An adaptive relaxation-refinement scheme for multi-objective mixed-integer nonconvex optimization

Gabriele Eichfelder[*], Moritz Link[†], Stefan Volkwein[†], Leo Warnow[*]

Abstract

In this work, we present an algorithm for computing an enclosure for multi-objective mixed-integer nonconvex optimization problems. In contrast to existing solvers for this type of problem, this algorithm is not based on a branch-and-bound scheme but rather relies on a relax-and-refine approach. While this is an established technique in single-objective optimization, several adaptions to the multi-objective setting have to be made in order to exploit the full potential of this idea. To that end, we propose an intensified individualization of the relaxations to the respective parts in the image space resulting in a novel adaptive box-based relaxation technique for nonconvex terms. We provide numerical tests for the new algorithm that show both its strength and limitations.

Key Words: multi-objective optimization, nonconvex optimization, mixed-integer optimization, enclosure, adaptive refinement.

Mathematics subject classifications (MSC 2010): 90C11, 90C26, 90C29, 90C59

1 Introduction

Multi-objective mixed-integer optimization problems often arise in real-world applications due to their ability to respect conflicting objectives [17, 18, 28]. Unsurprisingly, algorithmic treatment of linear as well as nonlinear versions of such problems has gained a lot of attention recently; cf., e.g., [6, 14, 15, 23, 28, 34]. A general difficulty in solving multi-objective optimization problems algorithmically is the presence of a potentially infinite number of optimal (value) points [12] – in the image space as well as in the preimage space. While having infinitely many global optimal solutions can also happen in the single-objective context, they all have the same optimal value. In contrast to that, it is the core idea of even considering multiple objective functions to get an as good as possible overview of all the different optimal value vectors – the so-called nondominated set (cf. Definition 2.1) – in the image space. Consequently, even when solely interested in finding the optimal value vectors, the corresponding solutions in the

*Institute for Mathematics, Technische Universität Ilmenau, Po 10 05 65, D-98684 Ilmenau, Germany, `{gabriele.eichfelder,leo.warnow}@tu-ilmenau.de`

†Department of Mathematics and Statistics, University of Konstanz, Universitätsstraße 10, 78464 Konstanz, Germany, `{moritz.link, stefan.volkwein}@uni-konstanz.de`

preimage space can be scattered widely across the feasible region. For instance, being in the mixed-integer context, this can lead to the fact that many or even all integer assignments could contribute to different optimal value vectors.
Unlike linear problems, it is in general not possible anymore to determine the full nondominated set when dealing with nonlinear multi-objective optimization problems. Therefore, similar to [14, 15, 16, 13, 28], our approach seeks to provide such an overview in form of a superset – an enclosure – of the nondominated set (cf. Definition 3.1). The underlying idea of such algorithms is successively shrinking an initially given set up to a predetermined tolerance while staying a superset of the nondominated set. Since our algorithm treats θ-feasible points [21] in the same way as feasible points it is not guaranteed that the output is still an enclosure. This gives rise to the novel concept of *pseudo enclosures* (cf. Definition 3.2).
While [14] utilizes a branch-and-bound framework for computing an enclosure for multi-objective mixed-integer nonconvex problems, we tackle this problem class with a fundamentally different approach. To be more precise, we extend the approach introduced in [28], where only nonconvex quadratic functions are allowed, to general nonconvex problems. Instead of utilizing a branch-and-bound tree, the method from [28] is based on ideas presented in [8, 31]. These single-objective methods successively refine mixed-integer linear outer approximations to increase accuracy and therefore get tighter and tighter lower bounds for the optimal value of the original problem. Such approaches particularly benefit from a problem structure, where the dimensions of the variable domains of the respective nonlinear functions are comparably small. For the remainder of this work, we refer to this approach as *relax-and-refine* approach. In the single-objective literature, there are plenty of different methods for solving MINLPs. We refer to [3, 26] for a detailed overview.
The reason for considering relaxations in relax-and-refine approaches is an expected advantage regarding the algorithmic handling compared to the original problem. As already mentioned, we utilize mixed-integer linear relaxations, and therefore our algorithm heavily relies on the availability of strong solvers for mixed-integer linear problems. However, as the relaxed problems get more complex with ongoing refinement – and possibly even too complex for the strong solvers –, a crucial factor for the numerical efficiency of relax-and-refine approaches is preventing the relaxed problems from getting too complex. In this paper – besides extending to general nonconvex functions –, we enhance the method from [28] with two features to promote low complexity of the relaxed problems. Namely, adaptivity in the refinement process and applying domain reduction techniques during the procedure – both well-known from the single-objective literature.
In [8, 31], the refinement process is concentrated on the parts of the preimage space that seem to influence the optimal value the most. As already encountered, these parts can cover large areas of the – sometimes even full – feasible set when considering multi-objective problems. Hence, naively applying this idea to multi-objective problems may lose its effect.
However, to keep the spirit of this idea, we propose an approach that individualizes the outer approximation problems with ongoing algorithmic progress to the specific optimal regions in the image space. The underlying hope is that rather small parts of the feasible set are relevant when approaching a specific part of the nondominated set. While in [28], there was only an individualized decision for refinement of the relaxations, we go one step further in the present work and also individualize the shape of the outer

approximation according to the regions in the image space. For reasons of numerical comparison, we present two variants of our approach: one using a uniform refinement scheme similar to [28], and one being able to concentrate on the respective seemingly optimal regions transferring the spirit of [8, 31] to the multi-objective setting. We call them the *uniform* and *adaptive* approach, respectively.
The same hope – rather small parts of the feasible set become relevant when only considering a specific part of the nondominated set – motivates the use of domain reduction techniques *during* the procedure. While this is a widespread tool for single-objective tree-based algorithms [20, 35], it is novel for the kind of relax-and-refine approaches like [8, 28, 31]. For instance, [31] employs so-called *sequential bound tightening* before starting the refinement process based on domain partitioning. Despite this works very well for the single-objective case, we again have the problem that possibly only small parts of the feasible set can be dropped without specifying a region of interest in the image space. This is exactly how our algorithm works in the image space: it successively specifies regions of interest – so-called search zones – in the image space. As the simplex-based relaxation technique presented in [8] has limitations when combined with dynamic bound tightening, we introduce a novel relaxation technique for general nonlinear functions based on box representations. Note that also [21, 31] utilize a box-based partitioning scheme although their problem formulation differs from ours and no dynamic bound tightening is applied.
The remainder of this work is structured as follows. In Section 2 we introduce basic notations and definitions. Afterwards, in Section 3 the main algorithmic framework is described and its finiteness and correctness are proven in Section 4. The box-based relaxation technique is presented in Section 5 whereas the relaxation-refinement schemes and the bound tightening techniques are provided in Section 6. Numerical experiments are given in Section 7.

2 Definitions and notations

Throughout this paper, for vectors $x, y \in \mathbb{R}^r$, the inequalities $x \leq y$ and $x < y$ are understood componentwise. The problems which we consider in the present paper are of the following form:

$$\min\ f(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_r(x) \end{pmatrix} \quad \text{s.t.} \quad \left\{ \begin{array}{c} Ax \leq b, \\ g_j(x_{\mathcal{J}_j}) \leq 0 \text{ for } j \in [p], \\ x \in X, \\ x_i \in \mathbb{Z} \text{ for } i \in I \end{array} \right. \tag{MOP}$$

with continuous objective functions $f_i \colon \mathbb{R}^n \to \mathbb{R}$, $i \in [r] := \{1, \dots, r\}$, continuous nonlinear constraint functions $g_j \colon \mathbb{R}^{l_j} \to \mathbb{R}$, $j \in [p]$, where $l_j = |\mathcal{J}_j|$ is the number of g_j's arguments and $\mathcal{J}_j \subseteq [n]$ the index set of the variables on which g_j depends. The vector $x_{\mathcal{J}_j}$ collects those components of the vector $x \in \mathbb{R}^n$ with $j \in \mathcal{J}_j$. Further, we have $A \in \mathbb{R}^{q \times n}$, $b \in \mathbb{R}^q$, and an index set $I \subseteq [n]$ determining the integer variables. The vector $x_I \in \mathbb{R}^{|I|}$ collects those components of the vector $x \in \mathbb{R}^n$ with $i \in I$. We introduce the compact n-dimensional box $X = [x^\ell, x^u] := \{x \in \mathbb{R}^n \mid x^\ell \leq x \leq x^u\}$ describing the box constraints of the variables, where $x^\ell, x^u \in \mathbb{R}^n$ and $x^\ell < x^u$ hold. Without loss of generality, we may assume $x^\ell_I, x^u_I \in \mathbb{Z}^{|I|}$. If $I = \emptyset$, i.e., if there are no integrality constraints for any variable, we call (MOP) a *continuous* multi-objective optimization problem, and if $I = [n]$ we call it an *integer* multi-objective optimization problem. If $r = 1$ we call (MOP) a *single-objective* optimization problem. Note that the components of the occurring functions f and g are neither assumed to be linear nor convex. However, without loss of generality, we assume that the f_i's are linear and the g_j's are nonlinear. We denote the feasible set of (MOP) by

$$S := \left\{x \in X \mid Ax \leq b,\ g_j(x_{\mathcal{J}_j}) \leq 0 \text{ for } j \in [p],\ x_I \in \mathbb{Z}^{|I|}\right\}.$$

Allowing a small violation of the nonlinear constraints determined by some prescribed tolerance $\theta > 0$ yields another important set for the present work. Similar as in [21], the so-called *θ-feasible set* is given by

$$S^\theta := \left\{x \in X \mid Ax \leq b,\ g_j(x_{\mathcal{J}_j}) \leq \theta \text{ for } j \in [p],\ x_I \in \mathbb{Z}^{|I|}\right\} \supseteq S.$$

Note that even for seemingly small θ, the set S^θ can be very large compared to S. Replacing the original feasible set S in (MOP) by S^θ results in the following optimization problem

$$\min f(x) \quad \text{s.t.} \quad x \in S^\theta, \tag{MOP(θ)}$$

which we call the *θ-feasible version* of (MOP). Due to the continuity of g together with the compactness of X, we have that S and S^θ are compact sets. Further, we assume the set S to be nonempty which then yields compact nonempty image sets $f(S)$ and $f(S^\theta)$. For a given integer assignment $\hat{x}_I \in \mathbb{Z}^{|I|}$ we define the set $S_{\hat{x}_I} = \{x \in S \mid x_I = \hat{x}_I\}$, as well as the corresponding continuous (possibly nonconvex) patch problem

$$\min f(x) \quad \text{s.t.} \quad x \in S_{\hat{x}_I}. \tag{pMOP($\hat{x}_I$)}$$

Note that we have $S_{\hat{x}_I} \neq \emptyset$ (and therefore feasibility of (pMOP($\hat{x}_I$))) if and only if $\hat{x}_I$ is a so-called feasible integer assignment of (MOP).
Generally, a multi-objective optimization problem is characterized by the presence of $r > 1$ conflicting objective functions. *Conflicting* means that, in general, there exists no feasible point that minimizes all of these r objective functions at the same time. This motivates the concept of *(non-)dominance* and *efficiency* (cf. [12]).

Definition 2.1 *(i) Let $y^1, y^2 \in \mathbb{R}^r$. Then, y^1* dominates *y^2 if $y^1 \leq y^2, y^1 \neq y^2$.*

(ii) Let $y \in \mathbb{R}^r$ and let $N \subseteq \mathbb{R}^r$. We say that the set N is nondominated *given y if there exists no $\hat{y} \in N$ such that y dominates $\hat{y}$.*

We call a point $\bar{y} \in f(S)$ a *nondominated point* of (MOP) if there exists no $y \in f(S)$ dominating $\bar{y}$. The set of nondominated points of (MOP) is called its *nondominated set* and we denote it by $\mathcal{N}$. Analogously, we denote the nondominated set of (MOP(θ)) by $\mathcal{N}^\theta$. Given $x \in S$ with $f(x) \in \mathcal{N}$ we call x an *efficient solution* of (MOP).
Similar to numerical methods for single-objective optimization, our method produces only approximations of the optimal value, i.e., of the nondominated points. This gives rise to the extension of ε-optimality to the multi-objective setting. For that we use the *all-one vector* $e := (1, \dots, 1)^\top \in \mathbb{R}^r$.

Definition 2.2 *Let $\varepsilon > 0$. A point $\bar{y} \in f(S)$ is called an ε-*nondominated point *of (MOP) if there exists no $y \in f(S)$ such that $y + \varepsilon e$ dominates $\bar{y}$.*

Instead of the all-one vector e one could also use another vector. This could be useful if, for example, the magnitudes of the objective functions differ largely as this is often the case for applications. However, one could also normalize the respective objective functions and stick to the all-one vector.

3 Main algorithm for computing an enclosure

The main objective of our algorithm is to provide an overview of the nondominated set $\mathcal{N}$ of (MOP). This is realized by computing both, a set intended to cover (or enclose) the nondominated set as well as a collection of points intended to represent nondominated points. As it turns out, we cannot guarantee full coverage of the nondominated set which is why we call such a set a pseudo enclosure. The novel concept of pseudo enclosures is closely related to and based on the concept of enclosures of the nondominated set. As discussed in [13], enclosures are a very natural extension of the approximation concept used in single-objective global optimization. While enclosures were introduced in [13] using nonempty and compact sets of lower and upper bounds, we use here the definition as in [15, 16, 28] which assumes these sets to be finite.

Definition 3.1 *Let $L, U \subseteq \mathbb{R}^r$ be two finite sets with $\mathcal{N} \subseteq L + \mathbb{R}^r_+$ and $\mathcal{N} \subseteq U - \mathbb{R}^r_+$. Then L is called a* lower bound set*, U is called an* upper bound set*, and the set E which is given as*

$$E = E(L, U) := (L + \mathbb{R}^r_+) \cap (U - \mathbb{R}^r_+) = \bigcup_{l \in L} \bigcup_{\substack{u \in U, \\ l \leq u}} [l, u]$$

is called the enclosure *of the nondominated set $\mathcal{N}$ of (MOP) given L and U.*

In contrast to the above, in this work, we construct a set U for which we cannot guarantee that $\mathcal{N} \subseteq U - \mathbb{R}^r_+$, but only $\mathcal{N}^\theta \subseteq U - \mathbb{R}^r_+$ for some predetermined $\theta > 0$. The role of the parameter θ throughout this work is to determine a tolerance for the violation of the nonlinear constraints. More precisely, whenever we encounter a point $x \in S^\theta$, we consider it as feasible for the original problem (MOP) and treat it accordingly. This idea has already been used in [8, 21] for the single-objective case and in [28] for the multi-objective one. E.g., in [21] the authors call a global solution of a single-objective version of (MOP(θ)) an *approximate global solution*. This gives rise to an *approximate nondominated set* $\mathcal{N}^\theta$ in the multi-objective setting. Since (MOP(θ)) does not have a simpler structure than (MOP) – and therefore, $\mathcal{N}^\theta$ is not easier to compute, approximate or enclose than $\mathcal{N}$ –, in [28] the authors choose a uniquely determined set $\tilde{S}$ with $S \subseteq \tilde{S} \subseteq S^\theta$, compute an enclosure of the nondominated set $\tilde{\mathcal{N}}$ corresponding to $\tilde{S}$, and declare it to be an enclosure also of $\mathcal{N}$ due to the arbitrarily small choice of θ. Now, by leaving the path of uniform refinement procedures – which is one key contribution of the present work (see Sections 5 and 6 for details) –, the unique choice of such a set $\tilde{S}$ is not possible anymore. This leads to the fact that we do neither compute an enclosure of $\mathcal{N}$ nor of $\mathcal{N}^\theta$ nor of some nondominated set corresponding to any a-priorily explicitly specified superset of S. This motivates the introduction of so-called *pseudo enclosures*.

Definition 3.2 *Let $\theta > 0$ and let $L, U^\theta \subseteq \mathbb{R}^r$ be two finite sets with $\mathcal{N}^\theta \subseteq U^\theta - \mathbb{R}^r_+$ and L a lower bound set of $\mathcal{N}$, i.e., $\mathcal{N} \subseteq L + \mathbb{R}^r_+$. Then, U^θ is called a θ-*pseudo upper bound set*, and the set E^θ given as*

$$E^\theta = E(L, U^\theta) = \bigcup_{l \in L} \bigcup_{\substack{u \in U^\theta, \\ l \leq u}} [l, u]$$

*is called a θ-*pseudo enclosure *of the nondominated set $\mathcal{N}$ of (MOP) given L and U^θ. If no confusion is likely to arise, we just speak of pseudo enclosures and pseudo upper bound sets, and denote them by E and U, respectively.*

Note that following Definition 3.2, in contrast to Definition 3.1, it is not ensured that pseudo enclosures are in general nonempty. Indeed, it could be the case that there is no single pair $(\ell, u) \in L \times U$ with $\ell \leq u$. Hence, the respective algorithm has to take care that one does not end up with an empty pseudo enclosure by, e.g., ensuring that for any $u \in U$ there exists at least one $\ell \in L$ with $\ell \leq u$.

Remark 3.3 *Due to the compactness of $f(S^\theta)$ there always exist $z^\ell, z^u \in \mathbb{R}^r$ such that $\mathcal{N}^\theta \subseteq f(S^\theta) \subseteq \operatorname{int}(B)$ for $B := [z^\ell, z^u]$. In particular, this means that both, the notion of pseudo enclosures as well as of enclosures, make sense for the class of* (MOP) *we consider in this work.*

In order to determine the quality of a pseudo enclosure E, we use the same measure as for classical enclosures, namely its width $w(E)$. It is defined in [13] as the optimal value of

$$\max_{l,u}\ s(l,u) \quad \text{s.t.} \quad l \in L,\ u \in U,\ l \leq u, \tag{3.1}$$

where $s(l,u) := \min\{u_i - l_i \mid i \in [r]\}$ denotes the shortest edge length of the box $[l,u]$. It might seem surprising, especially with regard to single-objective global optimization, that the shortest and not the largest edge length is used in this definition. However, this definition ensures that all attainable points with respect to S contained in an (pseudo) enclosure of width at most $\varepsilon > 0$ are also ε-nondominated points of (MOP), see [13, Lemma 3.1]. For a more detailed discussion of the width as a quality measure, we refer to [13, 16]. A commonly used method to compute the lower and upper bound sets for an enclosure is to make use of so-called local upper bounds. These have been introduced in [25] and are used in various enclosure algorithms, including [13, 14, 15, 16, 28]. Beyond that, the concepts from [25] are used in other solution algorithms for multi-objective optimization problems as well, including [10, 32, 40]. These local upper bounds depend on so-called *stable* sets. Thereby, a set $Y \subseteq \mathbb{R}^r$ is called *stable* if no two elements in Y dominate each other, i.e., for any two distinct $y^1, y^2 \in Y$ there exist indices $i, j \in [r]$ such that $y^1_i < y^2_i$ and $y^2_j < y^1_j$. For the following, we fix $\theta > 0$ and a closed box $B \subseteq \mathbb{R}^r$ with $f(S^\theta) \subseteq \operatorname{int}(B)$ whose existence is established in Remark 3.3.

Definition 3.4 *Let $N \subseteq \operatorname{int}(B)$ be a finite and stable set. Then, the* lower search region *for N is $s(N) := \{y \in \operatorname{int}(B) \mid y' \nleq y \text{ for every } y' \in N\}$, and the* lower search zone *for some $u \in \mathbb{R}^r$ is given by $c(u) := \{y \in \operatorname{int}(B) \mid y < u\}$. A set $U = U(N)$ is called* local upper bound set *given N if*

$$s(N) = \bigcup_{u \in U(N)} c(u) \qquad \text{and} \qquad c(u^1) \nsubseteq c(u^2) \text{ for any } u^1, u^2 \in U(N) \text{ with } u^1 \neq u^2.$$

Each point $u \in U(N)$ is called a local upper bound *(LUB).*

Note that a local upper bound set $U(N)$ depends solely on a stable set N and a box B. This contrasts the definition of a θ-pseudo upper bound set U^θ (cf. Definition 3.2) which relies heavily on the nondominated sets $\mathcal{N}$ and $\mathcal{N}^\theta$. Therefore, we do not have to use the term *pseudo* when referring to a local upper bound set. For now, we denote the stable set upon which the set of local upper bounds depends by N^1. In the original case from [25], the set N^1 consists of attainable points, i.e., points $f(x)$ with $x \in S$. In our case, we cannot guarantee the attainability w.r.t. the set S of the points added to the set N^1 anymore, i.e., we might add a point $f(x)$ to the set N^1 with $x \in S^\theta \setminus S$. This is the main reason why we need the concept of pseudo enclosures and therefore also the slight change in the definition of local upper bounds. While in [25] only the concept of local upper bounds was presented, it can easily be transferred to lower bounds as well, see also [16, Definition 3.4]. Note that there is no need to modify anything in the definition of local lower bounds in the context of pseudo enclosures.

Definition 3.5 *Let $N \subseteq \operatorname{int}(B)$ be a finite and stable set. Then, the* upper search region *for N is $S(N) := \{y \in \operatorname{int}(B) \mid y' \not\geq y \text{ for every } y' \in N\}$ and the upper search zone for some $l \in \mathbb{R}^r$ is given by $C(l) := \{y \in \operatorname{int}(B) \mid l < y\}$. A set $L = L(N)$ is called a* local lower bound set *given N if*

$$S(N) = \bigcup_{l \in L(N)} C(l) \qquad \text{and} \qquad C(l^1) \nsubseteq C(l^2) \text{ for any } l^1, l^2 \in L(N) \text{ with } l^1 \neq l^2.$$

Each point $l \in L(N)$ is called a local lower bound *(LLB).*

For now, we denote the stable set upon which the set of local lower bounds depends by N^2. In the present paper, the set N^2 consists of points $y \in \mathbb{R}^r \setminus (f(S) + \operatorname{int}(\mathbb{R}^r_+))$ which are either nondominated points of possibly different relaxations or points whose attainability w.r.t. S can be ruled out. Details on that are provided in the upcoming description of the algorithm. Indeed, with the described choices of the sets N^1 and N^2 the local upper and local lower bound sets from Definitions 3.4 and 3.5 are actually pseudo upper and lower bound sets, respectively, for a pseudo enclosure of the nondominated set of (MOP). The respective proofs in [16, Lemma 3.3, Corollary 3.6] have to be adapted to the pseudo setting by replacing the original feasible set with the θ-feasible set whenever it appears in relation to the upper bounding part.

Lemma 3.6 *Let $\theta > 0$ and let $B \subseteq \mathbb{R}^r$ be a closed box with $f(S^\theta) \subseteq \operatorname{int}(B)$. Further, let $N^1 \subseteq f(S^\theta) + \mathbb{R}^r_+$ and $N^2 \subseteq \mathbb{R}^r \setminus (f(S) + \operatorname{int}(\mathbb{R}^r_+))$ be finite and stable. Then, for the finite local upper bound set $U(N^1)$ and the finite local lower bound set $L(N^2)$ it holds that $\mathcal{N}^\theta \subseteq U(N^1) - \mathbb{R}^r_+$ and $\mathcal{N} \subseteq L(N^2) + \mathbb{R}^r_+$. Hence, $E(L(N^2), U(N^1))$ is a pseudo enclosure of the nondominated set of* (MOP).

Having the width measure in mind, the main task of our algorithm is to update the two sets N^1 and N^2 from Lemma 3.6 such that they form smaller and smaller boxes (w.r.t. the respective shortest edges). The role of the set N^1 is realized by the set N consisting of attainable points w.r.t. the θ-feasible set S^θ, i.e., $N \subseteq f(S^\theta)$, and is driven towards the set $\mathcal{N}^\theta$ from above. The set N^2 is embodied by a set $\tilde{N}$ consisting of points that are nondominated by $\mathcal{N}$ which is why we call $\tilde{N}$ the set of utopian points. Naturally, the set of utopian points $\tilde{N}$ is driven to $\mathcal{N}$ from below. This suggests that one uses relaxations $\tilde{S}$ of S to obtain candidates for $\tilde{N}$. Before explaining the idea of our algorithm in detail we shortly clarify the term relaxation for the means of the present paper.

Definition 3.7 *Let* (MOP) *be given. Further, let $\tilde{S}$ be a set such that $S \subseteq \tilde{S}$. Then, we call*

$$\min f(x) \quad \text{s.t.} \quad x \in \tilde{S} \tag{RMOP$_{\tilde{S}}$}$$

a relaxed (MOP) *and $\tilde{S}$ a* relaxation *of S. We denote the nondominated set of* (RMOP$_{\tilde{S}}$) *by $\mathcal{N}^{\tilde{S}}$.*

The relaxations relevant for this work are based on piecewise linear underestimation functions for any nonlinear constraint function g_j. Details on this are given in Section 5. Note that the set S^θ is a relaxation of S according to Definition 3.7. Nevertheless, the set S^θ has in fact the same structure as S and is therefore in general not easier to treat algorithmically. Hence, S^θ does not serve as a relaxation according to the above-described idea. Instead, it is used for determining if solutions coming from relaxations in the classical sense are close enough to the original set S – and can therefore be considered as feasible for (MOP). For the rest of this work, we summarize any information regarding a relaxation $\tilde{S}$, i.e., any information regarding the description of the set $\tilde{S}$, in a structure $\mathcal{R}$. In our context, this means that for any nonlinear constraint function g_j, we have a substructure $\mathcal{R}.g_j$ where any information of the (piecewise) linear underestimator of g_j is stored. By abuse of notation, we also identify a relaxation by its information collection $\mathcal{R}$, denote the corresponding feasible set by $S^{\mathcal{R}}$ and the corresponding nondominated set by $\mathcal{N}^{\mathcal{R}}$.

General scheme

We now turn to the general framework used for this work. In fact, the framework is based on an algorithm that was presented in [28, Algorithm 5] in the context of computing an enclosure of the nondominated set of quadratically constrained problems. In this work, we use the general idea from that method but extend it to general nonconvex MINLPs and pseudo enclosures. In Algorithm 1, the general framework of this approach is provided. Starting the method requires a feasibility parameter $\theta > 0$

Algorithm 1 General scheme for computing a pseudo enclosure

```
Input: feasibility parameter $\theta > 0$, initial box $B = [z^\ell, z^u] \subseteq \mathbb{R}^r$ with $f(S^\theta) \subseteq \operatorname{int}(B)$,
    termination tolerance $\varepsilon_{\text{encl}} > 0$, offset factor $\delta \in (0, \varepsilon_{\text{encl}})$, initial relaxation $\mathcal{R}^1$
 1: Initialize set of attainable points $N = \emptyset$ and set of local upper bounds $U = \{z^u\}$
 2: Initialize set of utopian points $\tilde{N} = \emptyset$ and set of local lower bounds $L = \{z^\ell\}$
 3: Initialize the set $\mathcal{D}(U) = \{(u, \mathcal{R}^1) \mid u \in U\}$
 4: while $w(E(L, U)) > \varepsilon_{\text{encl}}$ do
 5:     $U_{\text{loop}} = U$
 6:     for $u \in U_{\text{loop}}$ do
 7:         if there exists $\ell \in L$ with $\ell \leq u$ and $s(\ell, u) \geq \varepsilon_{\text{encl}}$ then
 8:             $[\tilde{N}, N, L, U, \mathcal{D}(U)] = \texttt{improve\_region}(u - \delta e, \tilde{N}, N, L, U, \mathcal{D}(U))$
 9:         end if
10:     end for
11: end while
Output: Pseudo enclosure $E(L, U)$ satisfying $w(E) < \varepsilon_{\text{encl}}$ and set $N$ consisting
    of $\varepsilon_{\text{encl}}$-nondominated points and nondominated points given $\mathcal{N}$ coming from $\theta$-
    feasible points
```

and an initial box $B \subseteq \mathbb{R}^r$ with $f(S^\theta) \subseteq \operatorname{int}(B)$. According to Remark 3.3, such an initialization always exists. This initial enclosure gets iteratively refined during the procedure. Furthermore, a termination tolerance $\varepsilon_{\text{encl}} > 0$ as well as an offset factor $\delta \in (0, \varepsilon_{\text{encl}})$ is required, the role of which becomes clear later on. Lastly, one requires an initial relaxation $\mathcal{R}^1$.

We start the method by initializing the set of attainable points N as an empty set as we have no attainable point found yet. The same for the set of utopian points $\tilde{N}$

which is driven toward the nondominated set $\mathcal{N}$ from below. Moreover, we initialize the set of local upper bounds as $U = \{z^u\}$ and the set of local lower bounds as $L = \{z^\ell\}$. Lastly, and this is one important contribution of [28, Algorithm 5], we introduce the set $\mathcal{D}(U)$. It is initialized as $\mathcal{D}(U) = \{(u, \mathcal{R}^1) \mid u \in U\}$ and serves as an information mediator between the preimage space and the image space. Namely, it associates specific relaxations (preimage space information) with search zones (image space information). The importance of this exchange of information becomes clearer later on. Note that in the present work, this connection is deepened in comparison to [28, Algorithm 5].
After the initialization phase, the method enters the so-called *outer loop* intending to iteratively decrease the width of the pseudo enclosure below a desired tolerance $\varepsilon_{\text{encl}}$. In each iteration of the outer loop, the algorithm loops through every currently present local upper bound $u \in U_{\text{loop}}$ checking if it belongs to a box having a too-large shortest edge. If this is the case, the method interprets this local upper bound u as a search zone which has to be improved. We say that a search zone is improved if we either close the search zone or find an attainable point $f(x) \in f(S^\theta)$ lying in the search zone. If one of these is the case, we can replace (some of) the boxes belonging to u with smaller ones.

Improve within a search zone

We now turn to the procedure improve_region presented in Algorithm 2. Besides the

Algorithm 2 improve_region routine

```
Input: shifted local upper bound determining the search zone $u - \delta e$, set of utopian
    points $\tilde{N}$, set of attainable points $N$, set of local lower bounds $L$, set of local upper
    bounds $U$, relaxation collection $\mathcal{D}(U)$
 1: Set $\mathcal{R}^{\text{current}} = \mathcal{R}^u$, where $(u, \mathcal{R}^u) \in \mathcal{D}(U)$
 2: Set improved = false
 3: while improved = false do
 4:     if there exists $f(\tilde{x}) \in \mathcal{N}^{S^{\mathcal{R}_{\text{current}}}}$ with $f(\tilde{x}) \leq u - \delta e$ then
 5:         [improved, $\tilde{N}, N, L, U, \mathcal{D}(U)$] = find_points($\tilde{x}, u - \delta e, \tilde{N}, N, L, U, \mathcal{D}(U)$)
 6:         if improved = false then
 7:             $\mathcal{R}_{\text{current}}$ = refine_relaxation($u - \delta e, \mathcal{R}_{\text{current}}, \tilde{x}$)
 8:         end if
 9:     else
10:         Update $\tilde{N}$ and $L$ w.r.t. $u - \delta e$
11:         Set improved = true
12:     end if
13: end while
Output: updated sets $\tilde{N}, N, L, U, \mathcal{D}(U)$
```

current assignments of the sets $\tilde{N}, N, L, U$ and $\mathcal{D}(U)$, the method needs knowledge of the search zone of interest. This information is passed by the point $u - \delta e \in \mathbb{R}^r$, i.e., the current local upper bound of interest shifted by δe. On the one hand, the offset factor δ serves to overcome numerical issues arising when employing the (original) search region constraint $f(x) < u$. As this formulation is numerically more or less intractable, it is replaced by $f(x) \leq u - \delta e$. On the other hand, it is used to ensure large enough progress of the method, i.e., to satisfy the decrease condition presented in the forthcoming Proposition 4.4.

At the beginning of Algorithm 2 the relaxation $\mathcal{R}^u$ corresponding to the current search zone is chosen, i.e., $\mathcal{R}_{\text{current}} = \mathcal{R}^u$, and the `improved`-flag is set to `false`. Then, another `while`-loop is entered which we call the *inner loop* of the general framework. This loop is repeated until the `improved`-flag is set to `true`. As mentioned earlier, the method wants to either find a new attainable point $f(x) \in f(S^\theta)$ in the current search zone or declare it to be sufficiently explored.

The chosen relaxation has two different jobs. Firstly, it should suggest if there is a chance of finding an attainable point $f(x) \leq u - \delta e$ with $x \in S$. This first task is incorporated in the `if`-statement asking if there exists $f(\tilde{x}) \in \mathcal{N}^{S^{\mathcal{R}_{\text{current}}}}$ with $f(\tilde{x}) \leq u - \delta e$. One can check for that by using, e.g., the Pascoletti-Serafini scalarization [33] or the weighted-sum scalarization together with the search zone constraint $f(x) \leq u - \delta e$, as explained in [28, Section 3]. Using a scalarization, even the weighted-sum scalarization with arbitrary weights despite the overall problem being nonconvex, is satisfying here, as finding any arbitrary nondominated point is sufficient.

If this is not the case, i.e., if there exists no such $f(\tilde{x})$, then due to $S \subseteq S^{\mathcal{R}_{\text{current}}}$ there is no attainable point with regard to S in the current search region, i.e., in particular $u - \delta e \in \mathbb{R}^r \setminus (f(S) + \mathbb{R}^r_+)$. Then, the set of utopian points $\tilde{N}$ as well as the set of local lower bounds L is updated w.r.t. the point $u - \delta e$. This can be realized by using [28, Algorithm 4] for updating (and preserving the stability of) $\tilde{N}$ and [16, Algorithm 2] for L. By conducting these update procedures the search region corresponding to u is declared to be sufficiently explored and is therefore closed. This is because any box of the pseudo enclosure $E(L, U)$ with upper bound u after this update of the lower bound set L has a shortest edge length smaller or equal to δ and therefore particularly smaller than $\varepsilon_{\text{encl}}$.

On the contrary, if there exists $f(\tilde{x}) \in \mathcal{N}^{S^{\mathcal{R}_{\text{current}}}}$ with $f(\tilde{x}) \leq u - \delta e$ the relaxation $\mathcal{R}_{\text{current}}$ suggests that there may be an attainable point $f(x) \leq u - \delta e$ with $x \in S$. Recall that we focus on linear relaxations. This means that $\mathcal{R}_{\text{current}}$ represents a MOMILP and $\tilde{x}_I \in \mathbb{Z}^{|I|}$. Hence, we can interpret the existence of such $f(\tilde{x})$ as an indicator for the existence of an attainable point $f(x)$ with $x_I = \tilde{x}_I$ and $x \in S$. To verify that suggestion and eventually find such a point, the algorithm calls the `find_points`-routine. If the suggestion was wrong, i.e., no such point was found based on the information of the relaxation $\mathcal{R}_{\text{current}}$, the `find_points`-routine returns `improved` $=$ `false` and the algorithm decides for refinement of $\mathcal{R}_{\text{current}}$ which is conducted by the `refine_relaxation`-routine. Details regarding this routine are given in Sections 5 and 6. If the suggestion was right, the `find_points`-routine returns `improved` $=$ `true` and the `improve_region`-routine terminates.

Find attainable points

We now turn to the description of the `find_points`-routine provided in Algorithm 3. This routine intends to verify the relaxation's suggestion of the existence of an attainable point in the current search zone determined by $u - \delta e$. To do so, it uses information provided by the solution $\tilde{x}$ of the relaxed problem. Firstly, it checks if this solution can be considered feasible, i.e., if $\tilde{x}$ acceptably violates the nonlinear constraints and therefore $\tilde{x} \in S^\theta$. If that is the case, we consider $\tilde{x}$ as feasible for (MOP) and therefore want to add $f(\tilde{x})$ to the set of attainable points N.

Algorithm 3 `find_points` routine

```
Input: solution $\tilde{x} \in S^{\mathcal{R}_{\text{current}}}$ with $f(\tilde{x}) \leq u - \delta e$, shifted local upper bound determining the search region $u - \delta e$, set of utopian points $\tilde{N}$, set of attainable points $N$, set of local lower bounds $L$, set of local upper bounds $U$, relaxation collection $\mathcal{D}(U)$
 1: if $\tilde{x} \in S^{\theta}$ then
 2:     if $\tilde{N}$ is nondominated given $f(\tilde{x})$ then
 3:         Update $\tilde{N}$ and $L$ w.r.t. $f(\tilde{x})$
 4:     else
 5:         Set $\tilde{N} = \{y \in \tilde{N} \mid y \text{ is not dominated by } f(\tilde{x})\} \cup \{f(\tilde{x})\}$
 6:         Rebuild $L$ w.r.t. $\tilde{N}$
 7:     end if
 8:     Update $N$ and $U$ w.r.t. $f(\tilde{x})$
 9:     Set improved = true
10: else
11:     if $\tilde{N}$ is nondominated given $f(\tilde{x})$ then
12:         Update $\tilde{N}$ and $L$ w.r.t. $f(\tilde{x})$
13:         if exists $x$ feasible for (pMOP($\hat{x}_I$)) with $f(x) \leq u - \delta e$ and $\hat{x}_I = \tilde{x}_I$ then
14:             Update $N$ and $U$ w.r.t. $f(x)$
15:             Set $\mathcal{R}^u = \mathcal{R}_{\text{current}}$ for any new local upper bound $u$, i.e., update $\mathcal{D}(U)$
16:             Set improved = true
17:         else
18:             Set improved = false
19:         end if
20:     else
21:         Set improved = false
22:     end if
23: end if
Output: improved and (possibly) updated sets $\tilde{N}, N, L, U, \mathcal{D}(U)$
```

Before that, we have to ensure that there are no points in the set of utopian points $\tilde{N}$ which are dominated by $f(\tilde{x})$. Such a situation may arise if there is a point $f(\tilde{x}') \in \tilde{N}$ with $\tilde{x}' \notin S^{\theta}$ and $\tilde{x}' \notin S^{\mathcal{R}_{\text{current}}}$. In general, that can happen if there are pairwise non-including relaxations present, i.e., if for two appearing relaxations $\tilde{S}$ and $\tilde{S}'$ we neither have $\tilde{S} \subseteq \tilde{S}'$ nor $\tilde{S}' \subseteq \tilde{S}$. Given that one uses a nonmonotone refinement scheme, i.e., it is not guaranteed that $S^{\mathcal{R}_{\text{coarse}}} \supseteq S^{\mathcal{R}_{\text{refined}}}$, this can even occur in single-objective optimization if $\tilde{x} \in (S^{\mathcal{R}_{\text{refined}}} \setminus S^{\mathcal{R}_{\text{coarse}}}) \cap S^{\theta}$ and $\tilde{x}' \in S^{\mathcal{R}_{\text{coarse}}} \setminus (S^{\mathcal{R}_{\text{refined}}} \cup S^{\theta})$. On top of that, the transition to multiple objective functions creates another serious possibility for encountering such a scenario. This is because in the multi-objective setting, several utopian points may be active coming from several differently shaped relaxations. This opposes the single-objective case where the set of utopian points $\tilde{N}$ and the set of local lower bounds both collapse to the singleton of the currently best available lower bound. Therefore – even using a monotone, but nonuniform relaxation scheme –, the active utopian points may come from pairwise nonincluding relaxations resulting in the above-described situation.
However, if the set of utopian points $\tilde{N}$ is nondominated given $f(\tilde{x})$ we can simply update $\tilde{N}$ and L w.r.t. $f(\tilde{x})$ using [28, Algorithm 4] and [16, Algorithm 2], respectively. If otherwise there exists $\tilde{y} \in \tilde{N}$ dominated by $f(\tilde{x})$, we remove all of them from $\tilde{N}$ and

add $f(\tilde{x})$ resulting in an updated and stable set of utopian points $\tilde{N}$. We now have to rebuild the set of local lower bounds L corresponding to the updated set $\tilde{N}$ from scratch. This means, we start with $L = \{z^\ell\}$ and iteratively use [16, Algorithm 2] for each point in $\tilde{N}$. Note that if we dispense the above steps, i.e., not caring about the fact that the declared attainable point $f(\tilde{x})$ dominates some utopian points from $\tilde{N}$ and just update N and U w.r.t. $f(\tilde{x})$, this would yield the situation, where for an incoming local upper bound u there is no local lower bound $\ell \in L$ with $\ell \leq u$. After we ensured that $f(\tilde{x}) \in \tilde{N}$ and $\tilde{N}$ is stable, we are ready to update the set of attainable points N and the corresponding set of local upper bounds U w.r.t. $f(\tilde{x})$ using [28, Algorithm 3] and [25, Algorithm 3], respectively. Afterward, since we improved the search zone by finding an attainable point, the `improved`-flag is set to `true`.
If otherwise, the solution of the relaxed problem $\tilde{x}$ violates the constraints by too much, i.e., if $\tilde{x} \notin S^\theta$, the second task of the relaxation comes into play. This is to also improve the lower bounding procedure of the method, i.e., to move the set of utopian points $\tilde{N}$ and therefore also the set of local lower bounds L in the direction of the nondominated set $\mathcal{N}$ by updating them w.r.t. $f(\tilde{x})$. In fact, this is successful if $\tilde{N}$ is nondominated given $f(\tilde{x})$ which is checked in line 11 of Algorithm 3. If that is not the case, we do not expect the relaxation to provide any helpful information, and in particular, its suggestion of finding an attainable point to be not very trustworthy. Consequently, the `find_points`-routine returns `improved` = `false` and a refinement step is conducted. If otherwise, $f(\tilde{x})$ improves the set $\tilde{N}$ towards $\mathcal{N}$, both, $\tilde{N}$ and L, are updated w.r.t. $f(\tilde{x})$ using [28, Algorithm 4] and [16, Algorithm 2], respectively.
After checking a possible update of the set of utopian points, the relaxation's suggestion is verified in line 13. The verification is executed by checking if there exists $x \in S$ with $x_I = \tilde{x}_I$ and $f(x) \leq u - \delta e$ by finding a feasible point of the continuous but non-convex multi-objective optimization problem (pMOP($\hat{x}_I$)) with additional search zone constraint. This can be done again by using, e.g., the Pascoletti-Serafini scalarization [33] or the weighted-sum scalarization together with the search zone constraint $f(x) \leq u - \delta e$ as in [28]. Note that, e.g., if one goes for the weighted sum approach, it suffices to solve the resulting NLP only to feasibility. Of course, solving to local or even global optimality might speed up the procedure in terms of iterations but it is not necessary since all we want to achieve is encoded in the search zone constraint $f(x) \leq u - \delta e$. If the corresponding (pMOP($\hat{x}_I$)) together with the search zone constraint is feasible, we found a new attainable point $f(x)$ which is used to update the sets N and U using [28, Algorithm 2] and [25, Algorithm 2].
In particular, this means that the relaxation $\mathcal{R}_{\text{current}}$ mimics the behavior of (MOP) in the search region good enough. This is why the relaxation $\mathcal{R}_{\text{current}}$ is assigned to the incoming local upper bounds after the update procedures of U and N w.r.t. $f(x)$. In this step, the information mediation between the image space information of search regions and the preimage space information of relaxations takes place. We set the `improved`-flag to `true`, terminate the `improve_region`-routine, and move on to the next search zone within the outer loop of the procedure.
If otherwise, (pMOP($\hat{x}_I$)) together with the search zone constraint is not feasible, the relaxation's suggestion was wrong, we set the `improved`-flag to `false` and go for a refinement step within the `improve_region`-routine.

4 Correct and finite termination

This section is dedicated to proving correctness and finiteness of Algorithm 1. We start with correctness, meaning we show that $E(L,U)$ forms a pseudo enclosure of (MOP) according to Definition 3.2 throughout Algorithm 1.

Proposition 4.1 *Let $\theta > 0$, $B = [z^\ell, z^u]$, $\varepsilon_{\mathrm{encl}} > \delta > 0$, and $\mathcal{R}^1$ be the input parameters of Algorithm 1. Then, the set $E(L,U)$ is a pseudo enclosure of (MOP) throughout the whole procedure.*

Proof. Due to Lemma 3.6 it suffices to show that $N \subseteq f(S^\theta) + \mathbb{R}^r_+$ and $\tilde{N} \subseteq \mathbb{R}^r \setminus (f(S) + \operatorname{int}(\mathbb{R}^r_+))$ throughout the algorithm. The set of attainable points N only gets updated in lines 8 and 14 of Algorithm 3. In both cases, N is updated w.r.t. a point $f(x) \in f(S^\theta)$ – note that $x \in S \subseteq S^\theta$ for the update in line 14 – and therefore particularly $N \subseteq f(S^\theta) + \mathbb{R}^r_+$. Stability of N throughout these updates follows by the correctness of [28, Algorithm 3] shown in [28, Lemma 4.6]. On the other hand, the set of utopian points $\tilde{N}$ gets only updated in line 10 of Algorithm 2 and in lines 3, 5 and 12 of Algorithm 3. Except the update in line 5 of Algorithm 3, all these updates are conducted using [28, Algorithm 4] which is correct by [28, Lemma 4.7], i.e., in particular stability of $\tilde{N}$ is ensured. In line 10 of Algorithm 2, the set $\tilde{N}$ is updated w.r.t. $u - \delta e$ and the guarantee (s. line 4 of Algorithm 2) that there exists no $\tilde{x} \in \tilde{S}$ with $f(\tilde{x}) \leq u - \delta e$ for some relaxation $\tilde{S} \supseteq S$. Consequently, we have that $u - \delta e \in \mathbb{R}^r \setminus (f(S) + \operatorname{int}(\mathbb{R}^r_+))$, as required. For all update steps in Algorithm 3, the set $\tilde{N}$ is updated w.r.t. a point $f(\tilde{x})$, where $\tilde{x} \in \tilde{S}$ and $f(\tilde{x}) \in \mathcal{N}^{\tilde{S}}$ for some relaxation $\tilde{S} \supseteq S$. By the relaxation property of $\tilde{S}$ w.r.t. S and the fact that $f(\tilde{x})$ is a nondominated point of a relaxed (MOP), there exists no $y \in \mathcal{N}$ dominating $f(\tilde{x})$, i.e., $f(\tilde{x}) \in \mathbb{R}^r \setminus (f(S) + \operatorname{int}(\mathbb{R}^r_+))$, as required. For the update in line 5 of Algorithm 3, all points in $\tilde{N}$ which are dominated by $f(\tilde{x})$ are removed to preserve stability of $\tilde{N}$ – which is also done in [28, Algorithm 4] but written explicitly here for clarity. □

Finiteness heavily depends on the specific technique in the `refine_relaxation`-routine. As there are plenty of different possibilities for that, and we want to highlight this aspect of our framework, we formulate an assumption on the refinement technique which is sufficient for finite termination of Algorithm 1.

Assumption 4.2 *Let $\{(\mathcal{R}_k, \tilde{x}^k)\}_{k\in\mathbb{N}}$ be a chain of relaxations and relaxation-feasible points produced by the chosen refinement scheme, i.e., the `refine_relaxation`-routine. That means, starting with an initial relaxation $\mathcal{R}^1 = \mathcal{R}_1$ and a given relaxation-feasible point $\tilde{x}^1 \in S^{\mathcal{R}_1}$, the refinement scheme produces a relaxation $\mathcal{R}_2$. Then, using another (arbitrarily chosen) relaxation-feasible point $\tilde{x}^2 \in S^{\mathcal{R}_2}$ the refinement scheme produces a relaxation $\mathcal{R}_3$ and so on. Then, for any $\theta > 0$, we assume that there exists $K \in \mathbb{N}$ such that $\tilde{x}^K \in S^\theta$ and is therefore considered feasible for (MOP).*

Assumption 4.2 allows us to prove that the `improve_region`-routine terminates after finitely many steps.

Proposition 4.3 *Let $u - \delta e$, $\tilde{N}$, N, L, U and $\mathcal{D}(u)$ be the input of Algorithm 2. Further, let Assumption 4.2 be satisfied for the chosen relaxation-refinement scheme. Then, Algorithm 2 terminates after finitely many steps.*

Proof. We have to show that after finitely many iterations of the `while`-loop in Algorithm 2, we have that `improved` = `true` holds. Assume for a contradiction that this is not the case. Then, in each iteration we enter the `if`-statements in lines 4 and 6 since otherwise `improved` = `true` in line 11. For any $k \in \mathbb{N}$, we denote by $\mathcal{R}_k$ and $\tilde{x}^k \in S^{\mathcal{R}_k}$ the relaxation and relaxation-feasible point, respectively, at the beginning of the k-th iteration. For any $k \in \mathbb{N}$, we have that $\mathcal{R}_{k+1}$ is obtained by the `refine_relaxation`-routine. Consequently, we have an infinite chain of relaxations $\mathcal{R}^1 = \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \ldots$ produced by the `refine_relaxation`-routine with corresponding relaxation-feasible points $\tilde{x}^k \in S^{\mathcal{R}_k}$ for all $k \in \mathbb{N}$. By Assumption 4.2, we know that there exists $K \in \mathbb{N}$ such that $\tilde{x}^K \in S^\theta$, i.e., is considered feasible for (MOP). Hence, in the K-th iteration we enter the `if`-statement in line 1 of the `find_points`-routine in Algorithm 3 and set `improved` = `true` in line 9, a contradiction. Thus, Algorithm 2 terminates after finitely many steps. □

Proposition 4.3 is one of two key ingredients for proving finiteness of Algorithm 1. The other one is a guaranteed decrease regarding the edge lengths of the boxes belonging to the enclosure before and after one call of the outer loop of Algorithm 1, see also [28, Theorem 5.9].

Proposition 4.4 *Let L^{start} and U^{start} be the set of local lower and local upper bounds, respectively, at the beginning of some iteration of the outer loop during Algorithm 1. Analogously, denote by L^{end} and U^{end} the bound sets at the end of this iteration. Further, let Assumption 4.2 hold for the chosen relaxation refinement scheme.*
Then, for any $\ell^e \in L^{\text{end}}$ and any $u^e \in U^{\text{end}}$ with $\ell^e \leq u^e$ there exist $\ell^s \in L^{\text{start}}$ and $u^s \in U^{\text{start}}$ such that the following hold:

(i) $\ell^s \leq \ell^e \leq u^e \leq u^s$, *i.e., the width does not increase during one iteration.*

(ii) There exists an index $i \in [r]$ such that $(u^e - \ell^e)_i < \max\{(u^s - \ell^s)_i - \delta, \varepsilon_{\text{encl}}\}$, i.e., for any pair (ℓ^e, u^e) we obtain a decrease w.r.t. the edge length for at least one edge $i \in [r]$.

This finally allows us to state and prove finite convergence and correctness of Algorithm 1 by applying the convergence result from [28].

Theorem 4.5 *Let $\theta > 0$, $\varepsilon_{\text{encl}} > \delta > 0$, $\mathcal{R}^1$ and $B = [z^\ell, z^u] \subseteq \mathbb{R}^r$ be the input parameters of Algorithm 1 and let Assumption 4.2 hold. We define*

$$\Delta := \max\{\|z^u - z^\ell\|_\infty, \varepsilon_{\text{encl}}\} \quad \text{and} \quad \kappa := r \left\lceil \frac{\Delta - \varepsilon_{\text{encl}}}{\delta} \right\rceil + 1 < \infty.$$

Then the number of iterations of the outer loop of Algorithm 1 is bounded by κ, i.e., Algorithm 1 terminates after finitely many steps. Furthermore, a pseudo enclosure E of the nondominated set $\mathcal{N}$ satisfying $w(E) < \varepsilon_{\text{encl}}$ is returned.

Proof. The finiteness part is analogous to [28, Theorem 5.10, Corollary 5.11]. The correctness part follows by Proposition 4.1. □

5 Box-based piecewise linear relaxations

In this section, we present one key contribution of this work. Namely, a novel method for relaxing general nonconvex terms by using only piecewise linear functions which is closely related to methods proposed in [8, 21]. While we tackle, in view of the feasible set, the same problem class as [8], we differ in not using a simplex partitioning scheme (triangulation) of the variable domain. Instead, we use a box-based one similar to [21] where – in contrast to our case – the nonlinear constraints are not assumed to be given explicitly but the respective global Lipschitz constants. We leverage the box-based scheme because it facilitates the use of bound tightening as a refinement step in our algorithm. One disadvantage of using a simplex partitioning scheme is that a triangulation of the variable domain cannot easily be transferred to a new one with tightened bounds. This is because in general, either new simplices are necessary for getting a triangulation of the tightened domain or a completely new triangulation has to be computed. An exemplary situation is depicted in Figure 1 for a two-dimensional box domain – the simplest case that may occur where boxes and simplices do not coincide. One can see that one either introduces new simplices to keep some structure of the former triangulation, or one has to use a completely new triangulation.

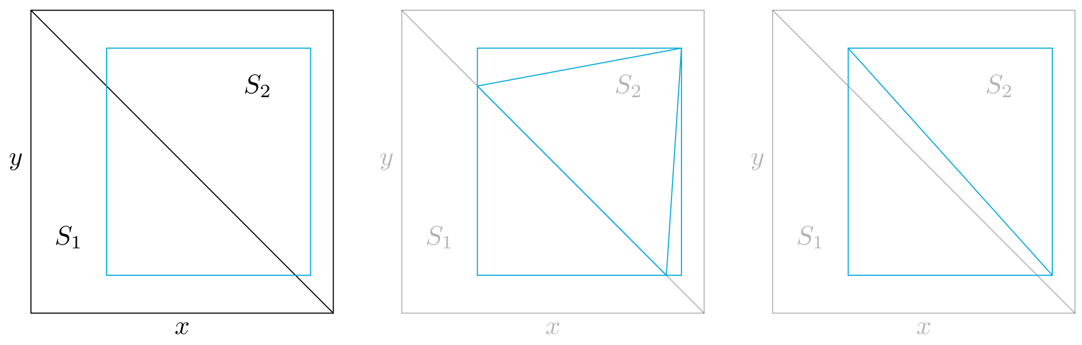

Figure 1: Left: initial triangulation of two-dimensional box domain and tightened box domain (blue). Mid: possible triangulation of tightened box domain based on information from former triangulation. Right: new triangulation of tightened box domain.

Furthermore, both options have serious drawbacks: with the first option, one increases the number of binary variables – which contradicts our idea of applying bound tightening. The second one requires building new partitions (or triangulations) of a domain which is not an easy task at all, especially in higher dimensions. To bypass these difficulties, we stick to a box-based relaxation scheme of the nonlinear constraints. Additionally, the idea of simplex-based partitions comes originally from the idea of approximating a nonlinear function on a given domain [2, 27, 30]. For doing that, simplices are very well suited due to their ability to interpolate any nonlinear function by piecewise-linear functions with interpolation points being their vertices. This interpolation property gets lost when turning to box-based partitions. But as we are aiming at relaxations instead of approximations, i.e., we are inexact in most of the cases anyway, the inexactness at the vertices is not a problem.
Subsequently, we describe the relaxation scheme for a nonlinear function $g_j, j \in [p]$. Proceeding like that for any $j \in [p]$, one obtains an MILP relaxation of (MOP).

Therefore, let $j \in [p]$. We denote by $X_{\mathcal{J}_j} \subseteq \mathbb{R}^{l_j}$ the projection of the box X to the l_j components appearing in g_j determined by the index set $\mathcal{J}_j \subseteq [n]$.
Finally, we describe how a relaxation of the constraint $g_j(x_{\mathcal{J}_j}) \leq 0$ is obtained by getting an underestimator of g_j on the box $X_{\mathcal{J}_j}$. Later on, we apply the same procedure on subboxes $\mathcal{B} \subseteq X_{\mathcal{J}_j}$ to get an underestimator of g_j also on subboxes $\mathcal{B}$ of $X_{\mathcal{J}_j}$. Note that the key ingredient for refining the resulting relaxations is an ongoing partitioning of the variable domain box $X_{\mathcal{J}_j}$ into smaller and smaller subboxes.
We use the vertices/corner points $v^1, \dots, v^{2^{l_j}} \in \mathbb{R}^{l_j}$ of $X_{\mathcal{J}_j}$ to set up a linear least squares problem

$$\min_{w \in \mathbb{R}^{l_j+1}} \|Aw - b\|_2, \tag{LSq}$$

where the 2^{l_j} rows of A are formed by the vectors $(v^s, 1)$, $s = 1, \dots, 2^{l_j}$, and the vector b collects the corresponding values $g_j(v^s)$, $s = 1, \dots, 2^{l_j}$. Given a solution $\bar{w} \in \mathbb{R}^{l_j+1}$ of (LSq), we define a linear function $\mathcal{L}^{\bar{w}}_{g_j}$ on the box $X_{\mathcal{J}_j}$ with a slope determined by the values of g_j at the corners of the box. This linear function is defined as

$$\mathcal{L}^{\bar{w}}_{g_j} : \mathbb{R}^{l_j} \to \mathbb{R}, \quad x \mapsto \bar{w}^\top \begin{pmatrix} x \\ 1 \end{pmatrix}. \tag{LSq-F}$$

To simplify the notation, we write $\mathcal{L}_{g_j}$ instead of $\mathcal{L}^{\bar{w}}_{g_j}$ for the remainder of this work. The linear function $\mathcal{L}_{g_j}$ is in general neither an underestimator nor an overestimator of g_j on $X_{\mathcal{J}_j}$ and also not an interpolation function. By shifting it using so-called *under-* and *overestimation errors* as done in [7, 8, 19], one obtains an over- and underestimator, respectively. As we only consider nonlinear inequality constraints of the form $g_j(x_{\mathcal{J}_j}) \leq 0$, we are only interested in finding underestimators and therefore in computing the overestimation error. In fact, one computes the overestimation error θ^o by solving the optimization problem

$$\theta^o := \max_{x \in X_{\mathcal{J}_j}} \left(\mathcal{L}_{g_j}(x) - g_j(x) \right). \tag{OEP}$$

Shifting $\mathcal{L}_{g_j}$ by $-\theta^o$, i.e. $\mathcal{L}^u_{g_j}(x) := \mathcal{L}_{g_j}(x) - \theta^o$, yields that $\mathcal{L}^u_{g_j}$ is an underestimator of g_j on $X_{\mathcal{J}_j}$ Note that (OEP) is in general a nonconvex continuous optimization problem. In [19], the authors show that θ^o can be computed by solving l_j convex optimization problems. However, these results rely on a simplex-based partition of the domain or, more precisely, on the interpolation property of the linear approximations on the vertices of the simplex. Hence, these results are not straightforwardly applicable to our box-based partitioning.
If there is an adequate global solver for (continuous) nonconvex problems available, one can directly solve the problem (OEP) (as it is done in the implementations for the present paper). On the other hand, if we do not have an adequate global solver, one could instead use convex relaxation techniques, e.g., the αBB method [1], in order to compute an upper bound on θ^o. Naturally, one has to ensure that the constraint functions g_j satisfy the requirements of the chosen solution technique. For instance, to apply the αBB method the functions g_j have to be twice continuously differentiable.
Now, by replacing any appearance of $g_j(x_{\mathcal{J}_j})$ in (MOP) with the term $\mathcal{L}^u_{g_j}(x_{\mathcal{J}_j})$, we obtain the desired linear relaxation of the nonlinear constraint function g_j.

For the upcoming theory, we need the concept of the aforementioned underestimation error θ^u. It is given by

$$\theta^u := \max_{x \in X_{\mathcal{J}_j}} g_j(x) - \mathcal{L}_{g_j}(x). \tag{UEP}$$

By shifting $\mathcal{L}_{g_j}$ by θ^u, i.e., $\mathcal{L}^o_{g_j}(x) := \mathcal{L}_{g_j}(x) + \theta^u$, one obtains an overestimator $\mathcal{L}^o_{g_j}(x)$ of $g_j(x)$ on $X_{\mathcal{J}_j}$.
Summarizing the above, we relax any nonlinear constraint function $g_j, j \in [p]$, using the described procedure. As explained before, for any nonlinear constraint g_j we collect the relaxation information in the structure $\mathcal{R}.g_j$. Note that proceeding like that yields a relaxation $\mathcal{R}$ of S in the sense of Definition 3.7. As already mentioned, the above-described relaxation technique is similar to the ones in [8, 21] but differs in essential aspects. Therefore, the results from [8, 21] cannot be applied which is why we provide the subsequent proofs. To the best of our knowledge, these results cannot be found in the literature.
We start with considering these relaxations within the context of the θ-feasible set S^θ.

Definition 5.1 *Let $\mathcal{R}$ be a relaxation of S and let $\tilde{x} \in S^{\mathcal{R}}$. For any $g_j, j \in [p]$, we define the* constraint satisfaction error $\theta^{\mathcal{R}}_{g_j}(\tilde{x})$ *of the relaxation-feasible point $\tilde{x}$ w.r.t. the constraint function g_j and the relaxation $\mathcal{R}$ as*

$$\theta^{\mathcal{R}}_{g_j}(\tilde{x}) := \max \left\{ g_j(\tilde{x}_{\mathcal{J}_j}), 0 \right\}.$$

Similarly, we define the constraint satisfaction error $\theta^{\mathcal{R}}(\tilde{x})$ *of $\tilde{x}$ and the relaxation $\mathcal{R}$ as*

$$\theta^{\mathcal{R}}(\tilde{x}) := \max \left\{ \theta^{\mathcal{R}}_{g_j}(\tilde{x}) \mid j \in [p] \right\}.$$

Note that for any $j \in [p]$, we have that $\theta^{\mathcal{R}}_{g_j}(\tilde{x}) \leq \max\{g_j(\tilde{x}_{\mathcal{J}_j}) - \mathcal{L}^u_{g_j}(\tilde{x}_{\mathcal{J}_j}), 0\}$ for any relaxation-feasible point $\tilde{x}$. Using the constraint satisfaction error of a given $\tilde{x} \in S^{\mathcal{R}}$ one can determine if $\tilde{x} \in S^\theta$ and therefore can be considered as feasible for (MOP).

Proposition 5.2 *Let $\theta > 0$, $\mathcal{R}$ be a relaxation of S, and let $\tilde{x} \in S^{\mathcal{R}}$. If $\theta^{\mathcal{R}}(\tilde{x}) \leq \theta$, then $\tilde{x} \in S^\theta$.*

To show that the constraint satisfaction error decreases with ongoing refinement of the corresponding relaxations, we give an overestimate of the constraint satisfaction error.

Lemma 5.3 *Let $\mathcal{R}$ be a relaxation of S and let $\tilde{x} \in S^{\mathcal{R}}$. Further, let $j \in [p]$ and $\mathcal{B} \subseteq X_{\mathcal{J}_j}$ be a box with $\tilde{x}_{\mathcal{J}_j} \in \mathcal{B}$. We denote by $\mathcal{L}^{o,\mathcal{B}}_{g_j}$ and $\mathcal{L}^{u,\mathcal{B}}_{g_j}$ the linear overestimation and underestimation function, respectively, and by $\theta^{o,\mathcal{B}}_{g_j}$ and $\theta^{u,\mathcal{B}}_{g_j}$ the solutions of* (OEP) *and* (UEP)*, respectively, w.r.t. the box $\mathcal{B}$. Then, we have that*

$$\theta^{\mathcal{R}}_{g_j}(\tilde{x}) \leq \theta^{u,\mathcal{B}}_{g_j} + \theta^{o,\mathcal{B}}_{g_j}.$$

Proof. We first note that $\mathcal{L}^{u,\mathcal{B}}_{g_j}(x) \leq g_j(x) \leq \mathcal{L}^{o,\mathcal{B}}_{g_j}(x)$ for all $x \in \mathcal{B}$ by definition. W.l.o.g. assume that $g_j(\tilde{x}_{\mathcal{J}_j}) \geq 0$. We calculate

$$\theta^{\mathcal{R}}_{g_j}(\tilde{x}) \leq g_j(\tilde{x}_{\mathcal{J}_j}) - \mathcal{L}^{u,\mathcal{B}}_{g_j}(\tilde{x}_{\mathcal{J}_j}) \leq \mathcal{L}^{o,\mathcal{B}}_{g_j}(\tilde{x}_{\mathcal{J}_j}) - \mathcal{L}^{u,\mathcal{B}}_{g_j}(\tilde{x}_{\mathcal{J}_j}) \leq \theta^{u,\mathcal{B}}_{g_j} + \theta^{o,\mathcal{B}}_{g_j},$$

which shows the result. □

The goal for the rest of this section is to show that we can control the errors $\theta^u_{g_j}$ and $\theta^o_{g_j}$ in the sense that we can bring them to zero by successively partitioning the original box $X_{\mathcal{J}_j}$ into smaller and smaller boxes. To that end, we are referring only to such partitioning schemes when talking about refinement schemes for the remainder of this work.
Let $\mathcal{B} \subseteq \mathbb{R}^n$ be a box. Recall that a collection of boxes $\mathcal{B}_1, \dots, \mathcal{B}_s$, $s \in \mathbb{N}$, is called a *box-partition* of $\mathcal{B}$, if $\mathcal{B} = \bigcup_{i\in[s]} \mathcal{B}_i$ and if for any $i, i' \in [s]$ with $i \neq i'$, we have that $\operatorname{int}(\mathcal{B}_i) \cap \operatorname{int}(\mathcal{B}_{i'}) = \emptyset$. Now, let $j \in [p]$ and let $\mathcal{B}_1, \dots, \mathcal{B}_s$ be a partition of the box $X_{\mathcal{J}_j} = [x^\ell_{\mathcal{J}_j}, x^u_{\mathcal{J}_j}]$. Then, the boxes are of the form

$$\mathcal{B}_i := [x^{\ell,i}_{\mathcal{J}_j}, x^{u,i}_{\mathcal{J}_j}] \subseteq [x^\ell_{\mathcal{J}_j}, x^u_{\mathcal{J}_j}],$$

where $i \in [s]$. Now, for each of these boxes $\mathcal{B}_i$ we proceed as before and get linear underestimation functions $\mathcal{L}^{u,i}_{g_j}$. Furthermore, we introduce indicator variables $b_i \in \{0, 1\}$ which switch on and off the corresponding linear underestimator constraints we have seen before. This is realized via the formulation

$$\sum_{i\in[s]} b_i \cdot \mathcal{L}^{u,i}_{g_j}(x_{\mathcal{J}_j}) \le 0.$$

In addition to that, to determine which is the current active box (and thus, which underestimator has to be used), we add the constraints

$$b_i(x^{\ell,i}_l - x_l) \le 0 \quad \text{and} \quad b_i(x_l - x^{u,i}_l) \le 0,$$

for all $i \in [s]$ and $l \in \mathcal{J}_j$. Finally, to ensure that there is exactly one active box, we add the constraint $\sum_{i\in[s]} b_i = 1$ (which can be implemented using the help of Special-Ordered-Set 1 constraints [2, 39]). The goal of successively partitioning the variable domain is to refine the relaxation, i.e., starting with an initial relaxation $\mathcal{R}^1 = \mathcal{R}_1$ we want to end up with a relaxation $\mathcal{R}_2$ that is finer than $\mathcal{R}_1$ in some sense. Note that the relaxation-refinement procedures (see Section 6) do not guarantee that $S^{\mathcal{R}_2} \subseteq S^{\mathcal{R}_1}$ – consequently, in the present work we are not talking about inclusion of the respective feasible sets concerning the term *refinement* of relaxations. Instead, refining a relaxation means refining the corresponding partitions of the variable domains as defined in the following.

Definition 5.4 *Let $(\mathcal{B}_i)_{i\in[s]}$ be a partition of a box $\mathcal{B}$. We call another partition $(\tilde{\mathcal{B}}_i)_{i\in[\tilde{s}]}$ of $\mathcal{B}$* finer than *(or a* refinement of*) $(\mathcal{B}_i)_{i\in[s]}$ if for any $i \in [s]$ there exists $i' \in [\tilde{s}]$ with $\tilde{\mathcal{B}}_{i'} \subseteq \mathcal{B}_i$. We call $(\tilde{\mathcal{B}}_i)_{i\in[\tilde{s}]}$* strictly finer *if it is finer than $(\mathcal{B}_i)_{i\in[s]}$ and there exists $i' \in [\tilde{s}]$ with $\tilde{\mathcal{B}}_{i'} \subsetneq \mathcal{B}_i$ for some $i \in [s]$.*

Furthermore, we want the relaxation-refinement technique to satisfy Assumption 4.2 and therefore yield finite termination of Algorithm 1. We give a reformulation of Assumption 4.2 to our setting based on Proposition 5.2.

Assumption 5.5 (Reformulation of Assumption 4.2) *Let $\mathcal{R}^1 = \mathcal{R}_1$ be the initial relaxation defined on the original variable domain boxes $X_{\mathcal{J}_j}$ for all $j \in [p]$. Further, let $\{(\mathcal{R}_k, \tilde{x}^k)\}_{k\in\mathbb{N}}$ be a chain of relaxations and relaxation-feasible points produced by the chosen `refine_relaxation`-routine like in Assumption 4.2. Then, for any $\theta > 0$, we assume that there exists $K \in \mathbb{N}$ such that after K refinement steps it holds that*

$$\theta^{\mathcal{R}_K}(\tilde{x}^K) \le \theta.$$

To prove that Assumption 5.5 holds for a given refinement scheme, we use a slightly different notion of δ-preciseness as introduced in [8]. But before introducing it, we need the notion of the *longest edge* $\lambda(\mathcal{B})$ of a box $\mathcal{B} = [x^\ell, x^u] \subseteq \mathbb{R}^n$. It is defined as $\lambda(\mathcal{B}) := \max_{l\in[n]}\{x^u_l - x^\ell_l\}$. In fact, δ-preciseness for a refinement scheme means that we obtain arbitrarily small boxes with ongoing refinement.

Definition 5.6 *Let $(\mathcal{R}_k)_{k\in\mathbb{N}}$ be a chain of relaxations produced by a refinement procedure and let $(\tilde{x}^k)_{k\in\mathbb{N}}$ be an arbitrary sequence of points with $\tilde{x}^k \in S^{\mathcal{R}_k}$. The relaxation refinement procedure is called* δ-precise *if for any $\delta > 0$ and for any $j \in [p]$ there exists $K \in \mathbb{N}$ such that*

$$\lambda\left(\mathcal{B}^k_{\mathcal{J}_j}\right) < \delta$$

for any $k \geq K$, where $(\mathcal{B}^k_{\mathcal{J}_j})_{k\in\mathbb{N}}$ denotes the sequence of active boxes within the respective partition of $X_{\mathcal{J}_j}$, i.e., $\tilde{x}^k_{\mathcal{J}_j} \in \mathcal{B}^k_{\mathcal{J}_j}$ for any $k \in \mathbb{N}$.

For the remainder of this section we keep the notations of Definition 5.6, i.e., $(\mathcal{R}_k)_{k\in\mathbb{N}}$ denotes a chain of relaxations produced by a δ-precise refinement procedure, $(\tilde{x}^k)_{k\in\mathbb{N}}$ a sequence of points with $\tilde{x}^k \in S^{\mathcal{R}_k}$, and, for any $j \in [p]$, the sequence $(\mathcal{B}^k_{\mathcal{J}_j})_{k\in\mathbb{N}}$ denotes the sequence of active boxes within the respective partition of $X_{\mathcal{J}_j}$, i.e., $\tilde{x}^k_{\mathcal{J}_j} \in \mathcal{B}^k_{\mathcal{J}_j}$ for any $k \in \mathbb{N}$. Subsequently, we prove that given a δ-precise refinement scheme, Assumption 5.5 holds. We start with the notion of the maximal elongation of a continuous function $f\colon \mathbb{R}^n \to \mathbb{R}$ over a compact domain $A \subseteq \mathbb{R}^n$. We denote it by $\mathrm{d}(f, A)$ and define it by

$$\mathrm{d}(f, A) := \max_{x\in A} f(x) - \min_{x\in A} f(x).$$

The first observation given a δ-precise refinement procedure is that for every $j \in [p]$ it holds that $\mathrm{d}(g_j, \mathcal{B}^k_{g_j}) \to 0$ for $k \to \infty$.

Lemma 5.7 *Let $j \in [p]$ and $\varepsilon > 0$. Then, there exists $K \in \mathbb{N}$ such that for any $k \geq K$ we have*

$$\mathrm{d}(g_j, \mathcal{B}^k_{\mathcal{J}_j}) < \varepsilon.$$

In particular, for $\mathcal{V}^k := \{v \in \mathcal{B}^k_{\mathcal{J}_j} \mid v$ is corner point of $\mathcal{B}^k_{\mathcal{J}_j}\}$, we have that $\mathrm{d}(g_j, \mathcal{V}^k) < \varepsilon$.

Proof. By the Extreme Value Theorem, we have that g_j attains its minimum and maximum value on all boxes $\mathcal{B}^k_{g_j}$, i.e., the function d is well-defined on each of these boxes. Now, for given $\varepsilon > 0$ there exists $\delta > 0$ such that for $x, x' \in X_{\mathcal{J}_j}$ with $\|x - x'\| < \delta$ we have that $|g_j(x) - g_j(x')| < \varepsilon$. In particular, by δ-preciseness of the refinement scheme, there exist $K \in \mathbb{N}$ such that $\|x - x'\| < \delta$ for any $x, x' \in \mathcal{B}^k_{\mathcal{J}_j}$ for any $k \geq K$. This shows the first statement and the second one follows immediately by the fact that $\mathcal{V}^k \subseteq \mathcal{B}^k_{\mathcal{J}_j}$. □

The analog of Lemma 5.7 also holds for the linear functions defined in (LSq-F).

Lemma 5.8 *Let $j \in [p]$ and $\varepsilon > 0$. For any $k \in \mathbb{N}$, let $\mathcal{L}^k\colon \mathcal{B}^k_{\mathcal{J}_j} \to \mathbb{R}$ be a linear function as defined in (LSq-F) w.r.t. $\mathcal{B}^k_{\mathcal{J}_j}$ and g_j. Then, there exists $K \in \mathbb{N}$ such that*

$$\mathrm{d}(\mathcal{L}^k, \mathcal{V}^k) < \varepsilon$$

for any $k \geq K$ and $\mathcal{V}^k$ as defined in Lemma 5.7. In particular, this means also that $\mathrm{d}(\mathcal{L}^k, \mathcal{B}^k_{\mathcal{J}_j}) < \varepsilon$.

Proof. We prove the first claim by showing that $\mathrm{d}(\mathcal{L}^k, \mathcal{V}^k)$ can be bounded depending on $\mathrm{d}(g_j, \mathcal{V}^k)$ and subsequently applying Lemma 5.7. To do so, we show that the maximal distance between $\mathcal{L}^k$ and g_j attainable on the corner points $\mathcal{V}^k$ can be bounded depending on $\mathrm{d}(g_j, \mathcal{V}^k)$. This is mainly shown by exploiting the norm-minimizing property of $\mathcal{L}^k$ w.r.t. (LSq) in contrast to constant functions taking the value $g_j(v_k)$ for some $v^k \in \mathcal{V}^k$.
Therefore, for any $k \in \mathbb{N}$, we choose some $v^k \in \mathcal{V}^k$ to define the constant function $\tilde{\mathcal{L}}^k : \mathcal{B}^k_{\mathcal{J}_j} \to \mathbb{R}, x \mapsto g_j(v^k)$ and let $\tilde{w}^k \in \mathbb{R}^{l_j+1}$ be such that

$$\tilde{\mathcal{L}}^k(x) = (\tilde{w}^k)^\top \begin{pmatrix} x \\ 1 \end{pmatrix},$$

i.e., $(\tilde{w}^k)^\top = (0, \dots, 0, g_j(v^k))$. Let $C > 0$ be such that $\|x\|_\infty \le C\|x\|_2$ for any $x \in \mathbb{R}^{2^{l_j}}$. We recall that $b^k = (g_j(v^1), \dots, g_j(v^{2^{l_j}}))^\top$, denote the solution of (LSq) used for $\mathcal{L}^k$ by $\bar{w}^k$ and note that

$$\max_{v \in \mathcal{V}^k} |\mathcal{L}^k(v) - g_j(v)| = \|A^k \bar{w}^k - b^k\|_\infty,$$

since for any $l \in [2^{l_j}]$ there exists some $v \in \mathcal{V}^k$ with $A^k_l \bar{w}^k = \mathcal{L}^k(v)$ and vice versa by definition of $\mathcal{L}^k$. Consequently, by the minimization-property of $\bar{w}^k$ w.r.t. (LSq), we obtain that

$$\max_{v \in \mathcal{V}^k} |\mathcal{L}^k(v) - g_j(v)| \le C\|A^k \bar{w}^k - b^k\|_2 \le C\|A^k \tilde{w}^k - b^k\|_2 \le C\sqrt{2^{l_j}}\, \mathrm{d}(g_j, \mathcal{V}^k).$$

Now, by Lemma 5.7, there exist $K_1 \in \mathbb{N}$ such that $\mathrm{d}(g_j, \mathcal{V}^k) < \varepsilon/2$ for any $k \ge K_1$ and $K_2 \in \mathbb{N}$ such that $C\sqrt{2^l}\, \mathrm{d}(g_j, \mathcal{V}^k) < \varepsilon/4$ for any $k \ge K_2$. We set $K := \max\{K_1, K_2\}$ and obtain

$$\begin{aligned}
\mathrm{d}(\mathcal{L}^k, \mathcal{V}^k) &\le \max_{v \in \mathcal{V}^k} \mathcal{L}^k(v) - \min_{v \in \mathcal{V}^k} \mathcal{L}^k(v) \\
&\le \max_{v \in \mathcal{V}^k} (\mathcal{L}^k(v) - g_j(v)) + \max_{v \in \mathcal{V}^k} g_j(v) + \max_{v \in \mathcal{V}^k} (-\mathcal{L}^k(v) + g_j(v) - g_j(v)) \\
&\le \max_{v \in \mathcal{V}^k} (\mathcal{L}^k(v) - g_j(v)) + \max_{v \in \mathcal{V}^k} g_j(v) + \max_{v \in \mathcal{V}^k} (g_j(v) - \mathcal{L}^k(v)) + \max_{v \in \mathcal{V}^k} -g_j(v) \\
&\le \mathrm{d}(g_j, \mathcal{V}^k) + 2\max_{v \in \mathcal{V}^k} |\mathcal{L}^k(v) - g_j(v)| < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon,
\end{aligned}$$

for any $k \ge K$. The second claim follows by the fact the $\mathcal{L}^k$ is linear and therefore attains its maximal and minimal values at a corner point of $\mathcal{B}^k_{\mathcal{J}_j}$. □

Furthermore, for a given constraint function $g_j, j \in [p]$, a linear function $\mathcal{L}$ as defined in (LSq-F) w.r.t. a box $\mathcal{B} \subseteq X_{\mathcal{J}_j}$ at least *touches* the function g_j in some point $x \in \mathcal{B}$, i.e., there exists $x \in \mathcal{B}$ such that $g_j(x) = \mathcal{L}(x)$. This is derived from the optimality of the defined linear function and applying the Intermediate Value Theorem.

Lemma 5.9 *Let $j \in [p]$ and $\mathcal{B} \subseteq X_{\mathcal{J}_j}$ be a box. Furthermore, let $\mathcal{L} : \mathcal{B} \to \mathbb{R}$ be a linear function as defined in (LSq-F) w.r.t. $\mathcal{B}$ and g_j. Then, there exist corner points v^1 and v^2 of the box $\mathcal{B}$ such that $g_j(v^1) \le \mathcal{L}(v^1)$ and $\mathcal{L}(v^2) \le g_j(v^2)$. In particular, there exists $x \in \mathcal{B}$ with $g_j(x) = \mathcal{L}(x)$.*

We use the previous results to show that the under- and overestimation error of $\mathcal{L}^k$ w.r.t. g_j on a box $\mathcal{B} \subseteq X_{\mathcal{J}_j}$ are bounded and converge to zero for ongoing partitioning of the box.

Lemma 5.10 *Let $j \in [p]$. For any $k \in \mathbb{N}$, let $\mathcal{L}^k \colon \mathcal{B}^k_{\mathcal{J}_j} \to \mathbb{R}$ be a linear function as defined in (LSq-F) w.r.t. $\mathcal{B}^k_{\mathcal{J}_j}$ and g_j. Furthermore, for any $k \in \mathbb{N}$, let $\theta^{o,k}$ and $\theta^{u,k}$ be the solutions of (OEP) and (UEP), respectively, w.r.t. $\mathcal{L}^k, g_j$ and $\mathcal{B}^k_{\mathcal{J}_j}$. Then, for any $k \in \mathbb{N}$, we have that*

$$\max\{\theta^{o,k}, \theta^{u,k}\} \leq \mathrm{d}(g_j, \mathcal{B}^k_{\mathcal{J}_j}) + \mathrm{d}(\mathcal{L}^k, \mathcal{B}^k_{\mathcal{J}_j}).$$

In particular, for any $\varepsilon > 0$ there exists $K \in \mathbb{N}$ such that $\max\{\theta^{o,k}, \theta^{u,k}\} < \varepsilon$ for $k \geq K$.

Proof. Let $k \in \mathbb{N}$ and $x \in \mathcal{B}^k_{\mathcal{J}_j}$. By Lemma 5.9 there exists $x^\star \in \mathcal{B}^k_{\mathcal{J}_j}$ with $g_j(x^\star) = \mathcal{L}^k(x^\star)$. Clearly, we have that $|g_j(x) - g_j(x^\star)| \leq \mathrm{d}(g_j, \mathcal{B}^k_{\mathcal{J}_j})$ and $|\mathcal{L}^k(x^\star) - \mathcal{L}^k(x)| \leq \mathrm{d}(\mathcal{L}^k, \mathcal{B}^k_{\mathcal{J}_j})$. Together with the fact that $g_j(x^\star) = \mathcal{L}^k(x^\star)$ this yields

$$|g_j(x) - \mathcal{L}^k(x)| \leq \mathrm{d}(g_j, \mathcal{B}^k_{\mathcal{J}_j}) + \mathrm{d}(\mathcal{L}^k, \mathcal{B}^k_{\mathcal{J}_j}). \tag{5.1}$$

Now, since (5.1) holds for any $x \in \mathcal{B}^k_{\mathcal{J}_j}$, we have that

$$\max\{\theta^{o,k}, \theta^{u,k}\} \leq \max_{x \in \mathcal{B}^k_{\mathcal{J}_j}} |g_j(x) - \mathcal{L}^k(x)| \leq \mathrm{d}(g_j, \mathcal{B}^k_{\mathcal{J}_j}) + \mathrm{d}(\mathcal{L}^k, \mathcal{B}^k_{\mathcal{J}_j}),$$

which shows the first part of the result. Let $\varepsilon > 0$. Then, by Lemma 5.7 there exists $K_1 \in \mathbb{N}$ such that $\mathrm{d}(g_j, \mathcal{B}^k_{\mathcal{J}_j}) < \varepsilon/2$ for any $k \geq K_1$ and by Lemma 5.8 there exists $K_2 \in \mathbb{N}$ such that $\mathrm{d}(\mathcal{L}^k, \mathcal{B}^k_{\mathcal{J}_j}) < \varepsilon/2$ for any $k \geq K_2$. Setting $K := \max\{K_1, K_2\}$ yields the second result. □

Using the previous results, we can prove that Assumption 5.5 holds for a δ-precise relaxation refinement scheme.

Theorem 5.11 *Let $\theta > 0$. Then, there exists $K \in \mathbb{N}$ such that we have $\theta^{\mathcal{R}_K}(\tilde{x}^K) < \theta$, i.e., Assumption 5.5 holds.*

Proof. For any nonlinear constraint function $g_j, j \in [p]$, we show that there exists $K_j \in \mathbb{N}$ such that

$$\theta^{\mathcal{R}_{k_j}}_{g_j}(\tilde{x}^{k_j}) < \theta \quad \text{for any } k_j \geq K_j.$$

Note that by setting $K := \max\{K_j \mid j \in [p]\}$, one obtains the statement. Let $j \in [p]$. By Lemma 5.10, we know that there exists $K_j \in \mathbb{N}$ such that

$$\max\{\theta^{u,k}_{g_j}, \theta^{o,k}_{g_j}\} < \frac{\theta}{2} \quad \text{for any } k \geq K_j,$$

where $\theta^{o,k}_{g_j}$ and $\theta^{u,k}_{g_j}$ denote the solutions of (OEP) and (UEP), respectively, w.r.t. g_j, $\mathcal{L}^k_j$ and $\mathcal{B}^k_{\mathcal{J}_j}$. By Lemma 5.3 we have that

$$\theta^{\mathcal{R}_k}_{g_j}(\tilde{x}^k) \leq \theta^{u,k}_{g_j} + \theta^{o,k}_{g_j} < \frac{\theta}{2} + \frac{\theta}{2} = \theta \quad \text{for any } k \geq K_j,$$

so that the claim follows. □

Theorem 5.12 *Algorithm 1 with a δ-precise refinement scheme terminates correctly after finitely many steps.*

Proof. Theorem 5.11 shows that Assumption 5.5 and therefore Assumption 4.2 hold. Thus, finite and correct termination of Algorithm 1 follows by Theorem 4.5. □

6 Relaxation-refinement schemes

Whereas in the previous section, we encountered how the relaxations of the feasible set look like, the upcoming section is dedicated to the relaxation-refinement procedures. Using the concept of δ-preciseness we have seen that reaching a certain accuracy relies on successively partitioning the variable domain boxes $X_{\mathcal{J}_j}, j \in [p]$. As for each of these subboxes, a binary variable enters the relaxed problem, the complexity of these problems increases with ongoing refinement. Surely, increasing complexity is an inherent consequence when contemplating the resolution of MINLPs through progressively refined MILP relaxations. The pivotal question revolves around the magnitude of this inherent complexity and the possibility of mitigating it.

Remark 6.1 *Note that one can decrease the number of binary variables in relation to the number of used boxes by employing some different formulation techniques; cf.* [39]*. Naturally, all of these different formulations perform differently in numerical experiments* [5]*.*

Subsequently, we present different approaches to keep the relaxed problems' complexity as low as possible. The two core ideas – namely longest-edge bisection of active boxes and employing bound tightening – are well-known from the single-objective context. However, to unleash their full potential in the multi-objective setting, some adaptions have to be made as described in the following.

Refine the partitions

The first idea to be adapted to the multi-objective setting comes from [7, 8, 31] where only these parts of the respective variable domain are refined which seem to influence optimality the most. As already mentioned before, when considering multiple objective functions these parts may be widely distributed across the feasible set due to the presence of (often infinitely) many optimal value vectors. Therefore, we individualize the relaxations to specific parts of the nondominated set in the image space. The underlying hope is that specifying regions of interest restricts the relevant areas in the preimage space. This then allows us to concentrate refinement efforts only on these relevant areas.
Like [8, 11, 31] we realize this by performing a longest edge bisection of the active box of the current relaxation, i.e., the box $\mathcal{B}^k_{\mathcal{J}_j}$ with $\tilde{x}^k_{\mathcal{J}_j} \in \mathcal{B}^k_{\mathcal{J}_j}$, where k denotes the refinement step. While the sole longest edge bisection of active boxes is well-known from the single-objective setting, we do not only have one incumbent lower (and upper) bound in the multi-objective setting. Therefore, the adaption to the multi-objective case lies in choosing the relaxed solution $\tilde{x}^k$ determining the active box in a meaningful way as well as keeping track of the respective relaxations. We do this by mitigating information between the image and preimage space. In fact, every time the algorithm decides for refinement this is done while considering a specific search zone represented by a local upper bound u and having a relaxation-feasible solution $\tilde{x}^k$ coming from a relaxation which is assigned to u. We have that $f(\tilde{x}^k) \leq u - \delta e$, i.e., the area around $\tilde{x}^k$ seems to influence optimality w.r.t. the search zone determined by u. Consequently, we use $\tilde{x}^k$ to determine the active boxes and perform the well-known longest edge bisection. By doing so, image space information in the form of the search zone local upper bound u influences the way the respective relaxation is refined. Since the relaxations associated with different search zones are independent of each other, this yields differently shaped

relaxations associated with different parts of the image space. We call this procedure an *adaptive* refinement scheme.
To make the effect of this individualization visible through numerical experiments, we also introduce a more basic – a *uniform* – refinement scheme. We call it uniform because the longest edge bisection is performed for any box appearing in a partition instead of only for the active one as in the adaptive scheme below. Therefore, like in [28], only the decision for refinement is individualized across the image space – but not the decision on how to refine. This is why we claim that the adaptive refinement technique deepens the mitigation of information between image and preimage space in comparison to [28].
The algorithm outlining the adaptive refinement scheme is detailed in Algorithm 4.

Algorithm 4 `adaptive_refinement` routine

```
Input: current relaxation information $\mathcal{R}_{\text{current}}$, current solution $\tilde{x}$ of relaxed problem,
    constraint satisfaction tolerance $\theta$, longest edge tolerance $\delta_{\text{edge}}$
 1: Set $\mathcal{R}_{\text{new}} = \emptyset$
 2: for $j \in [p]$ do
 3:     if $\theta_{g_j}^{\mathcal{R}_{\text{current}}}(\tilde{x}) < \theta$ then
 4:         Set $\mathcal{R}_{\text{new}}.g_j = \mathcal{R}_{\text{current}}.g_j$
 5:     else
 6:         Set $\mathcal{R}_{\text{new}}.g_j = \emptyset$
 7:         for $\mathcal{B}$ appearing in $\mathcal{R}_{\text{current}}.g_j$ do
 8:             if $\mathcal{B}$ is active, i.e., $\tilde{x}_{\mathcal{J}_j} \in \mathcal{B}$ then
 9:                 if $\lambda(\mathcal{B}) \geq \delta_{\text{edge}}$ then
10:                     Choose longest edge with smallest index $l' \in [l_j]$
11:                     Compute midpoint $m_{l'} = (x_{l'}^u - x_{l'}^\ell)/2$
12:                     Append box $\mathcal{B}_1$ defined by $\mathcal{B}$ with $x_{l'}^u$ replaced by $m_{l'}$ to $\mathcal{R}_{\text{new}}.g_j$
13:                     Append box $\mathcal{B}_2$ defined by $\mathcal{B}$ with $x_{l'}^\ell$ replaced by $m_{l'}$ to $\mathcal{R}_{\text{new}}.g_j$
14:                 else
15:                     Set $\mathcal{R}_{\text{new}}.g_j = \mathcal{R}_{\text{current}}.g_j$
16:                     break
17:                 end if
18:             else
19:                 Append $\mathcal{B}$ to $\mathcal{R}_{\text{new}}.g_j$
20:             end if
21:         end for
22:     end if
23: end for
Output: refined relaxation $\mathcal{R}_{\text{new}}$
```

The only difference between the uniform refinement scheme and the adaptive one is that one bisects the longest edge of every box instead of only the one of the active box, i.e., one omits the `if-else`-statement in lines 8 and 19 in Algorithm 4. We refer to that as the *uniform variant* of Algorithm 4.
It is worth noting that when the longest edge length tolerance δ_{edge} and θ are appropriately selected, a largest edge length smaller than δ_{edge} guarantees a maximal constraint satisfaction error less than θ. As a consequence, asking if $\theta_{g_j}^{\mathcal{R}_{\text{current}}}(\tilde{x}) < \theta$, can be omitted in Algorithm 4 as well as for its uniform variant. Nevertheless, we retain this

notation to emphasize the possibility of the procedure terminating even before reaching the δ_{edge}-criterion. We obtain the following correctness and finiteness results in regard to Algorithm 4.

Lemma 6.2 *Let $j \in [p]$ and let $(\mathcal{B}_i)_{i\in[s]}$ be a partition of the original variable domain box $X_{\mathcal{J}_j}$ belonging to a relaxation $\mathcal{R}_{\text{current}}$. Furthermore, let $\tilde{x} \in S^{\mathcal{R}_{\text{current}}}$, $\theta > 0$ and $\delta_{\text{edge}} > 0$ be the input of* Algorithm 4*. Then,* Algorithm 4 *returns a partition $(\tilde{\mathcal{B}}_i)_{i\in[\tilde{s}]}$ of $X_{\mathcal{J}_j}$ for some $s \leq \tilde{s} \leq s+1$ which is a refinement of $(\mathcal{B}_i)_{i\in[s]}$.*

One can prove that both Algorithm 4 and its uniform variant are δ-precise. To do so, one can closely follow the proofs of [8, Theorems 3.11 and 3.12]. Only the simplex-based partitioning structure has to be replaced with the box-based one, and one has to adapt to the slightly different notion of δ-preciseness used in the present work.

Proposition 6.3 *The refinement procedure presented in* Algorithm 4 *and its uniform variant are δ-precise.*

By Theorem 5.12 this immediately implies finiteness and correctness of Algorithm 1 with Algorithm 4 or its uniform variant as refinement scheme.

Corollary 6.4 Algorithm 1 *with* Algorithm 4 *or its uniform variant as refinement scheme terminates correctly after finitely many steps.*

Bound tightening

The second idea for keeping the complexity of the relaxed problems as low as possible is utilizing domain reduction techniques. For a detailed overview of this topic, we refer to [35]. Note that especially in many real-world applications no tight variable bounds are a-priorily available and therefore reducing the domain might become a crucial step towards successfully solving such problems. Obviously, having a smaller domain there is the well-founded hope that fewer boxes – and therefore fewer additional binary variables – are necessary to reach a relaxation with adequate accuracy.
The domain reduction technique relevant for this work is so-called *optimization-based bound tightening* (OBBT). For any variable x_l appearing in a nonlinear constraint function $g_j, j \in [p]$ and a given relaxation $\mathcal{R}$ one solves the problems

$$\min \pm x_l \quad \text{s.t.} \quad x \in S^{\mathcal{R}}. \tag{OBBT($\mathcal{R}$)(l)}$$

This method was first mentioned in the context of global optimization by [37] and subsequently applied at the root node of several tree search-based optimization algorithms (cf. [29, 36, 38]).
If one has an upper bound u on the optimal value, one can strengthen the effect of OBBT by including the objective cut-off $f(x) \leq u$ into (OBBT($\mathcal{R}$)(l)) as introduced in [41]. This yields problems like

$$\min \pm x_l \quad \text{s.t.} \quad x \in S^{\mathcal{R}},\ f(x) \leq u. \tag{OBBT($\mathcal{R}, u$)(l)}$$

Note that since we assume the components of f to be linear, (OBBT($\mathcal{R}, u$)(l)) is a MILP. In [41], this was applied at every node throughout a spatial branch-and-bound tree yielding a so-called *branch-and-contract* framework. In [20], the authors present numerical results showing that employing OBBT occasionally during a spatial branch-and-bound tree search can have computational benefits – especially for large problems.

To the best of our knowledge, when considering relax-and-refine approaches, bound tightening procedures are only applied *before* successively refining the relaxations as done in [31]. Instead of this two-phase approach – tighten the bounds first, partition the domain afterward – we present a relax-and-refine framework for multi-objective optimization that can switch between these two tasks. This is motivated by two reasons. The first one are the results from [20] for tree-based algorithms. There, cutting away irrelevant parts of the domain results in a possibly smaller number of subproblems that have to be solved to close the optimality gap. Aiming for a smaller number of subproblems that have to be solved in the tree-based algorithms transfers to aiming for a smaller number of boxes that are needed for an adequately accurate relaxation in our case.

The second reason originates from the differences between the single-objective and the multi-objective case. In fact, when applying OBBT with an objective cut-off u, i.e., solving ($\text{OBBT}(\mathcal{R}, u)(l)$), to a multi-objective problem one has to ensure that $y \leq u$ for any $y \in \mathcal{N}$. Otherwise, one would drop parts of the domain contributing to the nondominated set. Therefore, when employing it as a first-phase method, one cannot exclude any part of the domain containing an optimal solution – and they may be widely distributed across the feasible set as explained earlier. Note that it might even be the case that the whole feasible set contributes to the nondominated set. Thus, utilizing an upper bound u with $y \leq u$ for any $y \in \mathcal{N}$ for OBBT may be not that effective for significantly reducing the domain.

The idea is the following: when employing OBBT *during* the procedure, we may have different objective cut-offs – e.g., local upper bounds – available each of which specifies only a certain region in the image space. Consequently, with ongoing progress of the algorithms, i.e., when the local upper bounds specify a smaller and smaller part of the nondominated set, larger parts of the preimage space become irrelevant w.r.t. to the respective objective cut-offs. On the other hand, applying OBBT with a local upper bound u representing a specific search zone as objective cut-off makes the resulting shrunk feasible set useless for any other search zone represented by another local upper bound u' with $c(u') \nsubseteq c(u)$ where $c(u)$ denotes the lower search zone of u as introduced in Definition 3.4. In particular, this means that the mediation of image and preimage space information in order to individualize the relaxations to the specific search zones intensifies even more.

To formalize the above, we introduce some theoretical notions. We call them theoretical because we only need them to prove correctness of our procedure. We start with the possibly shrunk feasible set $S^y := \{x \in S \mid f(x) \leq y\}$ w.r.t. a point $y \in \mathbb{R}^r$. Given $y \in \mathbb{R}^r$ such that the set S^y is nonempty, for any $j \in [p]$ we denote by

$$X^y_{\mathcal{J}_j} = [x^{\ell,y}_{\mathcal{J}_j}, x^{u,y}_{\mathcal{J}_j}] \subseteq X_{\mathcal{J}_j}$$

the admissible variable domain for g_j within S^y. Having a local upper bound u, the set S^u consists of all feasible solutions satisfying the search zone constraint $f(x) \leq u$. Note that setting the optimal values of ($\text{OBBT}(\mathcal{R}, u)(l)$) with $S^{\mathcal{R}} = S$ as lower, respective upper bounds to all variables $x_l, l \in [n]$, yields a superset of S^u. Similarly, if we use a relaxation $S^{\mathcal{R}} \supseteq S$ and apply ($\text{OBBT}(\mathcal{R}, u)(l)$) only to variables appearing in nonlinear constraints $g_j, j \in [p]$, and denote the respective solutions and new variable bounds by $x^{\ell,(\mathcal{R},y)}_l$ and $x^{u,(\mathcal{R},y)}_l$, where $l \in \mathcal{J}_j$, we have that

$$x^\ell_l \leq x^{\ell,(\mathcal{R},y)}_l \leq x^{\ell,y}_l \leq x^{u,y}_l \leq x^{u,(\mathcal{R},y)}_l \leq x^u_l.$$

Note that at this point we fix the order of the variables to be considered to one given in Algorithm 5. That is, we start with $j = 1 \in [p]$ and continue with $l \in \mathcal{J}_j$ in ascending order according to the order within $\mathcal{J}_j \subseteq [n]$. Afterward, we repeat with $j = 2 \in [p]$. As a direct consequence from the inclusion $S \subseteq S^{\mathcal{R}}$ we have that $S^y \subseteq S^{(\mathcal{R},y)}$.

Corollary 6.5 *Let $y \in \mathbb{R}^r$, $\mathcal{R}$ be a relaxation of* (MOP)*, and let $\tilde{\mathcal{R}}$ be a relaxation of $S^{(\mathcal{R},y)}$. Then, we have that $f(S^y) \subseteq f(S^{\tilde{\mathcal{R}}})$ and for any $\tilde{y} \in \mathcal{N}^{S^{\tilde{\mathcal{R}}}}$ we have that $\tilde{y} \in \mathbb{R}^r \setminus (f(S^y) + \operatorname{int}(\mathbb{R}^r_+))$.*

In Algorithm 5 the procedure for applying optimization-based bound tightening is presented.
Note that once new bounds for a variable x_l are computed in line 7, all boxes, i.e., any box for any nonlinear constraint, are adapted or dismissed w.r.t. this new variable bound. This is what happens from line 8 to line 25.

Lemma 6.6 *Let $\mathcal{R}_{\text{start}}$, $\tilde{x}$, $\theta > 0$, and $\hat{u}$ be the input of* Algorithm 5*. We denote the relaxation after* Algorithm 5 *by $\mathcal{R}_{\text{end}}$. Then, for any nonlinear constraint $g_j, j \in [p]$, of* (MOP)*, the elements of $\mathcal{R}_{\text{end}}.g_j$ form a partition of the box domain $X_{\mathcal{J}_j}^{(\mathcal{R}_{\text{start}},\hat{u})}$ of g_j belonging to $S^{(\mathcal{R}_{\text{start}},\hat{u})}$.*

Proof. Let $j \in [p]$. We start by showing that the union of the boxes from $\mathcal{R}_{\text{end}}.g_j$ coincides with the box domain $X_{\mathcal{J}_j}^{(\mathcal{R}_{\text{start}},\hat{u})}$. Let $x' \in S^{(\mathcal{R}_{\text{start}},\hat{u})}$. Then, there exists a box $\mathcal{B}_{\text{start}}$ belonging to $\mathcal{R}_{\text{start}}.g_j$ with $x' \in \mathcal{B}_{\text{start}}$ since $S^{(\mathcal{R}_{\text{start}},\hat{u})} \subseteq S^{\mathcal{R}_{\text{start}}}$. If $\mathcal{B}_{\text{start}}$ belongs to $\mathcal{R}_{\text{end}}.g_j$, then clearly x' belongs to the union of the boxes in $\mathcal{R}_{\text{end}}$. If otherwise, $\mathcal{B}_{\text{start}}$ does not belong to $\mathcal{R}_{\text{end}}.g_j$ anymore, then $\mathcal{B}_{\text{start}}$ was adjusted and/or removed. We observe that for any variable x_l appearing in g_j we have $x_l^{\ell,\mathcal{B}_{\text{start}}} \leq x'_l \leq x_l^{u,\mathcal{B}_{\text{start}}}$ and $x_l^{\ell,(\mathcal{R}_{\text{start}},\hat{u})} \leq x'_l \leq x_l^{u,(\mathcal{R}_{\text{start}},\hat{u})}$ since $x' \in S^{(\mathcal{R}_{\text{start}},\hat{u})} \subseteq S^{\mathcal{R}_{\text{start}}}$. This implies that $x_l^{\ell,(\mathcal{R}_{\text{start}},\hat{u})} \leq x'_l \leq x_l^{u,\mathcal{B}_{\text{start}}}$ and $x_l^{\ell,\mathcal{B}_{\text{start}}} \leq x'_l \leq x_l^{u,(\mathcal{R}_{\text{start}},\hat{u})}$, and particularly that the box $\mathcal{B}_{\text{start}}$ was not removed without adjustment. Consequently, an adjustment of $\mathcal{B}_{\text{start}}$ belongs to $\mathcal{R}_{\text{end}}.g_j$ or has been removed due to redundancy. If it was removed by redundancy there exists at least one more box $\mathcal{B}'_{\text{start}}$ with $x' \in \mathcal{B}'_{\text{start}}$ and the box itself or an adjustment belongs to $\mathcal{R}_{\text{end}}.g_j$. By switching names, we may assume that an adjusted box $\mathcal{B}_{\text{end}}$ of $\mathcal{B}_{\text{start}}$ belongs to $\mathcal{R}_{\text{end}}.g_j$. For this box $\mathcal{B}_{\text{end}}$ and any $l \in \mathcal{J}_j$,

$$x_l^{\ell,\mathcal{B}_{\text{end}}} = \max\left\{x_l^{\ell,\mathcal{B}_{\text{start}}}, x_l^{\ell,(\mathcal{R}_{\text{start}},\hat{u})}\right\} \quad \text{and} \quad x_l^{u,\mathcal{B}_{\text{end}}} = \min\left\{x_l^{u,\mathcal{B}_{\text{start}}}, x_l^{u,(\mathcal{R}_{\text{start}},\hat{u})}\right\}$$

hold and therefore $x' \in \mathcal{B}_{\text{end}}$ as required.
To conclude this proof we note that for any box $\mathcal{B}_{\text{start}}$ belonging to $\mathcal{R}_{\text{start}}.g_j$ there exists at most one box $\mathcal{B}_{\text{end}}$ belonging to $\mathcal{R}_{\text{end}}.g_j$ with $\mathcal{B}_{\text{end}} \subseteq \mathcal{B}_{\text{start}}$. This is because no splitting occurs during Algorithm 5 and the elimination of redundant boxes. Together with the fact that the boxes in $\mathcal{R}_{\text{start}}.g_j$ yield a partition of the corresponding box domain of g_j, we obtain that for two boxes $\mathcal{B}_1$ and $\mathcal{B}_2$ belonging to $\mathcal{R}_{\text{end}}.g_j$ it holds that

$$\operatorname{int}(\mathcal{B}_1) \cap \operatorname{int}(\mathcal{B}_2) = \emptyset.$$

Thus, the boxes appearing in $\mathcal{R}_{\text{end}}.g_j$ form a partition of the box domain $X_{\mathcal{J}_j}^{(\mathcal{R}_{\text{start}},\hat{u})}$ of g_j belonging to $S^{(\mathcal{R}_{\text{start}},\hat{u})}$. $\square$

Algorithm 5 `optimization-based bound tightening` routine

```
Input: current relaxation information $\mathcal{R}_{\text{current}}$, current solution $\tilde{x}$ to relaxed problem, constraint satisfaction tolerance $\theta > 0$, local upper bound $\hat{u}$ representing the current search zone
 1: Set tightened_vars = [ ]
 2: Set $\mathcal{R}_{\text{start}} = \mathcal{R}_{\text{current}}$
 3: for $j \in [p]$ do
 4:     if $\theta_{g_j}^{\mathcal{R}_{\text{start}}}(\tilde{x}) \geq \theta$ then
 5:         for $l \in \mathcal{J}_j$ do
 6:             if $x_l$ not in tightened_vars then
 7:                 Solve (OBBT$(\mathcal{R}, u)(l)$) with $u = \hat{u}$ and $\mathcal{R} = \mathcal{R}_{\text{start}}$ and denote solutions by $x_l^{\ell,(\mathcal{R}_{\text{start}},\hat{u})}$, $x_l^{u,(\mathcal{R}_{\text{start}},\hat{u})}$
 8:                 for $j \in [p]$ do
 9:                     if $x_l$ appears in $g_j$ then
10:                         for $\mathcal{B}$ appearing in $\mathcal{R}_{\text{current}}.g_j$ do
11:                             if $x_l^{\ell,\mathcal{B}} < x_l^{\ell,(\mathcal{R}_{\text{start}},\hat{u})} \leq x_l^{u,\mathcal{B}}$ then
12:                                 Set $x_l^{\ell,\mathcal{B}} = x_l^{\ell,(\mathcal{R}_{\text{start}},\hat{u})}$
13:                             else if $x_l^{u,\mathcal{B}} < x_l^{\ell,(\mathcal{R}_{\text{start}},\hat{u})}$ then
14:                                 Remove $\mathcal{B}$ from $\mathcal{R}_{\text{current}}.g_j$
15:                                 continue
16:                             end if
17:                             if $x_l^{\ell,\mathcal{B}} \leq x_l^{u,(\mathcal{R}_{\text{start}},\hat{u})} < x_l^{u,\mathcal{B}}$ then
18:                                 Set $x_l^{u,\mathcal{B}} = x_l^{u,(\mathcal{R}_{\text{start}},\hat{u})}$
19:                             else if $x_l^{u,(\mathcal{R}_{\text{start}},\hat{u})} < x_l^{\ell,\mathcal{B}}$ then
20:                                 Remove $\mathcal{B}$ from $\mathcal{R}_{\text{current}}.g_j$
21:                             end if
22:                         end for
23:                         Remove redundant boxes in $\mathcal{R}_{\text{current}}.g_j$
24:                     end if
25:                 end for
26:                 Append $x_l$ to tightened_vars
27:             end if
28:         end for
29:     end if
30: end for
Output: relaxation $\mathcal{R}_{\text{current}}$ with tightened bounds
```

Applying the bound tightening scheme in Algorithm 5 is computationally expensive. In fact, for any variable x_l whose bounds should be tightened, one solves two MILP relaxations of (MOP). Especially since these relaxations get finer and finer, the relaxed problems get more complex. On the other hand, putting effort into the bound tightening procedure may prevent the relaxations from getting too complex or even reduce their complexity. Having these two sides in mind, one has to carefully balance the additional computational effort and the possible benefit in terms of complexity regarding the relaxations. In [20], the authors discuss ideas when bound tightening is most efficient during the optimization process. However, for this work, we restrict ourselves to very basic variants as can be seen in Section 7 where numerical experiments are presented.

7 Numerical experiments

This section is dedicated to numerical experiments with the algorithms presented. All numerical computations in this paper are realized using the Pyomo suite [9, 24] on a machine with Intel Core i7-8565U processor and 32GB of RAM. While the general framework is implemented using the Pyomo Modeling, we use the SCIP Optimization Suite [4] for globally solving the continuous nonlinear problems arising when computing the overestimation error in OEP. As MILP-solver we use Gurobi [22].
In the following, we consider problems, where not every component of the objective function is linear. To deal with this algorithmically, we compute an underestimation function for the nonlinear components as described above for nonlinear constraints. Afterward, we replace the nonlinear objective component with the piecewise linear underestimation function and preserve the lower bounding property of the relaxed problems.
For each of the following test instances we utilize Algorithm 1 with the uniform refinement scheme (UN), and the adaptive refinement scheme with (AD-BT) and without bound tightening (AD). In the variant AD-BT we apply the bound tightening procedure every third time a relaxation or its refinement is assigned to a new local upper bound. In addition, we consider two variants for dealing with the reduced problem (pMOP($\hat{x}_I$)) in Algorithm 3. In the first variant, we only solve it to feasibility and in the second one to global optimality.
Throughout this section, we use the following algorithm parameters: $\varepsilon_{\text{encl}} = 0.1$, $\delta = 0.095$ and $\theta =$ `1e-4` as well as a time limit of `maxtime` $= 3600s$ and a maximal number of iterations of `maxiter` $= 1000$.
We start with an illustrative tri-objective mixed-integer non-convex non-quadratic problem [14, P2]:

$$\begin{aligned}\min_{x} \quad & \begin{pmatrix} x_1 + x_4 \\ x_2 - x_4 \\ x_3 - \exp(x_4) - 3 \end{pmatrix} \\ \text{s.t.} \quad & x_1^2 + x_2^2 \leq 1, \\ & \exp(x_3) \leq 1, \\ & x_1 x_2 (1 - x_3) \leq 1, \\ & x \in [-2, 2]^3 \times ([-2, 2] \cap \mathbb{Z}). \end{aligned} \tag{P1}$$

The corresponding computational results are reported in Table 1.
One can see that using the uniform refinement scheme does not yield a satisfactory result within the time limit for both NLP-variants. An average number of around 212.3, respectively, 554.3 additional binary variables indicates a significantly increased complexity of the relaxed problems. In some cases, even over 5.000, respectively, 10.000 binary variables are added to the problem, making it comparably hard to solve. In contrast to that, one observes an enormous advantage in computational time when using the adaptive refinement scheme. Here, the size of the relaxed problems decreases significantly. There are only around 36.3, respectively, 29.6 binary variables added on average for the 5241, respectively, 469 MILP relaxations that have to be solved. Applying bound tightening, the computational time decreases even more. The reason

	feasibility NLP			global NLP		
	UN	AD	AD-BT	UN	AD	AD-BT
CPU (in s)	$\star$	1760.6	1216.3	$\star$	196.6	107.3
# of iterations	18	53	49	7	24	24
max # of preimage set boxes (MPSB)	5122	76	36	10257	59	26
average # of PSB (APSB)	212.4	36.3	14.9	554.3	29.6	12.3
# of solved relaxed problems (RP)	598	5241	3877+2443	140	469	467+399

Table 1: Numerical results for (P1). $\star$ stands for an exit of the method because of the time limit. For the AD-BT variant we count the number of MILP relaxations solved for tackling the search zones as well as the ones solved for tightening the variable bounds.

for this is a significantly smaller average of preimage set boxes, namely only around 14.9, respectively, 12.3, i.e., again a reduction of size of the relaxed problems. This results in a reduced computational time of 1216.3s, respectively, 107.3s with bound tightening opposing 1760.6s, respectively, 196.6s without bound tightening. Furthermore one can see a very significant advantage of solving the NLPs to global optimality instead of just to feasibility. While there is no significant difference in the size of the relaxed problems, there is a huge increase in the number of problems that have to be solved when using the feasibility approach compared to the case of solving the reduced NLPs to global optimality. One reason for this might be better progress in the image space when using nondominated points of the patch problems (pMOP($\hat{x}_I$)) instead of only feasible ones for the updates of the potentially nondominated set as well as the set of local upper bounds.
We continue with the attempt to illustrate the functionality of the adaptive refinement scheme and the possibility of tightening the variable bounds during the procedure. Note that for problem (P1), all five possible integer assignments contribute to the feasible set. In Figures 2a and 2b, different domain partitionings of the variable domain $[0,1] \times [-4,1]$ of the function $f_3(x) = x_3 - \exp(x_4) - 3$ (depicted in Figure 3) corresponding to different search zones are depicted. Figure 2a shows the respective partitionings obtained by employing the adaptive refinement scheme, whereas Figure 2b shows the partitionings when bound tightening is additionally applied. For each of the two variants, we report the partitioning of the variable domain $[0,1] \times [-4,1]$ associated with three different search zones. That is, the left partitionings belong to the respective relaxations associated with the search zone determined by the local upper bound $(-2.83, 1.64, -0.12)^\top$, the midparts are associated with the search zone determined by the local upper bound $(0.30, -1.45, -7.01)^\top$ and the right partitionings with the search zone determined by the local upper bound $(1.26, -2.45, -11.59)^\top$.
One can clearly see the effect of bound tightening as for all three cases the partitions consists of only one box contrasting multiple boxes for the sole adaptive refinement scheme. Note further, that the bound tightening reduces the dimension of the boxes as it fixes the values of the discrete variable x_4 to the respective relevant values. Nevertheless, one can also observe the concentration to only certain parts of the variable domain box by the adaptive refinement scheme.

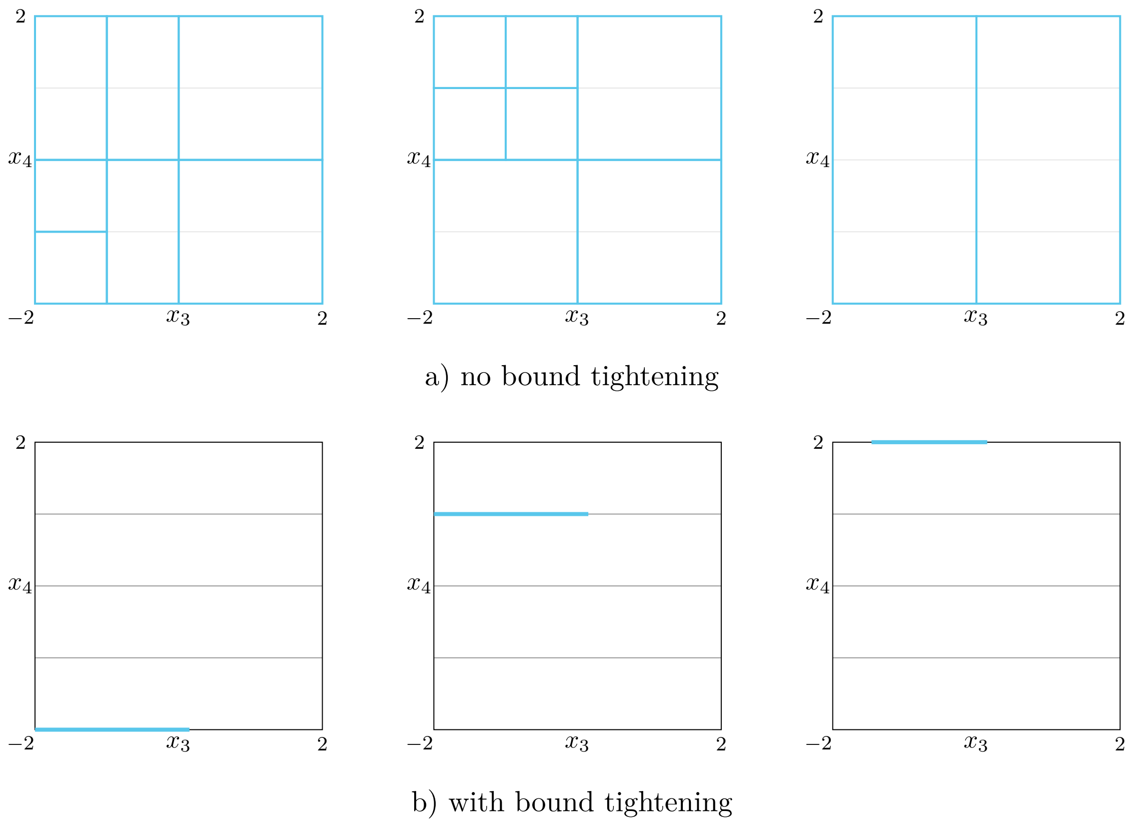

Figure 2: Domain partitionings of the domain belonging to $f_3(x) = x_3 - \exp(x_4) - 3$ corresponding to different search regions in the image space as performed by the Algorithm 1.

We continue with a scalable bi-objective quadratically constrained problem [14, P3]:

$$\begin{aligned} \min_x \; & \begin{pmatrix} \sum_{i=1}^{k/2} x_i + \sum_{i=k+1}^{k+l/2} x_i \\ \sum_{i=k/2+1}^{k} x_i + \sum_{i=k+l/2+1}^{n} x_i \end{pmatrix} \\ \text{s.t.} \; & \sum_{i=1}^{k} x_i^2 \geq 1, \quad \sum_{i=k+1}^{n} x_i^2 \leq 9, \\ & x_i \in [0,1] \text{ for } i \in [k], \quad x_i \in [-3,3] \cap \mathbb{Z}, i \in [n] \setminus [k]. \end{aligned} \tag{P2}$$

Problem (P2) is scalable in the even number $k \in \mathbb{N}$ of continuous variables as well as in the even number $l \in \mathbb{N}$ of integer variables. This sums up to a number of $n = k + l$ variables in total. The numerical results for different configurations of (P2) are displayed in Table 2.
One can see that the uniform approach is outperformed in terms of computational time by the other two approaches for all configurations of (P2), except for $(k = 2, l = 2)$ together with the global NLP approach. In this case, the uniform refinement method is faster than the AD-BT variant. However, one can observe that (for all correctly terminated instances) using the UN approach, the number of MILPs that have to be solved is smaller compared to the other approaches. This is not surprising at all as the uniform partitioning approach bisects all of the boxes in one refinement step

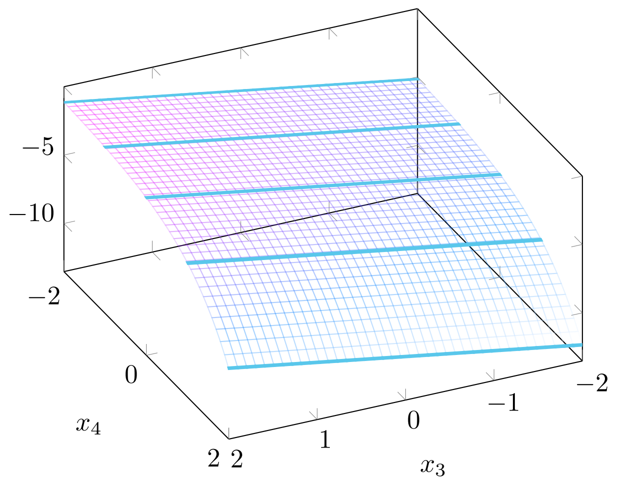

Figure 3: Surface of the function $f_3 = x_3 - \exp(x_4) - 3$ on the domain $[-2, 2] \times ([-2, 2] \cap \mathbb{Z})$.

and therefore possibly fewer relaxed problems have to be solved to approximate the relevant area in the preimage space fine enough compared to the adaptive approach. Nevertheless, the results displayed in Table 2 show that the benefit of solving smaller MILP relaxations is higher than the cost of solving more of them, i.e., the sparsity in the partitionings is more important than the number of problems solved. Note at this point that using the uniform refinement method, a lot more (OEP) have to be solved globally, as each box belonging to the partition is bisected. The increased number of (OEP) together with the increase in the size of the MILP relaxations are two reasons for the obvious inferiority of the uniform approach compared to the other two.
This clarity does not transfer to the question of whether applying bound tightening or not. While the positive impact of bound tightening is obvious when comparing the average size of the MILP relaxations (APSB), it is not that clear when comparing computational times. There are configurations of (P2) for each of the two adaptive variants being superior to the other. However, especially in the case of solving the NLPs (pMOP($\hat{x}_I$)) to feasibility, the AD-BT variant outperforms the AD variant quite clearly for most of the configurations – this might become apparent when the NLP part of the MINLP in question is too hard for solving to global optimality.
When comparing the feasible NLP approach with the global one, one observes that there is no significant difference in computational time for most configurations when using the AD-BT variant. This opposes the other two refinement schemes, where the difference in computational times is tremendous. For all cases the number of MILP relaxations increases. This is due to weaker progress in the image space for the single search zones visits as we only use feasible points instead of patch-nondominated ones for the update of the potentially nondominated set.
The results in Table 2 indicate that it seems to be beneficial to apply bound tightening when the number of continuous variables increases compared to the number of integer ones. Note that no variant succeeded in solving any configuration of (P2) with $l \geq 8$ within a reasonable time limit.
Note further that all of the continuous, respectively, integer variables appear in one of the constraints. Consequently, k, respectively l, also determine the dimension of the original variable domain of the corresponding constraint. Using, e.g., a simplex-based partitioning scheme to triangulate the six-dimensional variable domain one would have

to use at least 308 and in case of eight-dimensional variable domains more than 1500 simplices.

8 Conclusion

In the present paper, we have introduced an algorithm for computing pseudo enclosures of the nondominated set of multi-objective mixed-integer nonconvex problems. The algorithm allows to refine the piecewise linear outer approximations in an adaptive manner guided by image space information. The piecewise linear outer approximations are based on a novel box-based relaxation technique for general nonconvex terms which allows to additionally apply bound tightening during the procedure using better and better image space information in form of objective cut-offs.
While the proposed method performs overall very well for the above toy instances, it has clear limitations when facing problems that either have a more complex structure or are larger. To overcome this relevant issue, some numerical tuning has to be done in the future. Possible ideas for that are the following: for larger problems it is not a good idea to apply bound tightening to each and every variable appearing in a nonlinear constraint every time. Therefore, strategies, e.g., based on the results in [20], for determining when and which bound tightening problems should be solved are needed. Furthermore, as proposed in [8], one can think of selection strategies of the constraints for which a partioning refinement should be performed. Although the box-based partitioning scheme is able to handle dimensions larger than two (see the above results), it might be crucial of using expression trees for splitting constraints where many variables appear. Lastly, when the fix-integer-assignment strategy fails to find a feasible point, one might implement a feasibility version of the original problem together with the search zone constraint. By doing so, one avoids a lengthy (and costly) refinement-after-refinement sequence without any progress. First experiments have shown that utilizing the last of the above modifications, the computational time for the configuration ($k = 2, l = 8$) of (P2) could be reduced from no termination within $10h$ to correct termination within around $1.000s$.

Acknowledgments

The first and last author have been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 432218631.
The second and the third author are funded by the DFG via the German Excellence Strategy.

References

[1] Claire S. Adjiman and Christodoulos A. Floudas. “Rigorous convex underestimators for general twice-differentiable problems”. *J. Global Optim.* 9.1 (1996), pp. 23–40. DOI: 10.1007/BF00121749.

[2] Evelyn M.L. Beale and John H. Forrest. “Global optimization using special ordered sets”. *Math. Program.* 10.1 (1976), pp. 52–69. DOI: 10.1007/BF01580653.

[3] Pietro Belotti et al. “Mixed-integer nonlinear optimization”. *Acta Numer.* 22 (2013), pp. 1–131. DOI: 10.1017/S0962492913000032.

[4] Ksenia Bestuzheva et al. *The SCIP Optimization Suite 8.0.* Technical Report. Optimization Online, Dec. 2021.

[5] Kristin Braun and Robert Burlacu. *A Computational Study for Piecewise Linear Relaxations of Mixed-Integer Nonlinear Programs.* Optimization Online. 2023.

[6] Regina S. Burachik, C. Yalçın Kaya, and Mohammed Mustafa Rizvi. “Algorithms for generating Pareto fronts of multi-objective integer and mixed-integer programming problems”. *Eng. Optimiz.* 54.8 (2022), pp. 1413–1425. DOI: 10.1080/0305215X.2021.1939695.

[7] Robert Burlacu. “On refinement strategies for solving MINLPs by piecewise linear relaxations: a general red refinement”. *Optim. Lett.* 16 (2021), pp. 635–652. DOI: 10.1007/s11590-021-01740-1.

[8] Robert Burlacu, Björn Geißler, and Lars Schewe. “Solving mixed-integer nonlinear programmes using adaptively refined mixed-integer linear programmes”. *Optim. Methods Softw.* 35.1 (2020), pp. 37–64. DOI: 10.1080/10556788.2018.1556661.

[9] Michael L. Bynum et al. *Pyomo – Optimization Modeling in Python.* Springer, 2021. DOI: 10.1007/978-3-030-68928-5.

[10] Kerstin Dächert and Katrin Teichert. *An improved hyperboxing algorithm for calculating a Pareto front representation.* https://arxiv.org/abs/2003.14249. 2020.

[11] Marianna De Santis, Gabriele Eichfelder, Julia Niebling, and Stefan Rocktäschel. “Solving Multiobjective Mixed Integer Convex Optimization Problems”. *SIAM J. Optimiz.* 30.4 (2020), pp. 3122–3145. DOI: 10.1137/19M1264709.

[12] Matthias Ehrgott. *Multicriteria Optimization.* Springer, 2005. DOI: 10.1007/3-540-27659-9.

[13] Gabriele Eichfelder, Peter Kirst, Laura Meng, and Oliver Stein. “A general branch-and-bound framework for continuous global multiobjective optimization”. *J. Global Optim.* 80 (2021), pp. 195–227. DOI: 10.1007/s10898-020-00984-y.

[14] Gabriele Eichfelder, Oliver Stein, and Leo Warnow. “A Solver for Multiobjective Mixed-Integer Convex and Nonconvex Optimization”. *J. Optimiz. Theory App.* (2023). DOI: 10.1007/s10957-023-02285-2.

[15] Gabriele Eichfelder and Leo Warnow. “A hybrid patch decomposition approach to compute an enclosure for multi-objective mixed-integer convex optimization problems”. *Math. Method Oper. Res.* (2023). DOI: 10.1007/s00186-023-00828-x.

[16] Gabriele Eichfelder and Leo Warnow. “An approximation algorithm for multi-objective optimization problems using a box-coverage”. *J. Global Optim.* 83.2 (2022), pp. 329–357. DOI: 10.1007/s10898-021-01109-9.

[17] Levent Eriskin, Mumtaz Karatas, and Yu-Jun Zheng. “A robust multi-objective model for healthcare resource management and location planning during pandemics”. *Ann. Oper. Res.* 335.3 (2024), pp. 1471–1518. DOI: 10.1007/s10479-022-04760-x.

[18] Sunney Fotedar, Ann-Brith Strömberg, and Torgny Almgren. “Bi-objective optimization of the tactical allocation of job types to machines: mathematical modeling, theoretical analysis, and numerical tests”. *Int. T. Oper. Res.* 30.6 (2023), pp. 3479–3507. DOI: 10.1111/itor.13180.

[19] Björn Geißler, Alexander Martin, Antonio Morsi, and Lars Schewe. “Using piecewise linear functions for solving MINLPs”. In: *Mixed integer nonlinear programming.* Vol. 154. IMA Vol. Math. Appl. Springer, New York, 2012, pp. 287–314. DOI: 10.1007/978-1-4614-1927-3_10.

[20] Ambros M. Gleixner, Timo Berthold, Benjamin Müller, and Stefan Weltge. “Three enhancements for optimization-based bound tightening”. *J. Global Optim.* 67.4 (2017), pp. 731–757. DOI: 10.1007/s10898-016-0450-4.

[21] Julia Grübel, Richard Krug, Martin Schmidt, and Winnifried Wollner. “A successive linear relaxation method for MINLPs with multivariate Lipschitz continuous nonlinearities”. *J. Optimiz. Theory Appl.* 198.3 (2023), pp. 1077–1117. DOI: 10.1007/s10957-023-02254-9.

[22] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual.* 2023.

[23] Pascal Halffmann, Luca E. Schäfer, Kerstin Dächert, Kathrin Klamroth, and Stefan Ruzika. “Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey”. *J. Multi-Criteria Decis. Anal.* 29.5-6 (2022), pp. 341–363. DOI: 10.1002/mcda.1780.

[24] William E. Hart, Jean-Paul Watson, and David L. Woodruff. “Pyomo: modeling and solving mathematical programs in Python”. *Math. Program. Comp.* 3.3 (2011), pp. 219–260. DOI: 10.1007/s12532-011-0026-8.

[25] Kathrin Klamroth, Renaud Lacour, and Daniel Vanderpooten. “On the representation of the search region in multi-objective optimization”. *Eur. J. Oper. Res.* 245.3 (2015), pp. 767–778. DOI: 10.1016/j.ejor.2015.03.031.

[26] Jon Lee and Sven Leyffer, eds. *Mixed Integer Nonlinear Programming.* Vol. 154. The IMA Volumes in Mathematics and its Applications. Selected papers based on the IMA Hot Topics Workshop “Mixed-Integer Nonlinear Optimization: Algorithmic Advances and Applications” held in Minneapolis, MN, November 17–21, 2008. Springer, New York, 2012, pp. xvii+690. DOI: 10.1007/978-1-4614-1927-3.

[27] Jon Lee and Dan Wilson. “Polyhedral methods for piecewise-linear functions I: the lambda method”. *Discrete Appl. Math.* 108.3 (2001), pp. 269–285. DOI: 10.1016/S0166-218X(00)00216-X.

[28] Moritz Link and Stefan Volkwein. “Adaptive piecewise linear relaxations for enclosure computations for nonconvex multiobjective mixed-integer quadratically constrained programs”. *J. Global Optim.* 87 (2023), pp. 97–132. DOI: 10.1007/s10898-023-01309-5.

[29] Costas D. Maranas and Christodoulos A. Floudas. “Global optimization in generalized geometric programming”. *Comput. Chem. Eng.* 21.4 (1997), pp. 351–369. DOI: 10.1016/S0098-1354(96)00282-7.

[30] Alexander Martin, Markus Möller, and Susanne Moritz. “Mixed integer models for the stationary case of gas network optimization”. *Math. Program.* 105.2-3 (2006), pp. 563–582. DOI: 10.1007/s10107-005-0665-5.

[31] Harsha Nagarajan, Mowen Lu, Site Wang, Russell Bent, and Kaarthik Sundar. "An adaptive, multivariate partitioning algorithm for global optimization of non-convex programs". *J. Global Optim.* 74.4 (2019), pp. 639–675. DOI: 10.1007/s10898-018-00734-1.

[32] Luıs Paquete, Britta Schulze, Michael Stiglmayr, and Ana C. Lourenço. "Computing representations using hypervolume scalarizations". *Comput. Oper. Res.* 137 (Jan. 2022). 105349. DOI: 10.1016/j.cor.2021.105349.

[33] Anna Pascoletti and Paolo Serafini. "Scalarizing vector optimization problems". *J. Optimiz. Theory App.* 42.4 (1984), pp. 499–524. DOI: 10.1007/BF00934564.

[34] Tyler Perini, Natashia Boland, Diego Pecin, and Martin Savelsbergh. "A criterion space method for biobjective mixed integer programming: the boxed line method". *INFORMS J. Comput.* 32.1 (2020), pp. 16–39. DOI: 10.1287/ijoc.2019.0887.

[35] Yash Puranik and Nikolaos V. Sahinidis. "Domain reduction techniques for global NLP and MINLP optimization". *Constraints* 22.3 (2017), pp. 338–376. DOI: 10.1007/s10601-016-9267-5.

[36] Ignacio Quesada and Ignacio E. Grossmann. "A global optimization algorithm for linear fractional and bilinear programs". *J. Global Optim.* 6.1 (1995), pp. 39–76. DOI: 10.1007/BF01106605.

[37] Ignacio Quesada and Ignacio E. Grossmann. "Global optimization algorithm for heat exchanger networks". *Ind. Eng. Chem. Res.* 32.3 (1993), pp. 487–499. DOI: 10.1021/ie00015a012.

[38] Edward M.B. Smith and Constantinos C. Pantelides. "A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs". *Comput. Chem. Eng.* 23.4 (1999), pp. 457–478. DOI: 10.1016/S0098-1354(98)00286-5.

[39] Juan P. Vielma. "Mixed integer linear programming formulation techniques". *SIAM Rev.* 57 (2015), pp. 3–57. DOI: 10.1137/130915303.

[40] Kaifeng Yang, Michael Emmerich, André Deutz, and Thomas Bäck. "Efficient computation of expected hypervolume improvement using box decomposition algorithms". *J. Global Optim.* 75 (2019), pp. 3–34. DOI: 10.1007/s10898-019-00798-7.

[41] Juan M. Zamora and Ignacio E. Grossmann. "A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms". *J. Global Optim.* 14.3 (1999), pp. 217–249. DOI: 10.1023/A:1008312714792.

k	l	feasibility NLP									global NLP								
		UN			AD			AD-BT			UN			AD			AD-BT		
		CPU	RP	APSB	CPU	RP	APSB	CPU	RP	APSB	CPU	RP	APSB	CPU	RP	APSB	CPU	RP	APSB
2	2	18.1	110	20.8	9.4	113	7.3	12.3	88+132	5.75	8.8	75	18.5	6.1	77	7.0	11.9	68+154	5.3
2	4	116.1	143	169.5	37.6	203	35.6	50.7	156+230	24.0	82.7	104	159.3	22.5	134	31.5	45.0	120+246	22.9
2	6	$\star$	18	7397.3	499.8	595	220.6	453.7	538+109	179.3	$\star$	18	7397.3	454.3	533	213.6	472.9	514+131	180.5
4	2	204.1	173	49.4	41.2	317	12.9	13.1	82+168	5.1	54.7	119	26.7	21.6	185	10.2	13.9	68+196	4.8
4	4	493.7	266	292.8	82.9	379	50.2	71.7	218+297	35.2	133.7	161	255.3	57.8	276	42.9	59.1	156+308	30.8
4	6	$\star$	18	7510.3	670.4	792	231.8	606.2	572+402	173.8	$\star$	18	7510.3	492.2	628	214.8	574.4	536+352	176.1
6	2	$\star$	125	838.5	91.0	487	24.0	15.5	66+226	4.8	538.7	155	111.4	58.2	344	15.25	16.4	68+238	4.8
6	4	$\star$	84	33024	206.8	707	61.7	75.7	167+393	28.7	472.2	209	298.9	128.9	504	51.2	72.3	193+329	34.4
6	6	$\star$	18	7510.3	990.3	1125	272.7	779.3	568+495	203.8	$\star$	18	7510.3	699.9	830	250.3	877.7	572+535	203.7
8	2	$\star$	168	405.4	419.2	1445	46.1	17.7	95+240	5.3	$\star$	162	610.4	121.7	563	22.1	18.3	68+280	4.8
8	4	$\star$	117	873.4	800.1	1944	87.2	339.0	650+417	91.6	$\star$	237	512.2	225.7	756	58.4	96.1	243+413	38.7
8	6	$\star$	18	7510.3	2222.9	2033	304.5	693.1	638+461	170.7	$\star$	18	7510.3	923.0	1017	264.3	661.1	550+535	185.1

Table 2: Computational times in s, MILP count and average number of preimage set boxes for different configurations of (P2). $\star$ stands for an exit of the method because of the time limit.