Abstract This paper presents a novel partially distributed outer approximation algorithm, named PaDOA, for solving a class of structured mixed integer convex programming (MICP) problems to global optimality. The proposed scheme uses an iterative outer approximation method for coupled mixed integer optimization problems with separable convex objective functions, affine coupling constraints, and compact domain. PaDOA proceeds by alternating between solving large-scale structured mixed-integer linear programming problems and partially decoupled mixed-integer nonlinear programming subproblems that comprise much fewer integer variables. We establish conditions under which PaDOA converges to global minimizers after a finite number of iterations and verify these properties with an application to thermostatically controlled loads.

Keywords Mixed Integer Programming · Distributed Optimization · Outer Approximation · Global Optimization

1 Introduction

A mixed integer convex program (MICP) is an optimization problem with convex objective and constraint functions, where the only non-convex constraint is that a subset of the optimization variables need to be integer-valued [7,33]. MICPs arise in a plethora of application areas ranging from AC transmission expansion planning and robust power flow problems [1,29], via thermal unit design and control [10], a variety of scheduling and layout design problems [44].
design of multi-product batch plants [41], to obstacle avoidance and robotic motion planning problems [31].

Although MICPs are NP-hard in general, there exist a variety of algorithms for solving MICPs to global optimality [7]. State-of-the-art MICP solvers are based on tailored methods that exploit the fact that the integrality constraints are discrete while all other constraints are convex. Early attempts to develop tailored branch & bound methods for MICP have been proposed in [22], mostly focusing on computational experiments and heuristics for selecting the branching variables and nodes. Improved versions of these early branch & bound methods for MICP can be found in [8,32]. Other early methods for solving MICP include generalized Benders decomposition methods [4,20], which are, however, less frequently used in state-of-the-art MICP solvers.

Modern MICP implementations are often, in one or the other way, based on or related to outer approximation (OA), which goes back to Duran and Grossmann [14]. In contrast to branch & bound, OA alternates between solving nonlinear programs (NLP) with fixed integer values as well as mixed integer linear programs (MILP), which are constructed by linearizing the objective and constraint functions at the solutions of the NLP and which are used to update the integer variables. A notable extension of OA has been developed by Fletcher and Leyffer [17], who suggest to include curvature information in the relaxed integer program leading to a quadratic outer approximation method. Moreover, Kesavan and co-workers [26] have studied variants of OA for solving non-convex mixed integer problems. Another class of MICP methods is based on (extended) cutting plane methods [50] or combination of OA and branch-and-cut [40]; see also [17] for a general overview of polyhedral branch-and-cut methods. In recent years, there has been considerable progress in lift-and-project methods for MICP. An excellent overview and discussion of the state-of-the-art of such lift-and-project methods can be found in a recent article by Kilinç, Linderoth, and Luedtke [27].

Another recent trend in MICP solver development is the exploitation of separable structures by so-called extended formulations [23]. Here, the main idea is to introduce auxiliary variables in order to bound decoupled summands in additive expressions separately [49], which can lead to tighter polyhedral outer approximations. Such extended formulations have not only found their way into OA methods, as discussed in [23], but they can also be used to increase the performance of lift-and-project methods for MICP [27]. However, extended formulations exploit the separability for the construction of tighter outer approximations only, but neither existing OA methods nor state-of-the-art lift-and-project methods ever attempt to break a large-scale MICP into decoupled MICPs with fewer integer variables. This is in contrast to distributed continuous convex optimization methods, such as dual decomposition [16,36], alternating direction method of multipliers (ADMM) [9,15,18], or augmented lagrangian based alternating direction inexact newton (ALADIN) methods [24], which can all be used to solve large-scale convex optimization problems to

---

1 For more details see [7].
global optimality by alternating between solving small-scale convex optimization problems and sparse linear algebra operations. These methods typically require communication of the solutions of the decoupled problems between neighbors or to a central coordinator \([9]\). Although some researchers, \([46,34]\), have attempted to apply these distributed local optimization methods in a heuristic manner, these methods cannot find global minimizers of non-convex problems reliably. This is due to the fact that ADMM, ALADIN, or similar distributed convex optimization method typically rely on strong duality results for augmented Lagrangians \([45,43]\), which fail to hold in the presence of integrality constraints.

After reviewing Hijazi’s extended formulations and related existing outer approximation methods in Section 2, the main contribution of this paper is presented in Section 3 which introduces a partially distributed outer approximation (PaDOA) method for MICPs with separable objective functions. In contrast to existing algorithm for structured MICP, PaDOA alternates between solving MICPs with fewer integer variables and large-scale MILPs for which efficient algorithms exist. Section 3.5 discusses the global convergence properties of PaDOA, as summarized in Theorem 3. In this context, we additionally establish the fact that global optimality of a given feasible point of an MICP with \(N\) separable objectives and \(Nn\) optimization variables can be computationally verified by solving \(N\) partially-decoupled MICPs, each comprising at most \(n\) local integer variables, and one MILP with \(Nn\) integer variables. This result is summarized in Theorem 2, which analyzes one-step convergence conditions for PaDOA. In the sense that both MICPs as well as MILPs are NP hard in general \([19,35]\), this result is not in conflict with existing complexity results for mixed integer optimization problems. However, there are solvers such as CPLEX \([25]\), Gurobi \([38]\), and many others \([11]\), which can solve practical MILPs within reasonable computational run-times. Thus, the fact that one can reduce the task of verifying global optimality of a feasible point of a separable MICP with coupled affine constraints to the task of solving one MILP of a comparable size and several smaller subproblems, is—at least from a computational perspective—an important contribution. Last but not least, Section 4 illustrates the practical performance of PaDOA by applying the algorithm to MICP benchmark case studies. Section 5 concludes the paper.

1.1 Problem formulation

The present paper is concerned with mixed integer optimization problems of the form

\[
V^* = \min_{x \in X, z \in Z} f(x, z) \quad \text{s.t.} \quad Ax = b
\]  

(1)
with separable objective function \( f(x, z) = \sum_{i=1}^{N} f_i(x_i, z_i) \) and separable constraint sets

\[
X = X_1 \times \ldots \times X_N \quad \text{with} \quad X_1, X_2, \ldots, X_N \subseteq \mathbb{R}^n
\]
and
\[
Z = Z_1 \times \ldots \times Z_N \quad \text{with} \quad Z_1, Z_2, \ldots, Z_N \subseteq \mathbb{Z}^m.
\]

(2)

The coupling matrix \( A \) and the vector \( b \) are assumed to be given. In this context, the following blanket assumption is used.

**Assumption 1** The sets \( X_1, X_2, \ldots, X_N \subseteq \mathbb{R}^n \) are non-empty convex polytopes, the sets \( Z_1, Z_2, \ldots, Z_N \subseteq \mathbb{Z}^m \) are non-empty and compact, and the functions \( f_i \) are convex on the convex hull of \( X_i \times Z_i \).

The goal of this paper is to develop an efficient algorithm that finds \( \varepsilon \)-suboptimal points of (1), which are defined as follows:

**Definition 1** A feasible point \((x^\star, z^\star) \in X \times Z\) with \( Ax^\star = b \) is said to be an \( \varepsilon \)-suboptimal point of (1), with \( \varepsilon > 0 \), if

\[
f(x^\star, z^\star) \leq f(x, z) + \varepsilon.
\]

for all \((x, z) \in X \times Z\) with \( Ax = b \).

**Remark 1** Instead of (1), one could also consider more general optimization problems of the form

\[
\min_{x \in X, z \in Z} f(x, z)
\]

s.t. \( Ax + Bz = b \).

(3)

However, under mild regularity assumptions [37], this problem is equivalent to

\[
\min_{x \in X, y \in \text{conv}(Z), z \in Z} \sum_{i=1}^{N} \left\{ f_i(x_i, z_i) + \bar{\lambda}_i \|y_i - z_i\|_1 \right\}
\]

s.t. \( Ax + By = b \).

(4)

with real-valued auxiliary variables \( y \) and \( L_1 \)-penalty parameters \( \bar{\lambda}_i \gg 0 \). Here, \( \text{conv}(Z) \) denotes the convex hull of \( Z \). Thus, for all theoretical purposes, it is sufficient to analyze problems of the form (1), where only the real-valued variables are coupled.

**Remark 2** Modern MICP formulations, algorithms, and software can deal with rather general convex conic constraints [33]. Such constraints are left out for simplicity of presentation. Nevertheless all results in this paper can be easily extended for general conic constraints, as long as they are separable. From a purely theoretical perspective, one might argue that this can always be achieved by adding suitable convex penalty functions to the functions \( f_i \), because this paper makes no assumptions on the differentiability properties of \( f \).

However, more tailored, practical algorithms that could exploit the structures of particular conic constraints are beyond the scope of this paper.
1.2 Notation

We use the notation

\[ \partial_x g(x) = \{ a \in \mathbb{R}^n \mid \forall y \in \mathbb{R}^n, \ g(y) \geq g(x) + a^T(y - x) \} \]

to denote the set of subgradients of a convex function \( g : \mathbb{R}^n \to \mathbb{R} \) with respect to the variable \( x \).

2 Outer approximation

2.1 Polyhedral relaxations

In order to construct polyhedral outer approximations of the epigraph of the objective function \( f \) of (1), we consider the auxiliary optimization problem

\[
\begin{align*}
 f^*(z) &= \min_{x,y} f(x,z) \quad \text{s.t.} \quad \begin{cases} 
 x = y & | \lambda \\
 Ay = b \\
 y \in X.
\end{cases} \\
&= \max_{\lambda} \min_{x,y} f(x,z) + \lambda^T(y - x) \quad \text{s.t.} \quad \begin{cases} 
 Ay = b \\
 y \in X.
\end{cases}
\end{align*}
\]

(6)

for a fixed integer parameter \( z \in Z \). Here, \( x \) and \( y \) are real valued primal optimization variables and the notation \( "x = y \mid \lambda" \) is used to say that \( \lambda \) denotes the dual variable that is associated with the constraint \( x = y \).

**Proposition 1** If Assumption I is satisfied, strong duality holds for (6), i.e., we have

\[
 f^*(z) = \max_{\lambda} \min_{x,y} f(x,z) + \lambda^T(y - x) \quad \text{s.t.} \quad \begin{cases} 
 Ay = b \\
 y \in X.
\end{cases}
\]

for all \( z \in Z \).

**Proof** See Appendix A.1.

Let \( x^*(z), y^*(z), \lambda^*(z) \) denote any primal-dual solution of (6) in dependence on \( z \). By writing out the stationarity condition of (6) with respect to \( x \), we find that

\[
 \lambda^*(z) \in \partial_x f(x^*(z), z),
\]

i.e., \( \lambda^*(z) \) must be a subgradient of \( f \) at the optimal solution of (6). In order to understand the developments given below it is helpful to keep in mind that the reverse statement is not correct, i.e., a subgradient of \( f \) at \( (x^*(z), z) \) is not necessarily a dual solution of (5).

In contrast to the particular choice of the subgradient \( \lambda^* \) of \( f \) with respect to \( x \), the construction of a subgradient of \( f \) with respect to \( z \) is less critical for the construction of outer approximation methods. In the following, we assume that a function

\[
 \mu^*(z) \in \partial_z f(x^*(z), z),
\]
is given, which returns a subgradient of $f$ with respect to $z$ at the optimal solution of (5). Because $f(x, z) = \sum_{i=1}^{N} f_i(x_i, z_i)$ is separable, the $i$-th block components, $\lambda^*_i(z)$ and $\mu^*_i(z)$, of the subgradients of $f$ are subgradients of $f_i$. Thus, the inequality

$$
f_i(x_i, z_i) \geq f^*_i(\hat{z}) + [\lambda^*_i(\hat{z})]^T (x_i - x^*_i(\hat{z})) + [\mu^*_i(\hat{z})]^T (z_i - \hat{z}_i)
$$

holds for all $x_i \in X_i$, and $z_i \in Z_i$ and all $\hat{z} \in Z$. Here, the shorthand

$$
f^*_i(\hat{z}) = f_i(x^*_i(\hat{z}), \hat{z}_i)
$$

is used. In this context, it is important to notice that the function $f^*_i(\hat{z})$ depends on the whole vector $\hat{z}$, not only on its $i$-th component, $\hat{z}_i$, because the equality constraints in (6) introduce a non-trivial coupling. More generally, if $\Xi \subseteq Z_i$ denotes finite set of points in $Z$, we associate with $\Xi$ a set of hyperplane coefficients

$$
\mathcal{H}_i(\Xi) = \left\{ (\alpha, \beta, \gamma) \mid \begin{array}{l}
z \in \Xi \\
\alpha = \lambda^*_i(z) \\
\beta = \mu^*_i(z) \\
\gamma = f^*_i(z) - \alpha^T x^*_i(z) - \beta^T z_i
\end{array} \right\}.
$$

(7)

Notice that this set of hyperplane coefficients defines a polyhedral outer approximation of the epigraph of $f_i$. Thus, these coefficients can be used to construct a piecewise affine lower bound on $f_i$, which is for all $(x_i, z_i) \in X_i \times Z_i$ given by

$$
\Phi_i(x_i, z_i, \Xi) = \max_{(\alpha, \beta, \gamma) \in \mathcal{H}_i(\Xi)} \left\{ \alpha^T x_i + \beta^T z_i + \gamma \right\}.
$$

(8)

Finally, we can construct the function $\Phi(x, z, \Xi) = \sum_{i=1}^{N} \Phi_i(x_i, z_i, \Xi)$. This function is—by construction—a piecewise affine lower bound on $f$,

$$
\forall (x, z) \in X \times Z, \quad \Phi(x, z, \Xi) \leq f(x, z).
$$

(9)

Next, our particular choice of the subgradient $\lambda^*(z)$ of $f$ as the dual solution of (5) enables us to establish the following tightness property of the affine lower bound $\Phi$.

**Lemma 1** Let $\Xi \subseteq Z$ be any finite set of points. If Assumption [7] holds, then the equation

$$
f^*(z) = \min_{x \in X} \Phi(x, z, \Xi) \quad \text{s.t.} \quad Ax = b.
$$

holds for all $z \in \Xi$.

**Proof** See Appendix A.2.
Remark 3 If the set $\Xi$ consists of $m$ points, the computational cost for constructing the lower bound $\textbf{(9)}$ of $f$ has order $O(mN)$. Notice that if we would have ignored the separable structure of $f$, the computational cost of computing the same lower bounding function would have been of order $O(m^N)$. Thus, the construction of $\textbf{(9)}$ as a sum of the lower bounds of the separable objective function is much cheaper than a direct construction of lower bounds of $f$. This reduction in complexity has for the first time been observed and exploited by Hijazi and co-workers [23]. By now, the exploitation of separability via extended formulations can be considered as a standard that has been adopted in many modern MICP algorithms and software tools [27,33].

2.2 Outer approximation algorithm

Algorithm 1 outlines the main steps of the outer approximation algorithm. Notice that this algorithm basically coincides with the original outer approximation algorithm that has been proposed in [14]. The only notable differences of Algorithm 1 compared to traditional OA are that the MILP in Step 3 uses the extended formulation based outer approximation variant from [23]. Moreover, because we do not assume that $f$ is differentiable, we have to use the particular choice, $\lambda^*(z)$, of the subgradient, which is found as the dual solution $\textbf{2}$ of $\textbf{(5)}$. Notice that Step 1 of Algorithm 1 solves $\textbf{(1)}$ under the additional constraint that the integer $z$ is fixed. This implies that

$$f(x^*,z) \geq V^*$$

is an upper bound on the optimal objective value $V^*$ of $\textbf{(1)}$. Thus, the current upper bound $U$ can be updated in Step 2. Moreover, the MILP $\textbf{(21)}$ is (by construction) equivalent to solving the relaxed optimization problem $\textbf{3}$

$$\min_{x \in X, z \in Z} \Phi(x,z,\Pi) \quad \text{s.t.} \quad Ax = b,$$

which implies that

$$\sum_{i=1}^{N} y_i^+ \leq V^*$$

is a lower bound on $V^*$. Thus, the difference, $U - \sum_{i=1}^{N} y_i^+$, between the current upper and lower bounds can be used as a termination criterion, which is implemented in Step 3 of Algorithm 1. The following finite termination result for outer approximation is (at least in very similar versions) well-known in the literature [14,33].

---

2 The idea to use dual solutions as subgradients for the construction of polyhedral outer approximation is not new and can—in a very similar setting—be found in [33].

3 Since the inception of the idea of Gomory cuts in the 1960s [21], cutting plane methods for MILP have evolved significantly. Nowadays, there exist efficient algorithm and solvers for MILP and we refer to [11] for an overview.
Algorithm 1: Outer Approximation for MICP

Input: Initial guess \( z \in Z \) and a numerical tolerance \( \varepsilon > 0 \).

Initialization: Set \( \Pi = \emptyset \) and \( U = \infty \).

Repeat:

1. Solve the convex optimization problem

\[
\begin{align*}
\hat{f}(z) = \min_{x,y} & \quad f(x,z) \\
\text{s.t.} & \quad \begin{cases}
    x = y \\
    Ay = b \\
    y \in X.
\end{cases}
\end{align*}
\] (11)

2. If (11) has no feasible solution, return a certificate of infeasibility. Otherwise, update

\[
U \leftarrow \min\{U, \hat{f}(z)\} \quad \text{and} \quad \Pi \leftarrow \Pi \cup \{z\}.
\]

3. Solve the (extended) MILP

\[
(x^+, y^+, z^+) \in \arg\min_{x \in X, y \in Z} \sum_{i=1}^{N} y_i \\
\text{s.t.} \quad \begin{cases}
    \forall i \in \{1, \ldots, N\}, \\
    \forall (\alpha_i, \beta_i, \gamma_i) \in \mathcal{H}_i(\Pi), \\
    \alpha_i^T x_i + \beta_i^T z_i + \gamma_i \leq y_i \\
    Ax = b
\end{cases}
\] (12)

4. If \( U - \sum_{i=1}^{N} y_i^+ \leq \varepsilon \), terminate.

5. Update \( z \leftarrow z^+ \) and go to Step 1.

Theorem 1 If Assumption 2 is satisfied, then Algorithm 1 terminates after a finite number of iterations.

Proof See Appendix A.3.

Remark 4 Algorithm 1 uses Hijazi’s extended formulation [23] for constructing the MILPs (21), which arguably exploit separability of the objective function to some extent. However, Algorithm 1 is not a fully distributed algorithm. In fact, a major disadvantage of Algorithm 1 becomes apparent, if one considers the special case that the constraint \( Ax = b \) happens to be redundant. In this case, the optimal solution of (1) could have been found with much less effort by solving the separable MICPs,

\[
\min_{x_i \in X_i, z_i \in Z_i} f_i(x_i, z_i)
\]

which have much fewer integer variables. However, if Algorithm 1 is applied to such a problem with redundant equality constraint, this property is not detected and a large number of large-scale NLPs and large scale MILPs might have to be solved instead, until convergence is achieved. The goal of this paper is to mitigate this deficiency of Algorithm 1 by proposing a partially distributed...
outer approximation algorithm that exploits the structure of the separable objective in a better way.

3 Partially distributed outer approximation algorithm

This section introduces a partially distributed outer approximation optimization algorithm for finding $\varepsilon$-suboptimal solutions of (1).

3.1 Partially decoupled upper bounds

The main idea of many distributed convex and local optimization methods is to solve a set of smaller-scale decoupled optimization problems in place of a single large one [4,9,12]. Similarly, consider partially decoupled optimization problems of the form

$$
V_k(z) = \min_{x,y,\zeta_k} f_k(x_k, \zeta_k) + \Psi_k(x, z) \\
\text{s.t.} \quad \begin{cases}
x = y \\
Ay = b \\
y \in X \\
\zeta_k \in Z_k
\end{cases}
$$

for $k \in \{1, \ldots, N\}$. Here, the integer variable $z \in Z$ is regarded as a fixed parameter and only the much smaller dimensional integer vector $\zeta_k \in Z_k$ is optimized. However, concerning the real-valued variables, the whole vector $x \in X$ is kept as an optimization variable. In this context, the shorthands

$$
\forall (x, z) \in X \times Z, \quad \Psi_k(x, z) = \sum_{j \neq k} f_j(x_j, z_j)
$$

are introduced in order to keep the $x$-dependence of the remaining summands, i.e., all objective terms whose index is not equal to $k$. As in the previous section, $\lambda$ denotes the dual solution that is associated with the consensus constraint “$x = y$”. Because the constraint $\zeta_k \in Z_k$ enforces integrality, strong duality of (13) does not hold in general. However, if $\zeta_k^*(z)$ denotes an optimal solution of (13) for the integer variable and if Assumption [1] holds, we still have

$$
V_k(z) = \max_{\lambda} \min_{x,y} f_k(x_k, \zeta_k^*(z)) + \Psi_k(x, z) + \lambda^T(y - x) \\
\text{s.t.} \quad \begin{cases}
Ay = b \\
y \in X
\end{cases}
$$

The proof of this statement is completely analogous to Proposition [1], i.e., if the linear coupling constraint $Ax = b$ has a solution in $X$, a maximizer of (15)
exists and can be used to define a suitable subgradient. Also note that the functions \( V_k \) yield upper bounds on the objective value of (1),
\[
\forall z \in Z, \quad \min_k V_k(z) \geq V^* \tag{16}
\]
At this point, it should be mentioned that one basic assumption of the algorithmic developments in this paper is that the complexity of the mixed integer optimization problems of interest depends mostly on the number of integer variables. This is in contrast to the number of real-valued variables, which may be assumed to have a negligible influence on the overall complexity of the mixed-integer optimization problem. In other words, we assume that (13) is much easier to solve than (1) in the sense that it contains much fewer integer variables, although both problems have the same number of real-valued variables. Here, it is important to keep in mind that, although the algorithmic developments in this paper are inspired by the field of distributed optimization, the algorithm in this paper is (at least in the form in which we present and analyze it) not fully distributed. This is because solving (13) requires the evaluation of the function \( \Psi_k \), which, in turn, requires the evaluation of all functions \( f_j \) with \( j \neq k \).

### 3.2 Partially decoupled lower bounds

In this paper, we suggest to solve the decoupled MICPs (13) by lower level solvers that implement the traditional outer approximation algorithm that has been reviewed in Section 2.2. Notice that if Assumption 1 is satisfied, strong duality holds, i.e., these lower level solvers will return piecewise affine models
\[
\Theta_k^* : X \times Z_k \to \mathbb{R},
\]
which must satisfy the condition
\[
V_k(z) - \epsilon_L \leq \min_{x \in X, \zeta \in Z_k} \Theta_k^*(x, \zeta) \quad \text{s.t.} \quad Ax = b \tag{17}
\]
upon termination. Here, \( \epsilon_L \geq 0 \) denotes the numerical tolerance of the lower level OA solvers. Notice that the optimization problem on the right hand side of (17) corresponds to the last MILP relaxation that is solved by the lower level OA solver. In practice the function \( \Theta_k^* \) can be stored by maintaining a set of hyperplane coefficients as explained in detail in the previous section. Moreover, in order to avoid the accumulation of too many hyperplanes, one can discard all hyperplanes that are inactive at the optimal solution of the last MILP relaxation, because this operation does not affect the right hand expression of (17).

The main idea of partially distributed outer approximation is to communicate the piecewise lower bounding functions \( \Theta_k^* \) to a central coordinator, who constructs a piecewise affine lower bound on the function \( f \), solves a master
Partially Distributed Outer Approximation

MILP problem, and updates $z$. Here, one option is use the maximum over the function $\Theta_k^*$ in order to obtain the lower bound

$$\forall x \in X, \forall z \in Z, \max_k \Theta_k^*(x, z_k) \leq f(x, z). \quad (18)$$

However, in order to arrive at a practical implementation, it is recommendable to further refine this bound. This can be done by maintaining a collection of integers, $\Pi \subseteq Z$, such that the function

$$\Theta(x, z) = \max \left\{ \Phi(x, z, \Pi), \max_k \Theta_k^*(x, z_k) \right\}, \quad (19)$$

can be used as a piecewise affine lower bound on $f$. Recall that the function $\Phi$, which has been introduced in the previous section, exploits the separability properties of $f$. The integer collection $\Pi$ is then maintained by updating

$$\Pi \leftarrow \Pi \cup \{\zeta^*\},$$

where $\zeta^* = [\zeta^*_1, \zeta^*_2, \ldots, \zeta^*_N]$ is an integer vector, whose components are optimal solutions for the integer variables of the partially decoupled problems \[13\].

3.3 Partially distributed outer approximation (PaDOA)

Algorithm 2 outlines a partially distributed algorithms for solving (1). There are four main steps. In the first step, the partially decoupled MICPs of the form (13) are solved by using a traditional outer approximation method. Under the assumption that the original MICP (1) is feasible, the partially decoupled MICPs are feasible, too. Thus, the outer approximation solvers will return optimal integer solutions $\zeta_k^*$ and associated piecewise affine lower bounds $\Theta_k^*$ such that (17) is satisfied. The second step of Algorithm 2 updates the associated upper bound $U$ based on the inequality (16) as well as the piecewise affine lower bound. In practice, this step is implemented by storing the union of all supporting hyperplane coefficients that are needed to represent $\Theta$. Finally, the third step of Algorithm 2 solves a large scale MILP problem. This MILP is constructed in analogy to the corresponding step in the traditional outer approximation algorithm. It yields a lower bound,

$$\Theta(x^+, z^+) \leq V^*,$$

on the objective value $V^*$ of (1). Thus, the difference between the current upper and lower bounds,

$$U - \Theta(x^+, z^+),$$

can be used as a termination criterion, which is implemented in the fourth step of Algorithm 2. If the termination is not successful, the integer variables $z$ are updated, and the algorithm subsequently proceeds to the next iteration.

Notice that the main difference between Algorithms 1 and 2 is the introduction of partially decoupled MICP problems that can be solved separately.
Algorithm 2: Partially Distributed Outer Approximation (PaDOA)

**Input:** Initial guess $z \in Z$ and a numerical tolerance $\varepsilon > 0$.

**Initialization:** Set $\Pi = \emptyset$, $\Theta(\cdot, \cdot) = -\infty$, and $U = \infty$.

**Repeat:**
1. Solve for all $k \in \{1, \ldots, N\}$ the partially decoupled MICPs
   \[
   V_k(z) = \min_{x, y, \xi_k} f_k(x_k, \xi_k) + \Psi_k(x, z) \quad \text{with} \quad \Psi_k(x, z) = \sum_{j \neq k} f_j(x_j, z_j),
   \]
   s.t.\[
   \begin{align*}
   x &= y \\ Ay &= b \\
   y &\in X \\
   \xi_k &\in Z_k
   \end{align*}
   \] (20)
   If (20) is infeasible, terminate and return a certificate of infeasibility. Otherwise, update the set $\Pi \leftarrow \Pi \cup \{\xi_k\}$ and construct a piecewise affine model $\Theta_k^*$ such that condition (17) is satisfied.
2. Update the upper bound $U \leftarrow \min\{U, V_1(z), \ldots, V_N(z)\}$ and construct the piecewise lower bounding function $\Phi(x, z, \Pi)$ as in (8).
3. Update the lower bound
   \[
   \forall x \in X, \forall z \in Z, \quad \Theta(x, z) \leftarrow \max \left\{ \Theta(x, z), \Phi(x, z, \Pi), \max_k \Theta_k^*(x, z_k) \right\}
   \]
4. Solve the MILP problem
   \[
   (x^+, z^+) \in \arg\min_{x, z} \Theta(x, z) \quad \text{s.t.} \quad Ax = b
   \] (21)
5. If $U - \Theta(x^+, z^+) \leq \varepsilon$, terminate. Otherwise, update $z \leftarrow z^+$ and go to Step 1.

and which contain much fewer integer variables than the original MICP (1). The theoretical results in Section 3.5 will elaborate further on the benefits of this alternation strategy. Moreover, in Section 4 a numerical case study is examined, which illustrates the practical advantages of Algorithm 2.

3.4 Relation to distributed local optimization methods

The idea to “augment” the local objective functions $f_i$ with a suitable function $\Psi_i$ is frequently used in the context of distributed local optimization algorithms. For example, in the context of dual decomposition [16,36], one augments the separable functions $f_i$ with linear functions of the form [4]

$\Psi_i(x) = \sigma^T Ax$.

\[4\] In the context of convex optimization Problem (1) is considered without integer variables $z$—this is why the functions $\Psi_i$ depend in the convex case on $x$ only.
where $\sigma$ is the current dual iterate. Similarly, in the context of ADMM or ALADIN, one uses augmented Lagrangians [3,39], as in

$$\psi_i(x,y) = \sigma^T Ax + \frac{\rho}{2} ||x - y||^2,$$  \hspace{1cm} (22)

where $z$ and $\sigma$ are the current primal and dual iterates; see [9,15,24]. In fact, the construction of Algorithm 2 is inspired by the distributed local nonlinear programming method ALADIN. Here, we recall that ALADIN alternates between solving small-scale decoupled NLPs that are augmented by terms of the form (22) and large scale equality constrained quadratic programming problems that update $\sigma$ and $y$ [24]. This is in analogy to Algorithm 2, which alternates between solving decoupled MICPs (Step 1) and large-scale coupled MILPs (Step 3). However, unlike ALADIN, augmented Lagrangians are not used in Algorithm 2 as Lagrange multipliers in integer programming are not related to sensitivity and generally not applicable.

Also note that the construction of the functions $\Psi_i$ in Algorithm 2 also has similarities with Gauss-Seidel or more general block-coordinate descent methods [48,51] in the sense that a partial decoupling is obtained by fixing some of the integer variables while others are optimized. However, despite all these analogies and similarities of Algorithm 2 with methods from the field of local and convex optimization, we would like to highlight that all these existing distributed optimization methods are not reliably applicable [46].

3.5 Convergence Analysis

In this section we provide a concise overview of the convergence properties of Algorithm 2. The following theorem establishes one of the main results of this paper, namely, that Algorithm 2 converges after one iteration if the integer iterate, $z$, is initialized with an optimal solution of (1). This is contrast to Algorithm 1, which does not necessarily terminate after a small number of steps—not even if it is initialized at an optimal solution.

**Theorem 2** Let Assumption 1 be satisfied and let $(x^*, z^*)$ be a minimizer of (1). If Algorithm 2 is initialized with $z = z^*$ and if the termination tolerances of the lower level solvers satisfy $\epsilon_L \leq \epsilon$, then the termination criterion in Step 4 is satisfied. In other words, the algorithm terminates after one step.

**Proof** See Appendix A.4.

Notice that the statement of the above theorem is of fundamental relevance and a very favorable property of PaDOA. If we work with other global optimization methods, say branch-and-bound, an empirical observation is that such existing global optimization algorithm often find a global solution early on but then keep on iterating until the lower bound is accurate enough to prove global optimality. In contrast to this, PaDOA terminates as soon as a global minimizer is added to the collection $\Pi$. In fact, Theorem 2 implies that
global optimality of a point \( z^* \in Z \) can be verified by solving the \( N \) instances of the partially decoupled MICPs and the master MILP \((21)\). Notice that this result is not in conflict with existing results from the field of complexity theory, because the master MILP \((21)\) remains NP-hard \([19,35]\).

Remark 5 The result of Theorem 2 relies heavily on the convexity of the functions \( f_i \) on the convex hull of \( X_i \times Z_i \), although this fact is not highlighted explicitly in the proof. This convexity assumption is first of all required implicitly by our assumption that the lower level solvers return piecewise level models \( \Theta_k \), which satisfy the termination condition \((17)\) (this assumption is only reasonable if strong duality holds) and which need to be global lower bounds on \( f \). These properties are in general all not satisfied if one considers more general non-convex MINLPs.

The following theorem establishes the fact that Algorithm 2 converges after a finite number of iterations under exactly the same conditions under which convergence of Algorithm 1 can be established.

**Theorem 3** Let Assumption 1 be satisfied. If the termination tolerances of the lower level solvers satisfy \( \epsilon_L \leq \epsilon \), then Algorithm 2 terminates after a finite number of steps (independently of the initialization).

**Proof** See Appendix A.5.

### 4 Implementation and Case Study

The Partially Distributed Outer Approximation method is implemented in MATLAB R2017b. The optimization subproblems are solved using Gurobi \([38]\) implemented via CasADi v1.9.0 \([2]\). All numerical experiments were run on a 2.9GHz Intel Core i5-4460S CPU with 8GB of RAM.

#### 4.1 Problem Description

An important problem in the planning and operation of a heating and/or cooling system is the scheduling of so-called Thermostatically Controlled Loads (TCLs) \([30]\). These are devices that are used to regulate the temperature of a room/building within a certain user-defined interval known as a “deadband”. The optimal operation strategy is especially difficult to determine when a non-constant cost function is introduced for a population of heterogeneous TCLs \([52]\). The cost function may represent the cost of electricity or user-discomfort from noise generation. Regardless, such devices typically only have an “on” and an “off” setting and thus the resulting scheduling problem can be formulated as a binary MIP as seen in \((23)\) for \( R \) regions with a finite time horizon \( H \). The equations in \((23)\) are based on the formulation given in \([28]\), but with dynamics modeling the interaction between each region and a linear cost function instead of a quadratic.
The objective is to minimize the sum of the device costs and the comfort cost over time:

$$\min_{T_i(t), u(t)} \sum_{t=0}^{H-1} c(t)u(t) + \gamma(T_i(t) - T_{ref}(t))^2,$$

subject to:

$$\forall i \in \{1, \ldots, R\},$$

$$T_i(t) \leq T(t) \leq T_i,$$ $\forall t \in \{0, \ldots, H\}$

$$u_i(t) \in \{0, 1\}, \quad \forall t \in \{0, \ldots, H - 1\}$$

$$\forall t \in \{0, \ldots, H\},$$

$$T_i(t + 1) = T_i(t) + b_i u_i(t) + a_i \left( \frac{T_i(t) + T_{amb}(t) + \sum_{j \in N(i)} T_j(t)}{|N(i)| + 2} - T_i(t) \right),$$

where $c(t)$ is the vector of device costs at time $t$, $\gamma$ is a comfort parameter, $T_i$ and $T_{i}$ are the deadband temperature limits of device $i$, $a_i$ and $b_i$ are heat transfer parameters, $T_{amb}(t)$ is the ambient temperature at time $t$ and $N(i)$ are the number of regions neighbouring $i$. Equation (23d) models the thermodynamics of each room in a simplified manner, i.e., it takes an average of the current and surrounding temperatures to update the temperature of the next time step. This formulation results in $H + 1$ real-valued and $H$ binary variables per region. Figure 4.1 shows two possible initial configurations of (23). The ambient temperature is taken from [13] for two days in June 2017 in the Karlsruhe (Germany) area. High prices of $25.67$/kW are set from 2pm to 8pm (time steps 6 to 12 and 29 to 35) with low and medium prices of $2.46$/kW and $4.62$/kW in all other time steps. Each region is initialized at 20 degrees with $a_i = 0.2$ and $b_i = -2$.

![Fig. 1](image-url) Two room configurations with controlled cooling elements $u_i$, ambient temperature $T_{amb}$ and initial temperatures $T_i(0)$.

### 4.2 Results for MILP

If the comfort parameter $\gamma$ is taken to be zero then Problem (23) is linear and separable but coupled in both its discrete and real-valued variables. Shown in
Tables 1 and 2 are the simulation results for each configuration, respectively. The results of Algorithm 2 are compared with results obtained from a Branch and Bound approach as implemented in Bonmin with default settings [6] as well as the commercial MIQP solvers Gurobi and CPLEX [25]. An example solution for the 3 room case is depicted in Figure 4.2.

![Figure 2](image-url)

**Fig. 2** The red dotted line is the ambient temperature, the blue dotted lines are the limits of the temperature deadzone and the solid lines are the temperature trajectories of each region in the three room scenario. Highlighted on the trajectories are points where the coolers are activated.

### Table 1 Results obtained for the TCL problem with a 3-room configuration.

|       | Time steps: | 8   | 24  | 48  | 62  |
|-------|-------------|-----|-----|-----|-----|
| Alg. 2 | obj.        | 0   | 13.86 | 16.32 | 21.23 |
|       | time(s)     | 0.15 | 0.20 | 0.89 | 5.23 |
|       | iter.       | 2   | 1   | 2   | 2   |
| B-OA  | obj.        | 0   | 13.86 | 16.32 | 21.23 |
|       | time(s)     | 0.15 | 31.06 | 1,345 | 52,395 |
| B-BB  | obj.        | 0   | 13.86 | 16.32 | 21.23 |
|       | time(s)     | 0.15 | 9.13 | 480.27 | 828.08 |
| Gurobi | obj.        | 0   | 13.86 | 16.32 | 21.23 |
|       | time(s)     | 0.22 | 0.56 | 0.85 | 4.39 |
| CPLEX | obj.        | 0   | 13.86 | 16.32 | 21.23 |
|       | time(s)     | 0.07 | 0.16 | 0.54 | 2.81 |

At first glance, the results from Tables 1 and 2 may seem surprising since the 4 room case has more space to keep cool but nonetheless is able to do so at a lower cost than the 3 room case. This is due to an insulation effect that the 4 room configuration enjoys. With the activation of two coolers in the first six time steps, the room temperatures can stay within their deadbands for the
Table 2 Results obtained for the TCL problem with a 4-room configuration.

|                | Time steps: |     |     |     |     |
|----------------|-------------|-----|-----|-----|-----|
|                | 8           | 24  | 48  | 62  |     |
| Alg. 2         |             |     |     |     |     |
| obj.           | 0           | 9.24| 9.24| 11.69|     |
| time(s)        | 0.17        | 0.25| 0.39| 1.41 |     |
| iter.          | 2           | 2   | 2   | 2   |     |
| B-OA           |             |     |     |     |     |
| obj.           | 0           | 9.24| 9.24| 11.69|     |
| time(s)        | 0.14        | 28.34| 48.42| 1,610|     |
| B-BB           |             |     |     |     |     |
| obj.           | 0           | 9.24| 9.24| 11.69|     |
| time(s)        | 0.18        | 33.11| 56.83| 822.48|     |
| Gurobi         |             |     |     |     |     |
| obj.           | 0           | 9.24| 9.24| 11.69|     |
| time(s)        | 0.22        | 0.37| 0.51| 1.08 |     |
| CPLEX          |             |     |     |     |     |
| obj.           | 0           | 9.24| 9.24| 11.69|     |
| time(s)        | 0.07        | 0.09| 0.11| 0.48 |     |

Table 3 Results obtained for the TCL problem with a linear room configuration.

|                | Rooms: | 7 | 7 | 10 | 12 | 18 | 20 |
|----------------|--------|---|---|----|----|----|----|
|                | Time steps: | 8 | 24 | 48 | 48 | 36 | 24 | 24 |
| Alg. 2         |            |   |    |    |    |    |    |    |
| obj.           | 0          | 23.1| 28.02| 37.26| 41.88| 50.82| 55.44|     |
| time(s)        | 0.15       | 1.03| 4.533.4| 9.127.5| 8.294| 853.7| 16.658|     |
| iter.          | 2          | 2   | 2   | 2   | 2   | 2   | 2   |     |
| B-OA           |            |   |    |    |    |    |    |    |
| obj.           | 0          | 23.1| N/A | N/A | N/A | N/A | N/A | N/A |
| time(s)        | 0.16       | 51.34| N/A | N/A | N/A | N/A | N/A | N/A |
| B-BB           |            |   |    |    |    |    |    |    |
| obj.           | 0          | 23.1| 28.02| 37.26| 41.88| 50.82| 55.44|     |
| time(s)        | 0.18       | 690.23| 1,682.5| 3,076.0| 3,261.5| 2,999.1| 4,658.4|     |
| Gurobi         |            |   |    |    |    |    |    |    |
| obj.           | 0          | 23.1| 28.02| N/A | N/A | 41.88| 50.82| N/A |
| time(s)        | 0.25       | 0.93| 56.09| N/A | 56,842| 31,926| N/A | N/A |
| CPLEX          |            |   |    |    |    |    |    |    |
| obj.           | 0          | 23.1| 28.02| 37.26| 41.88| 50.82| N/A | N/A |
| time(s)        | 0.08       | 0.65| 150.31| 13,416| 12,885| 57,108| > 245,000|     |

The entire 48 hour period. In contrast, the 3 room configuration is more susceptible to the ambient temperature and requires more use of the coolers. This also seems to have increased the computational complexity of the problem and requires more time for the 3 room case to be solved than the 4 room case. It should be noted that several initializations were tested and the solution times were not significantly affected, implying that this was not the cause of the runtime differences in the two cases.

One of the advantages of using a distributed method is the ability to solve problems that would be otherwise intractable for a centralized solver. Tables 1 and 2 show results for cases containing up to 496 variables, but even larger problems may be considered. Table 3 shows results for a variety of time horizons and rooms. Here, the room configuration is instead arranged such that the
rooms are in a line. While unrealistic for most buildings, this setup is realistic for the temperature control of a train or rooms next to a corridor. Mathematically, this example differs somewhat from the other two. While the other problems have a significant amount of coupling between the control variables, the air is not the case here. This sparsity allows for Algorithm 2 to outperform both Gurobi and CPLEX (applied to the centralized problem).

4.3 Results for MIQP

If the comfort parameter $\gamma$ is larger than zero then Problem (23) is a convex MIQP$^5$ As in Section 4.2, the results of Algorithm 2 for each room configuration are compared with those obtained from Bonmin, Gurobi, and CPLEX. The value of $\gamma$ was chosen to be one to allow for an equal weighting of comfort and cost.

Shown in Figure 4.3 are the trajectories obtained for the three-room scenario with a temperature deviation penalization. In contrast to Figure 4.2, a quadratic penalty term is used to model discomfort caused by deviations from the set temperature. Indeed, the solution with $\gamma = 1$ yields a trajectory with a similar number of activations as when $\gamma = 0$ but with temperature trajectories that stay much closer to the middle of the deadband. Theoretically, the quadratic term should make the problem more computationally difficult, but in some cases both Gurobi and CPLEX actually require less time. In contrast, Algorithm 2 requires many more iterations than the MILP formulation. Future work could seek to use some of the cutting plane methods and other heuristics used by Gurobi and CPLEX to alleviate this issue. Furthermore, quadratic lower bounding functions could significantly reduce the number of iterations until convergence.

Fig. 3 Trajectories are defined as in Figure 4.2 for the three room scenario, except with the comfort parameter $\gamma = 1$.

$^5$ Convex in the same notion of convexity in MICP. That is, a problem where the continuous relaxation yields a convex quadratic program.
Table 4 Results obtained for Problem (23) with a 3-room configuration.

| Time steps: | 8  | 24 | 48  | 62  |
|-------------|----|----|-----|-----|
| Alg. 2      |    |    |     |     |
|             | obj. | 17.83 | 60.15 | 104.07 | 134.43 |
|             | time(s) | 0.24 | 0.49 | 1.36 | 3.17 |
|             | iter. | 4 | 3 | 3 | 3 |
| B-OA        |    | 17.83 | 60.15 | 104.07 | 134.43 |
|             | time(s) | 4.36 | 46.94 | 399.49 | 1,702.60 |
| B-BB        |    | 17.83 | 60.15 | 104.07 | 134.43 |
|             | time(s) | 9.09 | 70.40 | 502.76 | 1,152.90 |
| Gurobi      |    | 17.83 | 60.15 | 104.07 | 134.43 |
|             | time(s) | 0.44 | 0.50 | 0.66 | 0.83 |
| CPLEX       |    | 17.83 | 60.15 | 104.07 | 134.43 |
|             | time(s) | 0.19 | 0.27 | 0.25 | 0.54 |

Table 5 Results obtained for Problem (23) with a 4-room configuration.

| Time steps: | 8  | 24 | 48  | 62  |
|-------------|----|----|-----|-----|
| Alg. 2      |    |    |     |     |
|             | obj. | 21.68 | 52.26 | 99.68 | 134.43 |
|             | time(s) | 0.37 | 0.65 | 8.82 | 31.07 |
|             | iter. | 5 | 3 | 3 | 4 |
| B-OA        |    | 21.68 | 52.26 | 99.68 | 134.43 |
|             | time(s) | 14.61 | 6.78 | 613.95 | 7,024.10 |
| B-BB        |    | 21.68 | 52.26 | 99.68 | 134.43 |
|             | time(s) | 30.39 | 11.13 | 661.95 | 1,495.30 |
| Gurobi      |    | 21.68 | 52.26 | 99.68 | 134.43 |
|             | time(s) | 0.41 | 0.54 | 0.72 | 2.31 |
| CPLEX       |    | 21.68 | 52.26 | 99.68 | 134.43 |
|             | time(s) | 0.15 | 0.16 | 0.41 | 2.15 |

4.4 Higher Order Convex Problems

One of the advantages of the proposed algorithm is that it is applicable to a relatively large class of problems (namely, MICPs). While Section 4.3 shows favourable results for both Gurobi and CPLEX, if the problem were adjusted slightly such that it were no longer an MIQP then these solvers would no longer be applicable. For example, if the objective function of Problem (23) became

$$
\min_{T(\cdot), u(\cdot)} \sum_{t=0}^{H-1} c(t) u(t) + \gamma (T_i(t) - T_{ref}(t))^4,
$$
Table 6 Results obtained for Problem (23) with a linear 7 room configuration.

| Time steps | Alg. 2 | B-OA | B-BB | Gurobi | CPLEX |
|------------|--------|------|------|--------|-------|
|            | obj.   | time(s) | obj.   | time(s) | obj.   | time(s) |
| 8          | 39.50  | 1.57  | 39.50 | 358.12 | 39.50 | 593.17  |
| 24         | 112.08 | 2.55  | 112.08| 1,833.9| 112.08| 955.77  |
| 48         | 201.19 | 4     | 201.19| 6,095.6| 201.19| 6,095.6 |
| 62         | 267.04 | 3     | 267.04| 15,582 | 267.04| 15,582  |

then this would still be solvable via PaDOA, but not Gurobi or CPLEX. However, Bonmin can still be applied. The results for a variety of such problem configurations are shown below in Figure 7. Therein it can be observed that Algorithm 2 returns the same, global solution as Bonmin, and does so in less time. The runtime difference is particularly striking for the 7 room scenarios as these contain the most variables and have the greatest potential for parallelization.

Table 7 Results obtained for Problem (23), but with a 4th order objective function.

| Rooms | Time steps | Alg. 2 | B-B&B | B-OA | B-B&B | B-OA |
|-------|------------|--------|-------|------|-------|------|
|       |            | obj.   | time(s) | iter. | obj.   | time(s) | obj.   | time(s) |
| 3     | 8          | 16.72  | 4.20  | 5     | 16.72  | 4.20  | N/A   | N/A   |
| 3     | 24         | 76.86  | 19.14 | 4     | 76.86  | 30.86 | N/A   | N/A   |
| 3     | 48         | 115.62 | 95.56 | 3     | 115.62 | 547.98| N/A   | N/A   |
| 3     | 62         | 141.44 | 405.19| 6     | 141.44 | 779.41| N/A   | N/A   |
| 4     | 8          | 21.73  | 4.11 | 6     | 21.73  | 16.93 | N/A   | N/A   |
| 4     | 24         | 44.94  | 6.47  | 3     | 44.94  | 9.93  | N/A   | N/A   |
| 4     | 48         | 86.72  | 151.58| 5     | 86.72  | 477.60| N/A   | N/A   |
| 4     | 62         | 120.71 | 175.78| 6     | 120.71 | 781.46| N/A   | N/A   |
| 7     | 8          | 38.41  | 8.87 | 7     | 38.41  | 374.38| N/A   | N/A   |
| 7     | 24         | 122.09 | 34.86| 3     | 122.09 | 1291.29| N/A   | N/A   |
| 7     | 48         | 202.54 | 1631.6| 4     | 202.54 | 2830.93| N/A   | N/A   |
| 7     | 62         | 263.91 | 3614.54| 4     | 263.91 | 9453.89| N/A   | N/A   |

6 It should be noted that all results seen in this section for Algorithm 2 use Bonmin to solve the MICP subproblems.

7 With the Branch and Bound sub-algorithm.
Table 8 Runtime breakdown of Algorithm 2 applied to the 2nd-order version of Problem 23 with 7 room TCL problem with 48 time steps.

| Iter. 1 | Iter. 2 | Iter. 3 | Iter. 4 |
|---------|---------|---------|---------|
| MINLP time (s) | 0.27 | 0.20 | 0.20 | 0.20 |
| MILP time(s) | 2.20 | 86.89 | 82.62 | 87.62 |
| Hyperplane time (s) | 0.001 | 0.005 | 0.002 | 0.002 |

Fig. 4 Progression of the upper and lower bounds during each iteration while solving the 2nd-order version of Problem 23 with 7 rooms and 48 time steps. The blue line depicts the progression of the upper bound and the red shows that of the lower bound.

4.5 Outlook

The implementation of Algorithm 2 used in Section 4 was prototypical and could be improved in a number of ways. Interestingly, as Table 8 shows, the majority of the time spent by the algorithm was typically on the coupling problems. However, more time was required for the solution of the MICP subproblems in the higher order version of the problem. An example of which is shown in Table 9 along with the relevant convergence plot.

The reason why the linear approximations required so much time to be solved likely lies in the heuristics and presolving processes used by Gurobi. This is despite the fact that Gurobi was used to solve both the linear approximations and the full nonlinear program.

5 Conclusions

This paper has introduced the partially distributed outer approximation method PaDOA (Algorithm 2) for finding ε-suboptimal points of the struc-
Table 9 Runtime breakdown of Algorithm 2 applied to the 4th-order version of Problem 23 with 4 rooms and 8 time steps.

|                | Iter. 1 | Iter. 2 | Iter. 3 | Iter. 4 | Iter. 5 | Iter. 6 |
|----------------|---------|---------|---------|---------|---------|---------|
| MINLP time (s) | 0.73    | 0.75    | 0.67    | 0.65    | 0.65    | 0.66    |
| MILP time (s)  | 0.03    | 0.04    | 0.04    | 0.04    | 0.04    | 0.04    |
| Hyperplane time (s) | 0.004 | 0.006 | 0.002 | 0.002 | 0.002 | 0.002 |

Fig. 5 Progression of the upper and lower bounds during each iteration while solving the 4th-order version of Problem 23 with 4 rooms and 8 time steps. The blue line depicts the progression of the upper bound and the red shows that of the lower bound.

tured MICP \(^1\). PaDOA proceeds by alternating between solving partially de-coupled MICPs that comprise \(n\) integer variables and large scale MILPs with \(nN\) variables. Finite termination conditions for PaDOA have been established in Theorem 3. Moreover, we have discussed the major theoretical and practical advantages of PaDOA compared to existing extended formulation based OA solvers. In particular, Theorem 2 states that PaDOA terminates after the first iteration, if it is initialized at a global minimizer—an important property that is neither shared by existing OA nor by existing branch-and-bound based methods for MICP.

In Section 4, first, second and fourth order mixed integer problems were used to demonstrate the practical performance of PaDOA compared to other state of the art solvers by application to a scheduling problem of thermostatically controlled loads. While the solution and runtime are competitive for each of the case studies considered, the best performance was observed for problems with sparse Hessians and coupling constraints. Furthermore, it was observed that PaDOA was able to return a solution in several cases where the centralized approach could not due to memory constraints.

Future work will investigate the use of piecewise linear underapproximations and/or cutting planes in the MILP step to extend PaDOA to non-convex MINLPs. Furthermore, Step 4 requires full constraint information in order to
return a feasible solution. This restricts the applicability in terms of fully distributed settings and future work will focus on sidestepping this restriction.

References

1. N. Alguacil, A.L. Motto, and A.J. Conejo. Transmission expansion planning: A mixed-integer lp approach. *IEEE Transactions on Power Systems*, 18(3):1070–1077, 2003.
2. J.A.E. Andersson, J. Gillis, G. Horn, J.B. Rawlings, and M. Diehl. CasADi—A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.
3. R. Andreani, E.G. Birgin, J.M. Martinez, and M.L. Schuverdt. On augmented lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, 18:1286–1309, 2007.
4. J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.
5. D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.
6. P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, may 2008.
7. P. Bonami, M. Kiliç, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. In *Mixed Integer Nonlinear Programming*, volume 154, pages 1–39. Springer, New York, 2012.
8. B. Borchers and Mitchell J.E. An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers & Operations Research*, 21:359–368, 1994.
9. S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.
10. M. Carrion and J. M. Arroyo. A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems*, 21(3):1371–1378, 2006.
11. M. Conforti, G. Cornuéjols, and G. Zambelli. Polyhedral approaches to mixed integer linear programming. In *50 Years of Integer Programming 1958–2008*, pages 343–385. Springer, New York, 2009.
12. G.B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.
13. Deutscher Wetterdienst. [ftp://ftp-cdc.dwd.de/pub/CDC/observations_germany/climate/10_minutes/solar/historical/](ftp://ftp-cdc.dwd.de/pub/CDC/observations_germany/climate/10_minutes/solar/historical/) 2017.
14. M.A. Duran and I.E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3):307–339, oct 1986.
15. J. Eckstein and D.P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55:293–318, 1992.
16. H. Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3):399–417, 1963.
17. R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66(1-3):327–349, aug 1994.
18. D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximations. *Computers and Mathematics with Applications*, 2:17–40, 1976.
19. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness* (*Seres of Books in the Mathematical Sciences*). W. H. Freeman, first edition edition, 1979.
20. A. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.
21. R.E. Gomory. An algorithm for the mixed integer problem, 1960.
22. O.K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31:1533–1546, 1985.

23. H. Hijazi, P. Bonami, and A. Ouorou. An outer-inner approximation for separable mixed-integer nonlinear programs. *INFORMS Journal on Computing*, 26(1):31–44, 2014.

24. B. Houska, J. Frasch, and M. Diehl. An augmented lagrangian based algorithm for distributed Non-Convex optimization. *SIAM Journal on Optimization*, 26(2):1101–1127, Jan 2016.

25. IBM. Using the CPLEX callable library, version 12. 2009.

26. P. Kesavan, R.J. Allgor, E.P. Gatzke, and P.I. Barton. Outer approximation algorithms for separable nonconvex mixed-integer nonlinear programs. *Mathematical Programming*, 100(3):517–535, 2004.

27. M.R. Kılınç, J. Linderoth, and J. Luedtke. Lift-and-project cuts for convex mixed integer nonlinear programs. *Mathematical Programming Computation*, 9(4):499–526, 2017.

28. S. Koch, J.L. Mathieu, and D.S. Callaway, editors. *Modeling and Control of Aggregated Heterogeneous Thermodynamically Controlled Loads for Ancillary Services*. Proc. PSCC, 2011.

29. B. Kocuk, S.S. Dey, and X. Sun. New formulation and strong MISOCp relaxations for AC optimal transmission switching problem. *IEEE Transactions on Power Systems*, 32(6):4161–4170, 2017.

30. P. Kohlehepp and V. Hagenmeyer. Technical potential of buildings in Germany as flexible power-to-heat storage for smart-grid operation. *Energy Technology*, 5(7):1084–1104, 2017.

31. S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and T. Koolen. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots*, 40(3):429–455, 2016.

32. S. Leyffer. Integrating sqp and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization & Applications*, 18:295–309, 2001.

33. M. Lubin, E. Yamangil, R. Bent, and J.P. Vielma. Polyhedral approximation in mixed-integer convex optimization. *Mathematical Programming*, 2017.

34. A. Murray, T. Faulwasser, and V. Hagenmeyer. Mixed-integer vs. real-valued formulations of distributed battery scheduling problems. In *10th Symposium on Control of Power and Energy Systems (CPES 2018)*, 2018.

35. K.G. Murty and S.N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.

36. I. Necoara and J.A.K. Suykens. Application of a smoothing technique to decomposition in convex optimization. *IEEE Transactions on Automatic Control*, 53(11):2674–2679, 2008.

37. J. Nocedal and S.J. Wright. *Sequential quadratic programming*. Springer, 2006.

38. Gurobi Optimization. Gurobi optimizer reference manual, 2009.

39. M.J.D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*. Academic Press, 1969.

40. I. Quesada and I.E. Grossmann. An lp/nlp based branch-and-bound algorithm for convex minlp optimization problems. *Computers and Chemical Engineering*, 16:937–947, 1992.

41. D.E. Ravemark and Rippin W.T. Optimal design of a multi-product batch plant. *Computers & Chemical Engineering*, 22:177–183, 1998.

42. R.T. Rockafellar. Convex analysis, 1970.

43. J. Ruckmann and A. Shapiro. Augmented lagrangians in semi-infinite programming. *Mathematical Programming, Series B*, 116:499–512, 2009.

44. N. Sawaya. *Reformulations, relaxations and cutting planes for generalized disjunctive programming*. PhD thesis, Carnegie Mellon University, 2006.

45. A. Shapiro and J. Sun. Some properties of the augmented lagrangian in cone constrained optimization. *Mathematics of Operations Research*, 29(3):479–491, 2004.

46. R. Takapoui, N. Möhle, S. Boyd, and A. Bemporad. A simple effective heuristic for embedded mixed-integer quadratic programming. *International Journal of Control*, pages 1–11, Apr 2016.

47. M. Tawarmalani and N.V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.
A Proofs

A.1 Proof of Proposition 1

If there is no $x \in X$ with $Ax = b$, both sides of (6) are equal to infinity and the statement of the proposition holds in the extended value sense. Thus, we may assume that the constraints in (6) are feasible. Consequently, Assumption 1 implies that (5) is a convex optimization problem with compact and non-empty feasible set. Moreover, since $X$ is a polytope, all constraints in (5) are linear. It is well-known [5,42] that strong duality holds under these conditions. □

A.2 Proof of Lemma 1

If there is no $x \in X$ with $Ax = b$, both sides of (10) are equal to infinity and the statement of the lemma holds in the extended value sense. Thus, we may assume that the equation $Ax = b$ has a solution in $X$. Next, because we have $z \in \Xi$, our particular construction of $\Phi$ implies that

$$\Phi(x, z, \Xi) \geq f^\star(z) + [\lambda^\star(z)]^T (x - x^\star(z)) + [\mu^\star(z)]^T (z - z) = 0$$

Thus, we have

$$\min_{x \in X, Ax = b} \Phi(x, z, \Xi) = \min_{y \in X, Ay = b} f^\star(z) + [\lambda^\star(z)]^T (y - x^\star(z))$$

$$= \min_{x, y} f(x, z) + [\lambda^\star(z)]^T (y - x) \quad \text{s.t.} \quad \begin{cases} Ay = b \\ y \in X \end{cases}$$

$$= \max_{\lambda} \min_{x, y} f(x, z) + \lambda^T (y - x) \quad \text{s.t.} \quad \begin{cases} Ay = b \\ y \in X \end{cases}.$$  

Since Assumption 1 holds, we may substitute (6) (see Proposition 1), which yields the equation

$$\forall z \in \Xi, \quad \min_{x \in X, Ax = b} \Phi(x, z, \Xi) = f^\star(z).$$

□
A.3 Proof of Theorem 1

Notice that if the equation $Ax = b$ has no solution in $X$, this will be detected immediately by Step 2 of Algorithm 1, which causes termination. Thus, we may assume that all optimization problems are feasible. Now, the main idea of the proof is to show that the cardinality of the set $\Pi$ is strictly increasing in every iteration, if the algorithm does not terminate. For this aim, we first notice that any solution $(x^+, y^+, z^+)$ of the MILP (12) satisfies the equation
\[
\Phi(x^+, z^+, \Pi) = \sum_{i=1}^{N} y_i^+
\] (25)
by construction. Moreover, because we have $Ax^+ = b$, the inequality
\[
\min_{x, Ax=b} \Phi(x, z^+, \Pi) \leq \Phi(x^+, z^+, \Pi)
\] (26)
holds. If we further assume that the termination criterion is not satisfied, we must have
\[
\sum_{i=1}^{N} y_i^+ < U - \epsilon
\] (27)
Thus, if we had $z^+ \in \Xi$, then the result of Lemma 1 would imply that
\[
f^*(z^+) \leq \min_{x, Ax=b} \Phi(x, z^+, \Pi)
\] (28)
as well as $U \leq f(z^+)$, since $z^+$ has already been added to the collection $\Pi$. By substituting all the above relations we would then find that
\[
f^*(z^+) \leq \Phi(x^+, z^+, \Pi) \leq U - \epsilon \leq f(z^+) - \epsilon,
\]
which is a contraction. Thus, either our assumption that the algorithm does not terminate or our assumption $z^+ \in \Xi$ must be wrong. In other words, if the algorithm does not terminate in the current step, then the cardinality of the set $\Pi$ increases by 1 in the next step, because $z^+$ is added to the collection $\Pi \subseteq Z$. But this is only possible for a finite number of steps, because the set $Z$ contains only a finite number of points. Thus, Algorithm 1 must terminate after a finite number of iterations. □

A.4 Proof of Theorem 2

Let $V^* = \sum_{i=1}^{N} f_i(x^*_i, z^*_i)$ denote the optimal value of (1). Because we assume that such an optimal solution exists while Assumption 1 is satisfied, the partially decoupled optimization problems are all feasible and return piecewise affine lower bounds that satisfy the termination condition (17) with $V_k(z^*) = V^*$, i.e., we have
\[
V^* - \epsilon \leq \min_{x \in X, \zeta \in Z_k} \Theta_k^*(x, \zeta) \quad \text{s.t.} \quad Ax = b
\] (29)
for all $k \in \{1, \ldots, N\}$. Because the function $\Theta$ is by construction an upper bound on $\Theta_k$ (for any $k$), we further have
\[
\min_{x \in X, \zeta \in Z_k, Ax=b} \Theta_k^*(x, \zeta) \leq \min_{x \in X, Ax=b} \Theta(x, z) = \Theta(x^+, z^+)
\]
where $(x^+, z^+)$ denotes the solution of the master MILP (21). By substituting the above inequalities we find that
\[
V^* - \epsilon \leq \Theta(x^+, z^+).
\]
Because we assume that $\epsilon \leq \epsilon$, this implies that
\[
U - \Theta(x^+, z^+) = V^* - \Theta(x^+, z^+) \leq \epsilon \leq \epsilon
\]
Thus, the termination condition is satisfied and Algorithm 2 terminates after the first step. □
A.5 Proof of Theorem 3

We may assume that the coupled equality constraint is feasible, as infeasibility would be detected immediately in Step 1 of Algorithm 2. Similar to the proof of Theorem 1, we need to keep track of the integer solutions of the master MILPs. For this aim, we introduce the following “artificial” additional step:

\textbf{Step 3'}: After solving (21), update $\tilde{\Pi} = \tilde{\Pi} \cup \{z^+\}$.

If the set $\tilde{\Pi}$ is initialized with the empty set and if Step 3' is inserted in Algorithm 2 immediately after Step 3, the iterates of this algorithm remain unaffected. The main idea of the proof is now to show that the cardinality of the set $\tilde{\Pi}$ increases in every iteration of Algorithm 1 under the assumption that the termination criterion is not satisfied. Let us assume that the solution $z^+$ satisfies $z^+ \in \Pi$ (before $\Pi$ is updated in Step 3’). Then we have

\[ U \leq V_k(z^+) \leq \epsilon_L + \min_{x \in X, \zeta \in Z_k, Ax = b} \Theta^*_k(x, \zeta), \]

Because $\Theta$ is an upper bound on $\Theta_k$, this implies that we also have

\[ \min_{x \in X, \zeta \in Z_k, Ax = b} \Theta^*_k(x, \zeta) \leq \min_{x \in X, \zeta \in Z_k, Ax = b} \Theta^*(x, \zeta) = \Phi(x^+, z^+), \]

which yields $U - \Phi(x^+, z^+) \leq \epsilon_L \leq \epsilon$. Thus, either the termination criterion is satisfied or we have $z^+ \notin \Pi$. In the latter case, the cardinality of $\Pi$ increases by 1 in the current iteration. As this is only possible for a finite number of steps, Algorithm 2 must terminate. $\Box$