$$

Lemma 5 *Problem* (29) *can be solved analytically. With the shorthands* $\Delta y_{(i,k)} = y_{b(k)} - y_i$ *and* $r_{(i,k)} = J(y_{b(k)}) - J(y_i) - \nabla J(y_i)\Delta y_{(i,k)}$ *it results in*

$$g^{\text{corr}}_{(i,k)} = \nabla J(y_i) + \begin{cases} 0 & \text{if } r_{(i,k)} \geq 0, \\ \dfrac{r_{(i,k)}}{\Delta y^\top_{(i,k)} W^{-1} \Delta y_{(i,k)}} W^{-1} \Delta y_{(i,k)} & \text{if } r_{(i,k)} < 0. \end{cases} \tag{30}$$

Proof. Let us write problem (29) in the following form

$$\begin{aligned} \min_{\Delta g} \quad & \frac{1}{2}\|\Delta g\|^2_W \\ \text{s.t.} \quad & \Delta g^\top \Delta y - r = 0, \end{aligned} \tag{31}$$

where $\Delta g = g - \nabla J(y_k)$, $\Delta y = y_{b(k)} - y_k$, and $r = J(y_{b(k)}) - J(y_k) - \nabla J(y_k)\Delta y_k$. The Lagrangian function of the problem is given by

$$\mathcal{L}(\Delta g, \lambda) = \frac{1}{2}\|\Delta g\|_W^2 + (\Delta g^\top \Delta y - r)\lambda, \tag{32}$$

and its gradient

$$\nabla\mathcal{L}(\Delta g, \lambda) = W\Delta g + \lambda\Delta y. \tag{33}$$

The KKT system of the regarded optimization problem is given by

$$\begin{cases} W\Delta g + \lambda\Delta y & = 0, \\ \Delta g^\top \Delta y & = r. \end{cases} \tag{34}$$

By substitution, we can find

$$\Delta g = \begin{cases} W^{-1}\Delta y_k \cdot \frac{J(y_{b(k)}) - J(y_k) - \nabla J(y_k)\Delta y_k}{\Delta y_k^\top W^{-1}\Delta y_k} & \text{if } J(y_{b(k)}) - J(y_k) - \nabla J(y_k)\Delta y_k < 0, \\ 0 & \text{else.} \end{cases} \tag{35}$$

□

When the best point does not change in the current k-th iteration, we only need to check if the new point y_k verifies inequality (26). If necessary, we correct its gradient. However, when y_k is the new best point, we must verify if inequality (26) holds for all points $y_i, i \in \mathbb{T}_k \cup \mathbb{S}_k$ stored in $\mathcal{D}_k$ and correct the problematic gradients. We denote by $\mathcal{D}_k^{\text{corr}}$ the "corrected" dataset at iterate k, which contains the corrected gradients $g_{(i,k)}^{\text{corr}}$ instead of the true gradients $\nabla J(y_i)$.

Lemma 6 *Under Assumption 2, the corrected gradients equal the original ones, i.e.,* $\mathcal{D}_k^{\text{corr}} = \mathcal{D}_k$.

Proof. This directly follows from Lemma 3, since J is a convex function with convex domain Y. Hence, it holds $J(y_i) + \nabla J(y_i)^\top (y_j - y_i) \leq J(y_j)$, for any $y_i, y_j \in Y$. □

3.2 Region expansion via gradient amplification

When the gradient correction is computed in a nonconvex situation, we find the minimum correction that ensures that the best point found is lower bounded by every cut, which potentially makes such point the only one feasible for $\mathcal{P}_{\text{LB-MILP}}$. The gradient correction resolves the ambiguous termination described at the beginning of this section, but it can dramatically limit Algorithm 1 from further exploring the integer solution space. Therefore, we would like to create lower bounds for the current best point in a way that does not sacrifice convergence to the global optimum for *convex* MINLPs. For this purpose, we introduce a constant value $\rho \geq 1$, which amplifies all the gradients of the available linear model as follows:

$$g_{(i,k,\rho)}^{\text{ampl}} := \rho g_{(i,k)}^{\text{corr}}. \tag{36}$$

The amplification factor ρ is a hyper-parameter of Algorithm 1, which could, for example, be chosen offline and kept fixed at runtime. More elaborate strategies to choose ρ separately per inequality and iteration index are also possible but are beyond our interest in this work.

We denote by $\mathcal{D}_k^{\text{ampl}}$ the dataset where the corrected and amplified gradients replace the original gradients. This set is used as the modified dataset in line 16 of Algorithm 1, i.e., the final version of Algorithm 1 sets $\tilde{\mathcal{D}}_k := \mathcal{D}_k^{\text{ampl}}$.

At the end of this section, we demonstrate that incorporating gradient correction and amplification does not compromise the properties of Algorithm 1, namely, its finiteness and convergence to a global minimum under Assumption 2.

In Figure 5 we illustrate how the combination of gradient correction and gradient amplification can solve a deadlock situation caused by nonconvexity.

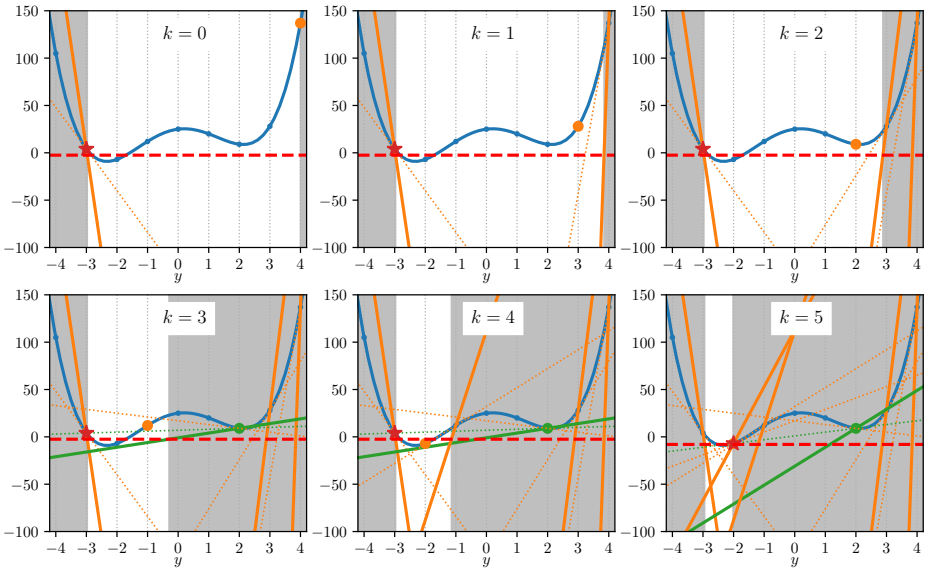

Fig. 5: We consider the same problem depicted in Figure 3 and same settings for Algorithm (1), i.e., hyper-parameter $\alpha = 0.5$. Here, we overcome the deadlock via the gradient correction and amplification, choosing $\rho = 5$. The orange dotted lines depict the linear models with the original gradient, the orange solid lines depict the linear models with gradient amplification. The linear models with gradient corrections are depicted with green lines. Specifically, the dotted lines are attained with the gradient correction only, and the solid lines with gradient correction and amplification.

3.3 Correction of the infeasibility cuts

As shown in Figure 4, the infeasibility cuts may cause early termination of Algorithm 1 by excluding the current feasible and best solution. To prevent this situation, we introduce a way to correct these cutting planes. We modify inequality (12) by introducing a correction offset $\sigma \geq 0$ as follows

$$(\hat{y} - \bar{y})^\top (y - \bar{y}) - \sigma \leq 0. \tag{37}$$

If a feasible solution $y_{b(k)}$ has been found by Algorithm 1, i.e., $\mathbb{T}_k \neq \emptyset$, it is possible to compute σ as

$$\sigma \coloneqq (\hat{y}_k - \bar{y}_k)^\top (y_{b(k)} - \bar{y}_k). \tag{38}$$

We emphasize that we compute this correction exclusively when Algorithm 1 has found at least one feasible solution $y_b \in \mathcal{F}$. In case no feasible solution is found, we enforce the infeasibility cuts as defined in (12). Figure 6 illustrates the correction of the infeasibility cut for the scenario depicted earlier in Figure 4.

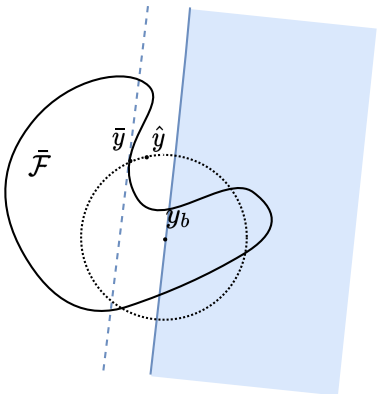

Fig. 6: Correction of the infeasibility cut of Figure 4 such that y_b remains feasible. The corrected line is depicted solid blue, while the original cut is dashed blue.

3.4 Properties of the introduced techniques

Lemma 7 *If the integer set Y is finite (Assumption 1), Algorithm 1 enhanced with gradient correction and amplification procedure for the Benders cuts, and correction for the infeasibility cuts, stops within a finite number of iterations.*

Proof. The introduced corrections maintain the property to exclude at least the respective visited points. Thus, visited points are not repeated and the finiteness of Algorithm 1 follows from Theorem 1. □

Theorem 2 *Under Assumption 2, Algorithm 1 enhanced with the gradient correction and amplification procedure for the Benders cuts, and correction for the the infeasibility cuts, stops at the global optimal solution $(x_{b(k)}, y_{b(k)})$ of problem (1) within a finite number of iterations.*

Proof. It follows from the combination of Theorem 1, Lemma 6 and Lemma 8. □

Finally, we aim to present an additional result for a class of MINLPs larger than the *convex* one.

Assumption 4. *We assume that the set $\bar{\mathcal{F}}$ is convex.*

Assumption 4 includes MINLPs that have nonconvexities in the space of continuous variables but are convex in the space of integer variables. A practical example of such case is a nonlinear control system where the discrete controls enters affinely in the dynamics.

Lemma 8 *Under Assumption 4, the correction offset for the infeasibility cuts is* $\sigma = 0$.

Proof. Similarly to Lemma 2. Since $\bar{\mathcal{F}}$ is a convex set, $\bar{y}_k$ belongs to the boundary of $\bar{\mathcal{F}}$ while $y_k \notin \bar{\mathcal{F}}$, it is possible to define a separating hyperplane such that the halfspace $\{y \mid a^\top y \leq a^\top \bar{y}_k\}$ contains $\bar{\mathcal{F}}$. The vector a is defined as $a \coloneqq (\hat{y}_k - \bar{y}_k)$. In addition, since $y_{b(k)} \in \bar{\mathcal{F}}$, inequality (12) holds for $y = y_{b(k)}$. □

Theorem 3 *If Assumptions 1, 3 and 4 hold, then Algorithm 1 terminates with a feasible solution or with a certificate of infeasibility.*

Proof. The theorem is trivial if Algorithm 1 is initialized with a feasible solution. In the other case, $b(k) \notin \mathbb{T}_k$, the constraint set for $\mathcal{P}_{\text{BR}-\text{MIQP}}$ is given by:

$$\mathbb{C}_k = \{y \in Y | (y_i - \bar{y}_i)^\top (y - \bar{y}_i) \leq 0,\ \forall k \in \{0, \dots, k\}\}$$

Note that Lemma 2 holds also under Assumption 3 only, since the infeasibility cuts operate exclusively in the integer space. Given that the set Y has finite cardinality, and for each $(y_i, \bar{y}_i) \in \{0, \dots, k\}$, Lemma 2 holds, thus $\mathcal{F} \subseteq \mathbb{C}_k$. Every visited infeasible point y_i is excluded from $\mathbb{C}_k$, therefore Algorithm 1 either terminates with a $y_k \in \mathcal{F}$, or with a certificate of infeasibility since $\mathcal{F} = \emptyset$. □

4 Numerical results

In this section, we illustrate the performance of Algorithm 1 via numerical experiments. First, we compare the proposed algorithm against two open-source solvers, Bonmin [8] and SHOT [40, 39], respectively, on a large number of MINLPs selected from the MINLPLib [16]. Secondly, we consider two instances of optimal control for a nonlinear system with binary control input. The first one is a textbook example contained in [46, §8.17], while the second one is a real-world complexity study case of a climate system for a building from [13]. Unfortunately, solving the two optimal control problems with SHOT has not been possible due to the lack of interface with CasADi. An attempt to write such interface is left for future work. All the presented results are obtained on a computer with a 12th-generation Intel Core i7-12800H processor and 32 GB of memory.

4.1 MINLPLib instances

This section compares Algorithm 1 with two existing solvers, SHOT v1.1[40, 39] and Bonmin v1.8 [8], on a subset of instances from MINLPLib [16], according to the following set of requirements:

1. Mixed-integer and mixed-boolean variables with nonlinear constraints and/or objectives. The problem should be formulated as a minimization problem.
2. Mix of convex and nonconvex instances with a known globally optimal objective.
3. The instances should have a nl-file representation.

Based on these criteria, 116 convex instances and 158 nonconvex instances were selected. Algorithm 1 is developed in Python using CasADi [3]. Besides the default stopping criterion, a heuristic stopping criterion is used. The heuristic stopping criterion is equivalent to setting the lower-bound equal to V_{MIQP}, the objective value of $\mathcal{P}_{\text{BR-MIQP}}$, and it is given by

$$\text{UB} \geq V_{\text{MIQP}}. \tag{39}$$

All solvers are configured to use the same NLP-solver, Ipopt 3.14 [54] with ma27 [30] to have a fair comparison. Both SHOT and Algorithm 1 use Gurobi 10.0.2 [29], and all solvers are configured to use one thread. For each problem, a solver time of a maximum of 300 seconds of computation time is allowed. The primal and integer tolerances are set to 10^{-6} and an absolute and relative tolerance of 10^{-2}. To avoid accounting for the inefficiency of the Python language, only the time spent in the NLP and MIQP/MILP solvers is considered. The algorithms are compared based on an objective ratio computed for every instance by

$$r_{\text{obj}} = \frac{V - \min(V_{\text{min}} - 1, 0)}{V_{\text{min}} - \min(V_{\text{min}} - 1, 0)}, \tag{40}$$

where V is the objective value of the instance computed with one given algorithm, and V_{min} is the objective value of the global optimum reported in the MINLPLib dataset. The adopted objective ratio allows the objective value to be negative or close to zero without issues in the comparison. This objective ratio is equal to the relative tolerance. A time ratio is computed for problems solved to the objective ratio below 1.01. The time ratio is the ratio of the computation time to the computation time of the fastest solver for this problem.

For the set of convex problems, Algorithm 1 finds a solution for 111 problems, while SHOT and Bonmin find 113 and 85, respectively. SHOT outperforms the other solvers also in the quality of the solutions, where it achieves an optimal solution faster and for more problems than the other solvers, as given in Figure 7. The results for the set of nonconvex problems are given in Figure 8. For this set of problems, Algorithm 1 finds a solution for 156 problems compared to 89 and 113 for SHOT and Bonmin, respectively. For 52 instances, it finds the optimal solution within the given time compared to 30 and 71 for SHOT and Bonmin, respectively. For nonconvex problems, Algorithm 1 is able to find a good solution for more instances than the other solvers but can not find the optimal solution for some of the instances. This is in line with the proofs from Section 3.

4.2 MIOC of an unstable nonlinear system

Consider a reference tracking task for a nonlinear unstable system with discrete control input. The dynamic is described by the ordinary differential equation (ODE)

$$\dot{x}(t) = x^3(t) - u(t), \quad t \in [t_0, t_{\text{f}}] \tag{41}$$

where the state is denoted by $x(t) \in \mathbb{R}$ and the control by $u(t) \in \{0, 1\}$. The control is subject to a minimum dwell time constraint of 0.1 s. The aim is to track the state reference $x_{\text{ref}} = 0.7$, starting from the initial state $\bar{x}_0 = 0.9$. By means of the multiple shooting approach for direct optimal control [7], we discretize the ODE over a fixed grid $N = 30$ shooting nodes such that $t_0 < t_1 < \ldots < t_N = t_{\text{f}}$ adopting a 4th order explicit Runge-Kutta integrator and a sampling

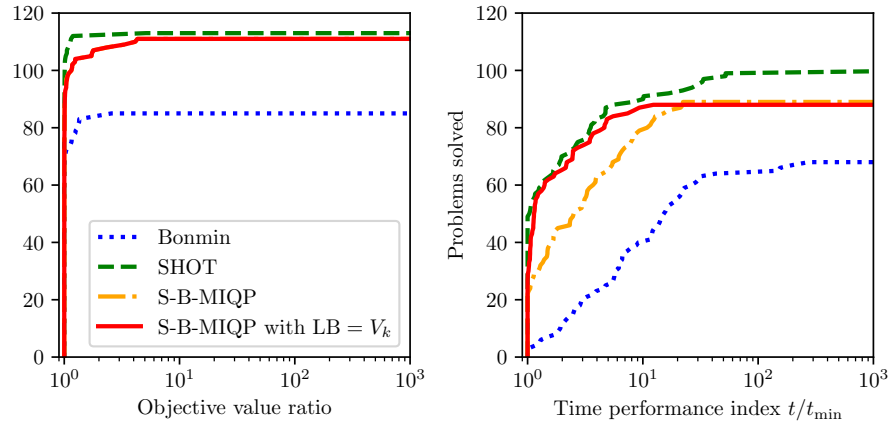

Fig. 7: Comparison of Algorithm 1 with two termination criteria against Bonmin and SHOT. Both figures share the same legend and y-axis. For this set of problems, SHOT outperforms the other solvers, but Algorithm 1 with termination criterion (39) attains very similar performance. The performance measured using the obtained objective value is almost equal for both termination criteria.
Left: Objective ratio for all solved problems, note that the green line is overlapped by the red line. Right: Time performance for the problems solved to optimality.

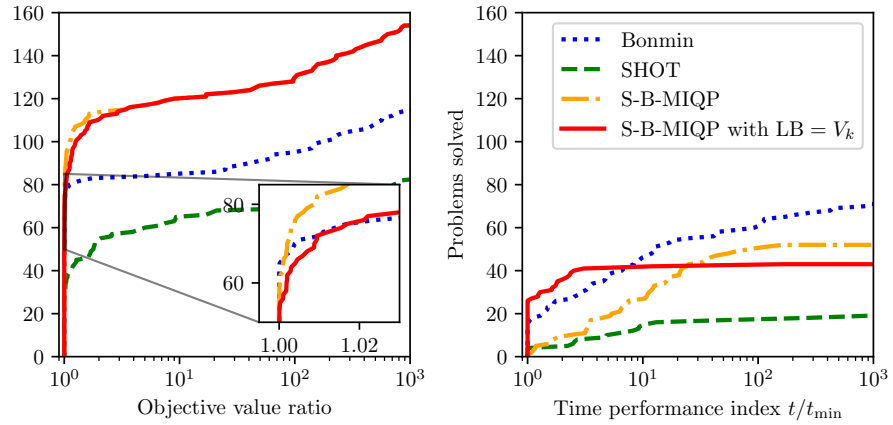

Fig. 8: Comparison of Algorithm 1 with two termination criteria against Bonmin and SHOT. Both figures share the same legend and y-axis. For this set of problems, Algorithm 1 with termination criterion (39) outperforms the other solvers on most problems. Only Bonmin achieves the optimal solution for some of the test problems.
Left: Objective ratio for all solved problems. Right: Time performance for the problems solved to optimality.

time $t_s = 0.05\,\mathrm{s}$. The resulting discretized optimal control problem is

$$\begin{aligned} \min_{\substack{x_0,u_0,\ldots,\\ u_{N-1},x_N}} \quad & \sum_{k=0}^{N} (x_k - x_{\mathrm{ref}})^2 \\ \text{s.t.} \quad & x_0 = \bar{x}_0, \\ & x_{k+1} = F_{\mathrm{RK}}(x_k, u_k), \quad k = 0,\ldots,N-1, \\ & u_k \in \mathcal{U} \qquad\qquad\quad k = 0,\ldots,N-1, \end{aligned} \tag{42}$$

where $\mathcal{U} := \{u \in \{0,1\}^{N-1} \mid u_k \geq u_{k-1} - u_{k-2},\ k = 0,\ldots,N-1\}$ imposes a minimum uptime for the control inputs of two consecutive time steps. The required previous values u_{-1}, u_{-2} are set to zero. It is possible to obtain a globally optimal solution of the MINLP (42) in a few seconds via a clever enumeration of all $2^{30} \approx 10^9$ possibilities.

We have solved problem (42) with different algorithms. The results are listed in Table 2, and Figure 9 depicts the globally optimal state and control trajectories of (42). The solvers share the same integer tolerances and MIP gap. Gurobi is used as an MIQP/MILP solver within S-B-MIQP and is constrained to run on a single thread to have a fair runtime comparison. Bonmin runs with its default nonlinear branch-and-bound routine. We did not implement a tailored branch-and-bound to globally solve the problem since we have noticed that both S-B-MIQP and Bonmin achieve the global optimum reported in [46, §8.17]. Therefore, the computation time shown for the global optimum is equivalent to the one reported for Bonmin. Algorithm 1, S-B-MIQP, is evaluated for two different initial guesses. The "good" guess is obtained by solving the first MIQP around the relaxed NLP solution with a MIP gap of 10^{-4} (default Gurobi setting), while for the "bad" guess, the MIP gap is 0.3. In all the successive iterations of S-B-MIQP the MIP gap is set to 10^{-4}. The "bad" guess is added to demonstrate that Algorithm 1 can converge to a valid solution without heavily relying on integer relaxation of MINLP. Overall, we can see that for this simple problem, the specialized algorithm CIA performs really well, yielding a solution very close to the global optimum in only 0.21 seconds. As an implementation note, the CIA problem has been solved with the tailored branch-and-bound solver pycombina [12]. Algorithm 1, which is a general purpose algorithm for solving MINLPs, computes a solution equal to the global optimum with a runtime of about 3 to 5 times higher than CIA, but about one order of magnitude faster than Bonmin.

Algorithm	Objective	Runtime [s]
Global minimum	0.1765	11.35*
Bonmin	0.1765	11.35
CIA [13]	0.1771	0.21
S-B-MIQP bad guess	0.1765	1.09
S-B-MIQP good guess	0.1765	0.64

Table 2: Comparison of different algorithms for the solution of (42).

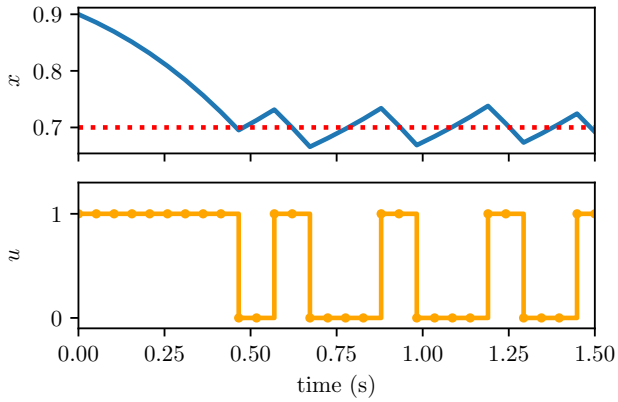

Fig. 9: Optimal state and control trajectories of (42).

4.3 MIOC of a renewable energy system

To assess the performance of Algorithm 1 on an example of real-world complexity, we consider an extended version of the solar-thermal-climate-system described in [13] and physically installed at Karlsruhe University of Applied Sciences. The system provides cooling for the building's main hall by means of thermal machines driven by solar-thermal energy. Specifically, the system consists of an absorption cooling machine (ACM) and a heat pump (HP). The first machine can be used in absorption cooling (AC) mode, during which the solar thermal heat drives the machine ($b_{\text{ac}} = 1$), or in free cooling (FC) ($b_{\text{fc}} = 1$), where the cooling tower installed on the roof of the building can directly cool down the medium at ambient temperature. The second machine is a regular HP that has been installed more recently to provide flexibility and extra cooling power. When it is switched on ($b_{\text{hp}} = 1$), it immediately provides cooling energy since the machine is connected to the low-temperature storage and driven by electric energy. The electric energy required to drive the system is either bought from the grid or supplied by on-site PV panels. A schematic of the current plant is contained in [27]. Experimental operations and numerical case studies have been carried out on this system in [10, 11, 13, 14, 27].

The system dynamics are modeled via a set of ODEs with $\xi \in \mathbb{R}^{19}$ differential states, $\mu \in \mathbb{R}^{6}$ continuous controls and $\nu \in \{0,1\}^3$ binary controls. The system state includes the temperature of the flat plate and vacuum tube solar collectors, T_{fpsc} and T_{vtsc} respectively, four temperature levels in the stratified high-temperature storage, $T_{\text{ht,i}}, i = 1,2,3,4$, the temperature of the low-temperature storage T_{lt}, and the temperature inside the primary and secondary solar circuit, T_{psc} and T_{ssc}, respectively. The continuous controls regulate the electric power absorbed from the grid, velocity and pressure of the pumps in the solar circuit, and the input/output flow rate of the high-temperature storage. As described above, there are three binary controls that decide the switching on/off of the different machines. The system is subject to several ambient conditions, represented as time-varying parameters in the NLP. These are the ambient temperature, the solar irradiance on the solar collectors, the power generated by the local PV panels, the price of electric energy, and the desired cooling load profile.

It is now possible to formulate a MIOCP that aims to provide the specified cooling power while operating the system safely and energy-efficiently. Via a direct approach, we discretize the MIOCP using Gauss-Radau collocation of order 3. The time horizon is 24 hours long and divided by a non-uniform grid with time steps of 15 minutes from 6 a.m. to 8 p.m. (UTC+2) and 30 minutes for the rest of the day, for a total of $N = 83$ steps. To guarantee feasibility of the MIOCP we have introduced $n_s = 24$ slack variables that are linearly and quadratically penalized in the cost function. Hence, the total number of variables is $N \cdot (19 \cdot d + 6 + 3 + n_s) = 7470$ of which $3 \cdot N = 249$ are binaries.

By a stage-wise concatenation of the variables, we define the vector of continuous and binary decision variables as x and y, respectively. Thus, the resulting MINLP can be stated compactly as

$$\begin{aligned} \min_{\substack{x\in\mathbb{R}^{n_x},\\ y\in\{0,1\}^{n_y}}} \quad & \|f_1(x,y)\|^2 + f_2(x,y) \\ \text{s.t.} \quad & g(x,y) \leq 0, \\ & h(x,y) = 0, \end{aligned} \tag{43}$$

where the cost function is the sum of a quadratic term, aiming to minimize constraint violation and achieve smooth actuation of the mixing valves, and a nonlinear one, defined by f_2, which incorporates the electricity cost for operating the system. The special structure of the cost function allows for positive semidefinite Hessians for the MIQPs via the Gauss-Newton approximation [42].

The MIOCP has been implemented in CasADi via its Python interface, the NLPs are solved with IPOPT [54] using HSL MA57 [30] as the internal linear solver, whereas the MIPs are solved via Gurobi v10.0.2 [29]. The integral gap for all MIPs is set to 15%, and the computation time limit is set to 900 seconds.

We consider $\mathcal{P}_{\text{BR}-\text{MIQP}}$ infeasible if no feasible solution is found within this time limit, meaning that the integral gap of the feasible solution might be larger than 15%. Conversely, we consider $\mathcal{P}_{\text{LB}-\text{MILP}}$ infeasible if, within the prescribed time limit, Gurobi cannot find a solution with an integral gap within 15%. In this case, Algorithm 1 terminates returning the best solution found.

Moreover, we use the feature *solution pool* of Gurobi with a maximum number of 3 solutions, meaning that Gurobi returns at most the three best solutions found during the solution of the MIQP/MILP. Naturally, we have to solve the corresponding $\mathcal{P}_{\text{NLP}}$ for each integral solution. By means of the *solution pool*, we obtain multiple cuts per MIP solved, making Algorithm 1 more efficient. The two hyper-parameters of Algorithm 1, α (18) and ρ (36) have been set to 0.2 and 1.5, respectively.

Figure 10 depicts Algorithm 1 performance in solving (43). In the top plot, we show the behavior of the objective of $\mathcal{P}_{\text{NLP}}$, $\mathcal{P}_{\text{BR}-\text{MIQP}}$, and $\mathcal{P}_{\text{LB}-\text{MILP}}$ for every iteration of Algorithm 1. We achieve the lowest objective of $\mathcal{P}_{\text{NLP}}$ for $J(y_{16})$, and after iteration sixteen we continue solving $\mathcal{P}_{\text{BR}-\text{MIQP}}$ but the corresponding $\mathcal{P}_{\text{NLP}}$ objective given by J is always greater than the current UB. At iteration 28, Algorithm 1 starts solving the related $\mathcal{P}_{\text{LB}-\text{MILP}}$, until termination. Note that the termination of Algorithm 1 happens at iteration 128 because the corresponding $\mathcal{P}_{\text{LB}-\text{MILP}}$ cannot yield a solution with an integral gap lower than 15%. In the bottom plot of Figure 10, we depict the cumulative runtime of Algorithm 1, detailing the wall time of every problem solved in each iteration. Up to iteration 28, the most expensive component is the solution of $\mathcal{P}_{\text{BR}-\text{MIQP}}$. However, after this point, the solution of $\mathcal{P}_{\text{LB}-\text{MILP}}$ becomes nearly as time-consuming as $\mathcal{P}_{\text{NLP}}$. Nonetheless, the solution time of the former gradually increases with each iterations due to the additional inequalities. Finally, although there are few occasions where we solve the feasibility problem $\mathcal{P}_{\text{FNLP}}$, they are in general

expensive. We attribute this to the inability of IPOPT to detect infeasibility early. Indeed, before transitioning to the feasibility problem $\mathcal{P}_{\text{FNLP}}$, a considerable amount of time is spent attempting to solve the related $\mathcal{P}_{\text{NLP}}$.

Figure 11 depicts the optimal trajectory for a selected subset of the state and for binary controls obtained by solving (43) with Algorithm 1. The solution depicts an almost optimal operation of the system, indeed the temperature bounds and predictions of the external parameters over the horizon have been exploited by the optimizer. However, since constraints are imposed in a soft way, the solution exhibits a small violation of the upper bound of the low-temperature storage. Note that the free-cooling mode is never active, meaning that the system never dissipates heat into the environment.

Table 3 presents the computation results obtained using different methods. We report the solution of the continuous relaxation, whose objective serves as a lower bound for the mixed-integer problem. Despite the problem's considerable size and nonlinearity, IPOPT takes only 9 seconds to solve it. For Bonmin we indicate "N/A" because it fails to compute even a feasible mixed-integer solution. The error encountered with Bonmin indicates that the LP relaxation is either infeasible or too computationally expensive. Further investigation reveals that Bonmin can actually solve a lower dimensional version of (43) with a horizon length set to $N = 20$. Hence, we believe that when the complexity of the MINLP grows, the nonlinear branch-and-bound method implemented by Bonmin performs poorly.

For the CIA algorithm, we quickly obtain a solution but the corresponding objective is fairly high, underlying a quite suboptimal operating strategy for the machine. However, at the current stage, CIA is the only method that can be adopted for devising a real-time controller in a receding horizon fashion.

When we enhance the CIA algorithm with a Gauss-Newton MIQP (GN-MIQP), as presented in [14], the computation time increases, but the reduction in the objective is dramatic compared to the standard CIA algorithm. The GN-MIQP approach has been implemented by constructing the GN-MIQP around the solution of the relaxed NLP. The GN-MIQP problem has been solved to a integrality gap of 5%. Finally, the sequence of integers computed by GN-MIQP has been fixed in the MINLP (43), and the resulting NLP has been solved.

Regarding S-B-MIQP, the computation time is higher than GN-MIQP because of the sequential nature of the algorithm. Moreover, in this case, the termination of Algorithm 1 is related to the inability of the LB-MILP problem to compute a solution with the prescribed integrality gap within the given time limit. As shown in the top plot of Figure 10, the lower bound increases very slowly during the iterations. So, without the additional termination criterion, we would likely have had to wait much longer to obtain a tight lower bound for the incumbent solution. Despite the longer computation time, we could further improve the solution compared to the GN-MIQP approach. Note that only in nine iterations, i.e., eight MIQPs solved, the objective drops to 2375.84, and in sixteen iterations, we achieve the best incumbent solution with objective 2063.18, which is a 27% improvement with respect to objective obtained after only one iteration. In this case one iteration of Algorithm 1 corresponds to the GN-MIQP algorithm [14].

To conclude, with this example we aimed to show that Algorithm 1 can be applied out-of-the-box to a large and complex MINLP with satisfactory results. The computation time is prohibitive for real-time control applications, but Algorithm 1 can be used to compute a well performing solution offline. Also, the user can truncate the iterations early and expect to attain a better solution compared to a single iteration.

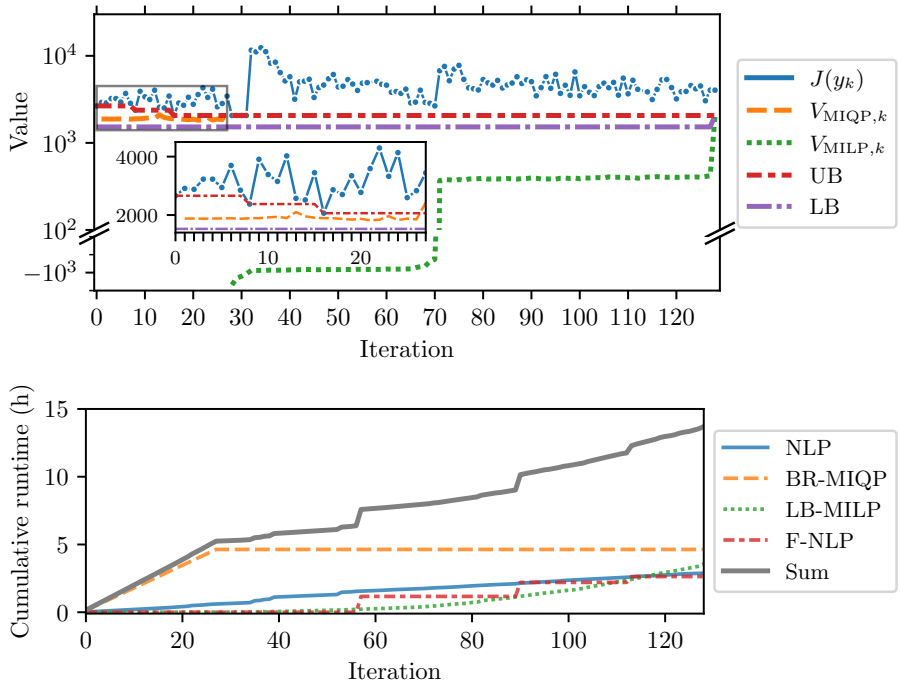

Fig. 10: Top: Objective value of the problems (43) solved during the iterations of Algorithm 1 and upper/lower bounds. The inset shows the results for the first 27 iterations. Bottom: Cumulative wall time for the problems and in gray the sum of all components.

Algorithm	Objective	Runtime (hh:mm:ss)
Relaxed NLP	1526.01	00:00:09
Bonmin	N/A	N/A
CIA [13]	5875.35	00:01:43
GN-MIQP [14]	2825.11	01:06:14
S-B-MIQP	2063.18	13:42:34

Table 3: Comparison of different algorithms for the solution of (43).

5 Conclusion and outlook

We presented a novel algorithm for solving mixed-integer nonlinear programming problems. We showed that the algorithm combines cutting planes based on generalized Benders' decomposition and outer approximation in an efficient way and converges to the global optimum or with a certificate of infeasibility for convex MINLPs. We proposed an extension for treating nonconvex MINLPs employing a heuristic to modify the generated cutting planes. The extension does not alter the results for convex MINLPs while it makes the algorithm directly applicable to nonconvex problems. The algorithm was compared to SHOT and Bonmin for a large subset of generic MINLPs from the MINLPLib, resulting in lower

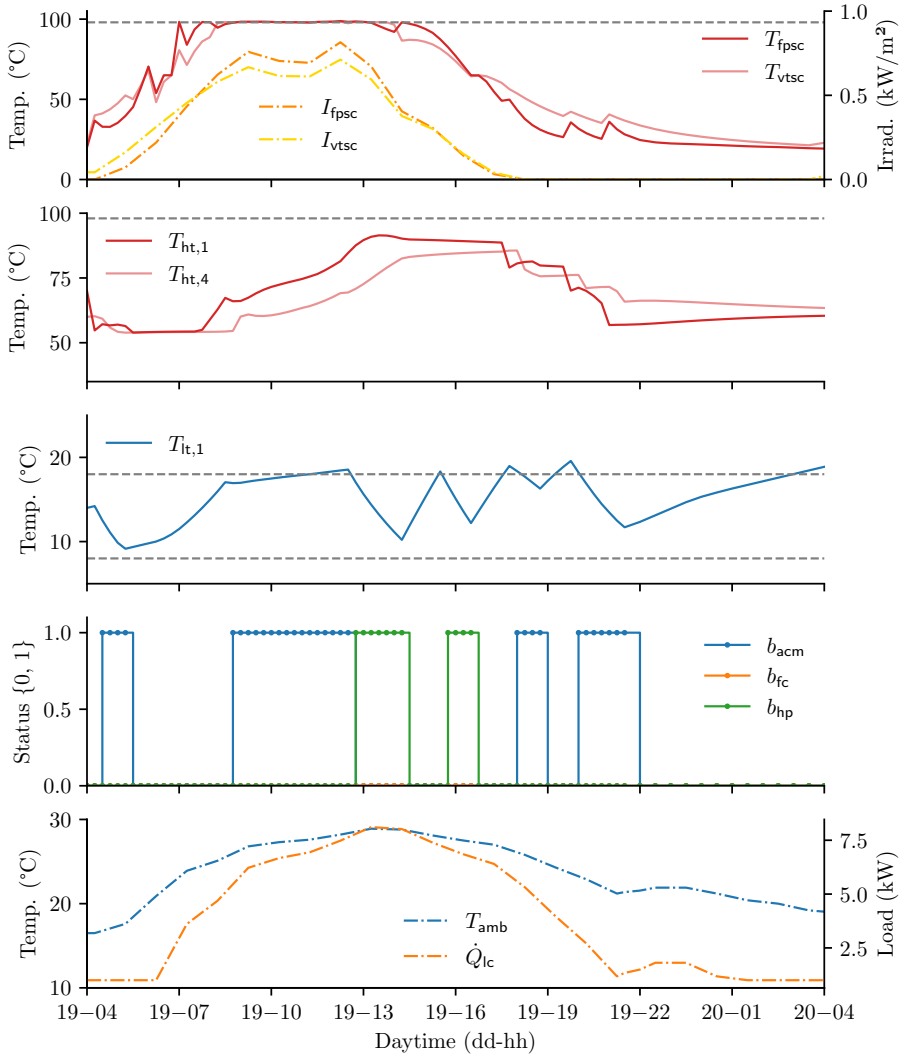

Fig. 11: State and binary control trajectory obtained by solving (43) via Algorithm 1. The plotted solution is computed at iteration 16 and corresponds to the one with lowest objective. The bounds on state and control are the dashed gray lines. The ambient conditions are represented by dashed-dotted lines, we have solar irradiance on the flat plate and vacuum tube solar collectors, $I_{\text{fpsc}}, I_{\text{vtsc}}$, respectively, the ambient temperature, T_{amb}, and the cooling load profile, $\dot{Q}_{\text{lc}}$.

objective values for many instances. Finally, we presented the results obtained with the proposed algorithm in two cases of optimal control for switched systems. Unfortunately, for these problems we could not compare the results against SHOT due to the lack of a working interface between CasADi and SHOT.

The software package, developed to implement the proposed algorithm, is coded in Python and relies on CasADi for modelling the optimization problems and interfacing with required solvers. Additionally, it includes implementations of various methods found in the literature. This comprehensive inclusion enables users to evaluate and compare the performance of different methods for their specific MINLP. On the implementation level, a welcome addition would be an interface between SHOT and CasADi, possibly using AMPL-file generation from CasADi. In general, having an efficient generation of AMPL-file for optimal control problems (OCPs) treated via direct approaches within CasADi, it would make possible to have a prompt interface to every AMPL-compatible solver. An effort in this direction has been carried out in TACO [33], to make MUSCOD-II [19], a multiple shooting code for optimal control, compatible with AMPL syntax.

Overall, the proposed algorithm shows promising solution quality and computation time results. This performance might be further improved by heuristics and other cutting strategies. One way could be by adding second-order approximation cutting planes. Another direction to improve the heuristic for nonconvex problems is by including a branch-and-bound strategy when nonconvexities are detected. Another strategy to improve the computational efficiency of the proposed algorithm is employing a branch-and-check strategy [52] where the solution from the MIQP-step can be reused between the different iterations. Future work also includes the extension of the proposed algorithm to Model Predictive Control.

Acknowledgements

A preliminary version of the algorithm proposed in this work has been presented in the Oberwolfach meeting “Mixed-integer Nonlinear Optimization: A Hatchery for Modern Mathematics” in August 2023 [20]. The preliminary algorithm only addressed *convex* MINLP without proving any of the theoretical findings.

We thank Adrian Bürger, Clemens Zeile, Léo Simpson, and Ramin Abbasi-Esfeden for inspiring discussions. Andrea Ghezzi and Moritz Diehl acknowledge funding from the European Union via ELO-X 953348, the German Research Foundation (DFG) via Research Unit FOR 2401, project 424107692, 525018088, and the German Federal Ministry for Economics and Climate Action (BMWK) via 03EI4057A and 03EN3054B. Wim Van Roy is supported by the Baekeland scholarship (Grant 182076), given by Atlas Copco Airpower NV, Wilrijk, Belgium, and by the Institute for the Promotion of Innovation through Science and Technology in Flanders (VLAIO), Belgium. Sebastian Sager acknowledges funding from the DFG via RTG 2297 and SPP 2331.

References

1. Abbasi-Esfeden R, Van Roy W, Swevers J (2023) Iterative switching time optimization for mixed-integer optimal control problems. In: Proceedings of the European Control Conference (ECC), IEEE, pp 1–6
2. Achterberg T (2009) SCIP: solving constraint integer programs. Mathematical Programming Computation 1:1–41

3. Andersson JAE, Gillis J, Horn G, Rawlings JB, Diehl M (2019) CasADi – a software framework for nonlinear optimization and optimal control. Mathematical Programming Computation 11(1):1–36
4. Axelsson H, Wardi Y, Egerstedt M, Verriest E (2008) Gradient descent approach to optimal mode scheduling in hybrid dynamical systems. Journal of Optimization Theory and Applications 136(2):167–186
5. Belotti P, Lee J, Liberti L, Margot F, Wächter A (2009) Branching and bounds tightening techniques for non-convex MINLP. Optimization Methods & Software 24(4-5):597–634
6. Belotti P, Kirches C, Leyffer S, Linderoth J, Luedtke J, Mahajan A (2013) Mixed-integer nonlinear optimization. Acta Numerica 22:1–131
7. Bock HG, Plitt KJ (1984) A multiple shooting algorithm for direct solution of optimal control problems. In: Proceedings of the IFAC World Congress, Pergamon Press, pp 242–247
8. Bonami P, Biegler L, Conn A, Cornuéjols G, Grossmann I, Laird C, Lee J, Lodi A, Margot F, Sawaya N, Wächter A (2005) An Algorithmic Framework For Convex Mixed Integer Nonlinear Programs. Tech. rep., IBM T. J. Watson Research Center
9. Boyd S, Vandenberghe L (2004) Convex Optimization. University Press, Cambridge
10. Bürger A, Zeile C, Altmann-Dieses A, Sager S, Diehl M (2019) Design, implementation and simulation of an MPC algorithm for switched nonlinear systems under combinatorial constraints. Journal of Process Control 81:15 – 30
11. Bürger A, Bohlayer M, Hoffmann S, Altmann-Dieses A, Braun M, Diehl M (2020) A whole-year simulation study on nonlinear mixed-integer model predictive control for a thermal energy supply system with multi-use components. Applied Energy 258:114064
12. Bürger A, Zeile C, Hahn M, Altmann-Dieses A, Sager S, Diehl M (2020) pycombina: An open-source tool for solving combinatorial approximation problems arising in mixed-integer optimal control. In: Proceedings of the IFAC World Congress, vol 53, pp 6502–6508
13. Bürger A, Bull D, Sawant P, Bohlayer M, Klotz A, Beschütz D, Altmann-Dieses A, Braun M, Diehl M (2021) Experimental operation of a solar-driven climate system with thermal energy storages using mixed-integer nonlinear model predictive control. Optimal Control Applications and Methods pp 1–27
14. Bürger A, Zeile C, Altmann-Dieses A, Sager S, Diehl M (2023) A Gauss–Newton-based decomposition algorithm for nonlinear mixed-integer optimal control problems. Automatica 152:110967
15. Büskens C, Wassel D (2013) The ESA NLP solver WORHP. Modeling and optimization in space engineering pp 85–110
16. Bussieck MR, Drud AS, Meeraus A (2003) Minlplib – a collection of test models for mixed-integer nonlinear programming. INFORMS Journal on Computing 15(1):114–119
17. Byrd RH, Nocedal J, Waltz RA (2006) KNITRO: An integrated package for nonlinear optimization. In: Pillo G, Roma M (eds) Large Scale Nonlinear Optimization, Springer Verlag, pp 35–59
18. Dakin R (1965) A tree-search algorithm for mixed integer programming problems. The Computer Journal 8:250–255
19. Diehl M, Leineweber D, Schäfer A (2001) MUSCOD-II Users' Manual. IWR-Preprint 2001-25, University of Heidelberg
20. Diehl M, Ghezzi A, Van Roy W, Sager S (2023) A sequential mixed-integer quadratic programming algorithm for solving MINLP arising in optimal control. Mixed-integer

Nonlinear Optimization: A Hatchery for Modern Mathematics (35), DOI 10.4171/OWR/2023/35
21. Duran M, Grossmann I (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical Programming 36(3):307–339
22. Exler O, Schittkowski K (2007) A trust region sqp algorithm for mixed-integer nonlinear programming. Optimization Letters 1:269–280
23. Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Mathematical Programming 66:327–349
24. Forrest J, Lougee-Heimer R (2005) Cbc user guide. In: Emerging theory, methods, and applications, INFORMS, pp 257–277
25. Garey M, Johnson D (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York
26. Geoffrion A (1972) Generalized Benders Decomposition. Journal of Optimization Theory and Applications 10:237–260
27. Ghezzi A, Simpson L, Buerger A, Zeile C, Sager S, Diehl M (2023) A Voronoi-based mixed-integer Gauss-Newton algorithm for MINLP arising in optimal control. Proceedings of the European Control Conference (ECC)
28. Gupta O, Ravindran A (1985) Branch and Bound experiments in convex nonlinear integer programming. Management Science 31:1533–1546
29. Gurobi Optimization, LLC (2024) Gurobi Optimizer Reference Manual. URL `https://www.gurobi.com`, Last accessed: 2024-04-01
30. HSL (2024) A collection of Fortran codes for large scale scientific computation. URL `http://www.hsl.rl.ac.uk`, Last accessed: 2024-04-01
31. Huangfu Q, Hall JJ (2018) Parallelizing the dual revised simplex method. Mathematical Programming Computation 10(1):119–142
32. IBM Corp (2022) IBM ILOG CPLEX V22.1, User's Manual for CPLEX. URL `https://www.ibm.com/products/ilog-cplex-optimization-studio`, Last accessed: 2024-04-01
33. Kirches C, Leyffer S (2013) Taco: a toolkit for ampl control optimization. Mathematical Programming Computation 5:227–265
34. Köppe M (2011) On the complexity of nonlinear mixed-integer optimization. In: Mixed Integer Nonlinear Programming, Springer, pp 533–557
35. Kronqvist J, Lundell A, Westerlund T (2016) The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. Journal of Global Optimization 64:249–272
36. Kronqvist J, Bernal DE, Grossmann IE (2020) Using regularization and second order information in outer approximation for convex MINLP. Mathematical Programming 180(1):285–310
37. Lee HJ, Teo KL, Rehbock V, Jennings LS (1999) Control parametrization enhancing technique for optimal discrete-valued control problems. Automatica 35(8):1401–1407
38. Li D, Sun X (2006) Nonlinear integer programming, vol 84. Springer
39. Lundell A, Kronqvist J (2022) Polyhedral approximation strategies for nonconvex mixed-integer nonlinear programming in SHOT. Journal of Global Optimization 82(4):863–896
40. Lundell A, Kronqvist J, Westerlund T (2022) The supporting hyperplane optimization toolkit for convex MINLP. Journal of Global Optimization 84(1):1–41
41. Matthias Miltenberger (2024) Visualizations of Mittelmann benchmarks. URL `https://mattmilten.github.io/mittelmann-plots/`, Last accessed: 2024-04-01

42. Messerer F, Baumgärtner K, Diehl M (2021) Survey of sequential convex programming and generalized Gauss-Newton methods. ESAIM: Proceedings and Surveys 71:64–88
43. Misener R, Floudas CA (2014) Antigone: algorithms for continuous/integer global optimization of nonlinear equations. Journal of Global Optimization 59(2-3):503–526
44. MOSEK ApS (2024) `https://www.mosek.com`, Last accessed: 2024-04-01
45. Quesada I, Grossmann I (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Computers and Chemical Engineering 16:937–947
46. Rawlings JB, Mayne DQ, Diehl MM (2017) Model Predictive Control: Theory, Computation, and Design, 2nd edn. Nob Hill
47. Robuschi N, Zeile C, Sager S, Braghin F (2021) Multiphase mixed-integer nonlinear optimal control of hybrid electric vehicles. Automatica 123:109325
48. Sager S, Jung M, Kirches C (2011) Combinatorial integral approximation. Mathematical Methods of Operations Research 73(3):363–380
49. Sahinidis NV (1996) Baron: A general purpose global optimization software package. Journal of global optimization 8:201–205
50. Sahinidis NV (2019) Mixed-integer nonlinear programming 2018. Optimization and Engineering 20:301–306
51. Smith EM, Pantelides CC (1999) A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex minlps. Computers & Chemical Engineering 23(4-5):457–478
52. Thorsteinsson ES (2001) Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In: Principles and Practice of Constraint Programming—CP 2001: 7th International Conference, Springer, pp 16–30
53. Tsang T, Himmelblau D, Edgar T (1975) Optimal control via collocation and non-linear programming. International Journal on Control 21:763–768
54. Wächter A, Biegler LT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106(1):25–57
55. Westerlund T, Pettersson F (1995) A cutting plane method for solving convex MINLP problems. Computers and Chemical Engineering 19:S131–S136
56. Zeile C, Robuschi N, Sager S (2021) Mixed-integer optimal control under minimum dwell time constraints. Mathematical Programming 188(2):653–694