Computing an enclosure for multiobjective mixed-integer nonconvex optimization problems using piecewise linear relaxations

Moritz Link* and Stefan Volkwein*

July 11, 2022

Abstract

In this paper, a new method for computing an enclosure of the nondominated set of multiobjective mixed-integer problems without any convexity requirements is presented. In fact, our criterion space method makes use of piecewise linear relaxations in order to bypass the nonconvexity of the original problem. The method chooses adaptively which level of relaxation is needed in which parts of the image space. Furthermore, it is guaranteed that after finitely many iterations an enclosure of the nondominated set of prescribed quality is returned. We demonstrate the advantages of this approach by applying it to multiobjective energy supply network problems.

Keywords. Mixed-integer nonlinear programming, multiobjective optimization, box enclosure, adaptive piecewise linear relaxation, energy supply networks.

1 Introduction

Multiobjective optimization problems arise if there is more than one objective of interest and the objectives are in conflict with each other, i.e. in general it is not possible to obtain a solution which is optimal w.r.t. each of the objective functions simultaneously. Therefore, the nondominated set (cf. Definition 2.1) consists of so-called *optimal compromises*. To have an overview of (at least some of) these optimal compromises is an advantage as one can observe the interplay between the objective functions in a more direct and easy way. Especially in real-world applications it occurs rather rarely that

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

only one objective is of interest. This is also the case in our application, where we are looking for energy supply network plans which are optimal w.r.t. the occurring costs as well as its accompanying CO_2-emissions. Naturally, a network plan being low in emissions is more costly and vice versa. Thus, having a list of optimal compromises between these two extreme solutions helps the political decision makers to get a broader picture of the situation.

In most cases, solving a multiobjective optimization problem (MOP) numerically means that one computes either an enclosure (cf. [16, 17, 18, 19]) or an approximation (cf. [2, 6, 13, 22, 25]) of the nondominated set. This is due to the fact that the nondominated set is in general infinite. In the linear case it is also possible to compute the entire nondominated set (cf., e.g., [34]). In this paper we introduce a deterministic approach to compute an enclosure of the nondominated set with a prescribed quality $\varepsilon > 0$. Although the focus of our algorithm is to compute such an enclosure, it returns also an approximation of the nondominated set, namely a finite set of ε-nondominated points of (MOP) (see Definition 2.3).

In this work we present two new methods to compute such enclosures. The first method (see Algorithm 3) tackles the problem more directly without being too complicated. In fact, it is a slight modification of the method presented in [18] in the way that we use the Restricted Weighted Sum Method (cf. [23, 24]) to obtain nondominated points of the problem. While being very effective for problems where single-objective subproblems can be solved very quickly, it is not applicable (in terms of efficiency) to more complex problems – which can happen even with a comparably small number of variables in the nonconvex mixed-integer setting. Furthermore, this approach depends heavily on the availability of solvers for the resulting single-objective problems. These drawbacks are addressed with the second method, where we combine the direct approach with piecewise linear relaxations to bypass the nonlinearities of the original problems – and therefore only single-objective mixed-integer linear problems have to be solved, where powerful solvers are available (cf. [10, 21]). These relaxations are chosen adaptively by the method itself during the solution process. Furthermore, it is possible to use different levels of relaxations in different parts of the criterion space, which can be of importance if also the relaxed problems are difficult solve.

This approach is the first of its kind. In [19] a method is presented to solve multiobjective convex mixed-integer optimization problems via relaxations. It makes use of cuts in the criterion space in order to discard infeasible integer assignments as well as to separate parts which do not belong to the image of the feasible set. In [17] the authors present the first deterministic method being able to deal with multiobjective mixed-integer nonconvex problems. However, the method presented there is based on a branch-and-bound scheme in the decision space which is a fundamental difference to the methods presented in this paper.

The manuscript is organized as follows: After introducing preliminary notions in Section 2 we define enclosures and related notions as well as provide a basic approach for computing such an enclosure in Section 3. In Section 4 we introduce and relate piecewise linear relaxations of (MOP) to so-called lower bound sets of the nondominated set, which results in a new procedure for computing an enclosure using adaptively chosen relaxations presented in Section 5. We further prove correct and finite termination

of this procedure. In Section 6 we demonstrate the advantages of using this novel approach by applying it to different instances of an energy supply network optimization problem, which are in fact multiobjective mixed-integer nonconvex problems. Finally, a conlusion is drawn in Section 7.

2 Notations and Definitions

In the present paper we write $x \leq y$ for some $x, y \in \mathbb{R}^k$ if for any $i \in [k]$ we have that $x_i \leq y_i$. In the same manner we write $x < y$ if $x_i < y_i$ for any $i \in [k]$.
The ingredients of a general multiobjective optimization problem are the continuous components $f_i \colon \mathbb{R}^{n+m} \to \mathbb{R}$, where $i \in [k] \coloneqq \{1, \ldots, k\}$, of the objective function $f = (f_1, \ldots, f_k)^\top \colon \mathbb{R}^{n+m} \to \mathbb{R}^k$, the continuous components $h_i \colon \mathbb{R}^{n+m} \to \mathbb{R}$, $i \in [p]$, of the equality constraint function $h = (h_1, \ldots, h_p)^\top \colon \mathbb{R}^{n+m} \to \mathbb{R}^p$ and the continuous components $g_i \colon \mathbb{R}^{n+m} \to \mathbb{R}$, $i \in [q]$, of the inequality constraint function $g = (g_1, \ldots, g_q)^\top \colon \mathbb{R}^{n+m} \to \mathbb{R}^q$. The corresponding multiobjective optimization problem is then given by

$$\min f(x) \quad \text{s.t.} \quad h(x) = 0,\ g(x) \leq 0,\ x \in X \coloneqq X_C \times X_I, \tag{MOP}$$

where $X_C = [\ell_C, u_C] \subsetneq \mathbb{R}^n$ and $X_I = [\ell_I, u_I] \cap \mathbb{Z}^m \subsetneq \mathbb{R}^m$ are non-empty boxes[1] with $\ell_C, u_C \in \mathbb{R}^n$, $\ell_C < u_C$, and $\ell_I, u_I \in \mathbb{Z}^m$, $\ell_I < u_I$, respectively. These boxes represent the box constraints of the continuous and integer variables, respectively. If $m = 0$, i.e. if there are no integrality constraints for any variable, we call (MOP) a *continuous* multiobjective optimization problem and if $n = 0$ we call it a *pure integer* multiobjective optimization problem. Note that the components of the occurring functions f, g and h are neither assumed to be linear nor convex.
As we may require integrality of some variables and due to the possible non-convexity of the occurring functions of (MOP) we obtain its possibly nonconvex feasible set

$$S = \{x \in X \mid e(x) = 0,\ g(x) \leq 0\},$$

which we assume to be non-empty, as well as its possibly non-convex image set

$$f(S) = \left\{f(x) \in \mathbb{R}^k \mid x \in S\right\}.$$

We define the projection of the feasible set on $\mathbb{R}^m$ by

$$S_I = \{x_I \in \mathbb{Z}^m \mid \exists\, x_C \in \mathbb{R}^n \colon\ (x_C, x_I) \in S\},$$

which we call the set of feasible integer assignments of (MOP). For a given integer assignment $\hat{x}_I \in \mathbb{Z}^m$ we define the sets

$$S_{\hat{x}_I} = \{x \in S \mid x_I = \hat{x}_I\} \text{ and } X_{\hat{x}_I} = \{x \in X \mid x_I = \hat{x}_I\},$$

[1] A closed n-dimensional box is defined as $[\ell, u] \coloneqq (\{\ell\} + \mathbb{R}^n_+) \cap (\{u\} - \mathbb{R}^n_+) = \{x \in \mathbb{R}^n \mid \ell \leq x \leq u\}$, where $\ell, u \in \mathbb{R}^n, l < u$. The open box is defined as $(\ell, u) \coloneqq (\{\ell\} + \operatorname{int}(\mathbb{R}^n_+)) \cap (\{u\} - \operatorname{int}(\mathbb{R}^n_+)) = \{x \in \mathbb{R}^n \mid \ell < x < u\}$.

describing the feasible and box-feasible solutions corresponding to $\hat{x}_I$, respectively. Notice that $S_{\hat{x}_I} = \emptyset$ holds provided $\hat{x}_I \in \mathbb{Z}^m \setminus S_I$. Since we assume all constraint functions to be continuous and by the definition of X, the feasible set S is compact. Furthermore, by the continuity of the objective functions, we know that $f(S)$ is also compact and we can therefore find a box $B = \left[z^\ell, z^u\right] \subseteq \mathbb{R}^k$ with $f(S) \subseteq \operatorname{int}(B)$ for some $z^\ell, z^u \in \mathbb{R}^k$, $z^\ell \lneq z^u$.

In general, the k objective functions of (MOP) are in conflict with each other, i.e. we cannot assume that there is a solution $\bar{x} \in S$ minimizing all objective functions simultaneously. This motivates the concepts of (non-)dominance and efficiency (cf. [14]).

Definition 2.1. 1) Let $y^1, y^2 \in \mathbb{R}^k$. Then y^1 *dominates* y^2 (and y^2 is dominated by y^1) if

$$y^1_i \leq y^2_i \text{ for all } i \in [k] \quad \text{and} \quad y^1_j < y^2_j \text{ for some } j \in [k].$$

We write $y^1 \leq y^2$, $y^1 \neq y^2$. If furthermore $y^1_i < y^2_i$ for all $i \in [k]$, then y^1 *strictly dominates* y^2 (and y^2 is strictly dominated by y^1). We then write $y^1 < y^2$.

2) Let $y \in \mathbb{R}^k$ and let $N \subset \mathbb{R}^k$. Then the vector y is *(strictly) dominated* by the set N if there exists a vector $\hat{y} \in N$ (strictly) dominating y. Similarly, if y is not (strictly) dominated by N we call y *(weakly) nondominated* by N.

3) A point $\bar{x} \in S$ is called an *efficient solution* of (MOP) if there exists no $x \in S$ such that $f(x)$ dominates $f(\bar{x})$. If there exists no $x \in S$ such that $f(x)$ strictly dominates $f(\bar{x})$, then $\bar{x}$ is called a *weakly efficient solution* of (MOP). The set of (weakly) efficient solutions of (MOP) is called its *(weakly) efficient set*.

4) A point $\bar{y} \in f(S)$ is called a *(weakly) nondominated point* of (MOP) if there exists no $y \in f(S)$ (strictly) dominating $\bar{y}$. The set of (weakly) nondominated points of (MOP) is called its *(weakly) nondominated set* and is denoted by $\mathcal{N}$ (and $\mathcal{N}_w$).

Remark 2.2. 1) Note that $\mathcal{N} = \left\{ f(x) \in \mathbb{R}^k \mid x \in S \text{ is efficient} \right\}$.

2) Note that even if we assume all objective and constraint functions of (MOP) to be convex (or linear), the nondominated set of (MOP) can be disconnected and nonconvex due to integrality constraints (e.g., c.f. [12, Figure 1]). ◊

The goal of multiobjective optimization is to determine the nondominated set $\mathcal{N}$ of a given (MOP). As the set $\mathcal{N}$ can be infinite or of a very complicated structure it can be impossible to compute it exhaustively using numerical methods. Therefore, numerical methods for multiobjective optimization mostly focus on computing a finite approximation $\mathcal{A}$ of $\mathcal{N}$ subsequent to the following three goals (cf. [36]):

- **Coverage:** all parts of the nondominated set $\mathcal{N}$ should be represented in the approximation $\mathcal{A}$.
- **Uniformity:** the points of the approximation $\mathcal{A}$ should be distributed uniformly along the nondominated set $\mathcal{N}$.

- **Cardinality:** the approximation set $\mathcal{A}$ should contain an appropriate number of points naturally depending on the number and extent of the cost functions.

Another approach of numerically solving an (MOP) is – instead of computing a sole approximation – to compute a coverage of the nondominated set $\mathcal{N}$ as presented in, e.g., [16, 17, 18, 19]. Besides the introduction of enclosures of the nondominated set, the authors present a branch-and-bound framework for computing such an enclosure of the nondominated set with a prescribed quality. As we are aiming at the same goal in this paper we present the enclosure related notions in more detail in Section 3.

However speaking of enclosures of a certain quality we come to the issue of obtaining convergence of numerical methods mostly only in the limit, i.e. after possibly infinitely many iterations. In general, this is overcome by using termination criteria together with specific tolerances – in fact, one terminates a method if it computed a solution satisfying an a-priori chosen criterion up to a specified tolerance. In single objective optimization, one way to use such tolerances is embodied in branch-and-bound methods which iteratively compute lower and upper bounds of the optimal value while checking if the gap between these two becomes smaller than the prescribed tolerance. In particular, these methods terminate with so-called ε-optimal solutions. This concept can also be applied to nondominance in multiobjective optimization.

Definition 2.3. Let $\varepsilon > 0$. A point $\bar{y} \in f(S)$ is called a *(weakly) ε-nondominated point* of (MOP) if there exists no $y \in f(S)$ such that $y + \varepsilon e$ (strictly) dominates $\bar{y}$, where $e \in \mathbb{R}^k$ denotes the all one vector. Similarly, a solution $\bar{x} \in S$ is called *(weakly) ε-efficient* for (MOP) if there is no $x \in S$ such that $f(x) + \varepsilon e$ (strictly) dominates $f(\bar{x})$. The set of ε-nondominated points is denoted by $\mathcal{N}_\varepsilon$.

Remark 2.4. Depending on the considered problem (MOP) one could also use another vector instead of e. E.g., this could be useful if the attained magnitudes of the objective functions differ largely. For an example see the explanations of the numerical examples in Section 3 and Section 6.

3 Enclosures and local bounds

In [16, 18, 19] the authors present methods for solving problems like (continuous or convex versions of) (MOP). The aim of these methods is to find an enclosure of the nondominated set $\mathcal{N}$. Following the sandwiching idea of single objective branch-and-bound methods an extension to the multiobjective setting is needed.

Definition 3.1. Let $L, U \subseteq \mathbb{R}^k$ be two finite sets satisfying

$$\mathcal{N} \subseteq L + \mathbb{R}^k_+ \text{ and } \mathcal{N} \subseteq U - \mathbb{R}^k_+.$$

Then L is called a *lower bound set*, U is called an *upper bound set*, and the set $E := E(L, U)$ defined as

$$\mathcal{N} \subseteq E(L,U) := \left(L + \mathbb{R}^k_+\right) \cap \left(U - \mathbb{R}^k_+\right) = \bigcup_{\ell \in L} \bigcup_{\substack{u \in U, \\ \ell \leq u}} [\ell, u]$$

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

For transferring the gap termination criterion to the multiobjective setting we need some sort of quality measure for the set $E(L,U)$. In [16, 17, 18, 19] the authors use the so-called width $w(E)$ which is defined as the optimal value of the problem

$$\max_{\ell,u} s(\ell,u) \quad \text{s.t.} \quad \ell \in L,\ u \in U,\ \ell \leq u, \tag{W(E)}$$

where $s(\ell,u) \coloneqq \min\{u_i - \ell_i \mid i \in [k]\}$ represents the shortest edge length of a given box $[\ell,u]$. The justification of the choice of this width measure is given by the following lemma.

Lemma 3.2 (cf. [16, Lemma 3.1])**.** *For sets $L,U \subseteq \mathbb{R}^k$ with $\mathcal{N} \subseteq L + \mathbb{R}^k_+$ and some $\varepsilon > 0$ let $w(E) < \varepsilon$. Then the relation*

$$E(L,U) \cap f(S) \subseteq \mathcal{N}_\varepsilon \tag{1}$$

holds, i.e. for any feasible point $x \in S$ with $f(x) \in E$ we know that $f(x)$ is an ε-nondominated point of (MOP)*.*

Due to Lemma 3.2 finding lower and upper bound sets L,U and thus an enclosure E with $w(E) < \varepsilon$ becomes important. In fact, computing such an enclosure is the aim of the methods from [18, 19], whereas in the branch-and-bound based method from [16] the authors use the width measure in the preimage space to declare a box sufficiently branched. As we propose a criterion space method we focus on computing an enclosure E such that $w(E) < \varepsilon$. In order to do so, we make use (as done in [17, 18, 19]) of the concept of Local Lower and Local Upper Bounds (LLBs and LUBs, respectively) which is an extension of the work in [11], [26] and also [15].
In [11] the authors call a set $Y \subseteq \mathbb{R}^k$ *stable* if there are no elements in Y dominating each other, i.e. for any two distinct $y^1, y^2 \in Y$ there are $i,j \in [k]$ such that $y^1_i < y^2_i$ and $y^2_j < y^1_j$. One can immediately see that for any (MOP) the corresponding nondominated set $\mathcal{N}$ is stable. We now start with the definition of local upper bounds as introduced in [26].

Definition 3.3. Let $N \subseteq f(S)$ be a finite and stable set. Then the *lower search region* for N is

$$s(N) \coloneqq \{y \in \operatorname{int}(B) \mid y' \not\leq y \text{ for every } y' \in N\},$$

and the *lower search zone* for some $u \in \mathbb{R}^k$ is given by

$$c(u) \coloneqq \{y \in \operatorname{int}(B) \mid y < u\}.$$

A set $U = U(N)$ is called *local upper bound set* given N if

(i) $s(N) = \bigcup_{u \in U(N)} c(u)$,

(ii) $c(u^1) \not\subseteq c(u^2)$ for any $u^1, u^2 \in U(N)$ with $u^1 \neq u^2$.

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

In [18, 19] the authors extended the concept of LUBs to the so-called Local Lower Bounds.

Definition 3.4. Let $N \subseteq \operatorname{int}(B)$ be a finite and stable set. Then the *upper search region* for N is

$$S(N) \coloneqq \{y \in \operatorname{int}(B) \mid y' \ngeq y \text{ for every } y' \in N\},$$

and the *upper search zone* for some $\ell \in \mathbb{R}^k$ is given by

$$C(\ell) \coloneqq \{y \in \operatorname{int}(B) \mid \ell < y\}.$$

A set $L = L(N)$ is called a *local lower bound* set given N if

(i) $S(N) = \bigcup_{\ell \in L(N)} C(\ell)$,

(ii) $C\left(\ell^1\right) \nsubseteq C\left(\ell^2\right)$ for any $\ell^1, \ell^2 \in L(N)$ with $\ell^1 \neq \ell^2$.

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

In [16] the authors show that given a stable set N the corresponding local upper bound set is uniquely determined. In the same manner one can show that also the corresponding local lower bound set is unique. The following result from [18] relates the concept of local lower/upper bounds to lower/upper bounds as introduced in Definition 3.1.

Lemma 3.5 (cf. [18, Corollary 3.6])**.** *Suppose that the sets $N^1 \subseteq f(S)$ and $N^2 \subseteq \operatorname{int}(B) \setminus \left(f(S) + \operatorname{int}(\mathbb{R}^k_+)\right)$ are finite and stable. Then $U(N^1)$ is an upper bound set and $L(N^2)$ is a lower bound set in the sense of Definition* 3.1.

As the local lower and upper bound sets depend on the stable set N, we have to update both if we add a new point y to the set N. These procedures come from [26, Algorithm 3], [18, Algorithm 2] and are summarized in Algorithms 1 and 2. Similar to [26] we use the following notation: for $y \in \mathbb{R}^k$, $c \in \mathbb{R}$ and $i \in [k]$ we define

$$\begin{aligned} y_{-i} &:= (y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_k)^\top \text{ and} \\ (c, y_{-i}) &:= (y_1, \ldots, y_{i-1}, c, y_{i+1}, \ldots, y_k)^\top. \end{aligned}$$

Furthermore, by analyzing the update procedure of $U(N)$ w.r.t. a point y one can relate the local upper bounds from $U(N)$ to the ones from $U(N \cup \{y\})$ in the following way: for a local upper bound $u \in U(N \cup \{y\})$ we have either $u \in U(N)$ or $u = \left(y_i, u'_{-i}\right)$ for some $i \in [k]$ and $u' \in U(N)$. For the latter case we call u' the *parent* of u. Otherwise, u is its own parent.

Lemma 3.6 ([18, Lemma 3.8])**.** *Let $u \in U(N \cup \{y\})$ be a local upper bound. Then its parent $u' \in U(N)$ is unique.*

Algorithm 1 Updating a Local Upper Bound Set: `UpdateLUB` $(U(N), y)$

Require: Local upper bound set $U(N)$ and update point $y \in f(S)$

$A = \{u \in U(N) \mid y < u\}$
for $i \in [k]$ **do**
  $B_i = \{u \in U(N) \mid y_i = u_i \text{ and } y_{-i} < u_{-i}\}$
  $P_i = \emptyset$
end for
for $i \in [k]$ **do**
  for $u \in A$ **do**
    $P_i = P_i \cup \{(y_i, u_{-i})\}$
  end for
end for
for $i \in [k]$ **do**
  $P_i = \{u \in P_i \mid u \not\leq u' \text{ for all } u' \in P_i \cup B_i,\ u' \neq u\}$
end for
$U(N \cup \{y\}) = (U(N) \setminus A) \cup \bigcup_{i \in [k]} P_i$
return Updated local upper bound set $U(N \cup \{y\})$

Algorithm 2 Updating a Local Lower Bound Set: `UpdateLLB` $(L(N), y)$

Require: Local lower bound set $L(N)$ and update point $y \in \text{int}(B)$

$A = \{\ell \in L(N) \mid y > \ell\}$
for $i \in [k]$ **do**
  $B_i = \{\ell \in L(N) \mid y_i = \ell_i \text{ and } y_{-i} > \ell_{-i}\}$
  $P_i = \emptyset$
end for
for $i \in [k]$ **do**
  for $\ell \in A$ **do**
    $P_i = P_i \cup \{(y_i, \ell_{-i})\}$
  end for
end for
for $i \in [k]$ **do**
  $P_i = \{\ell \in P_i \mid \ell \not\geq \ell' \text{ for all } \ell' \in P_i \cup B_i,\ \ell' \neq \ell\}$
end for
$L(N \cup \{y\}) = (L(N) \setminus A) \cup \bigcup_{i \in [k]} P_i$
return Updated local lower bound set $L(N \cup \{y\})$

Similarly, we define the parents of a given local lower bound ℓ. Furthermore, the analogue of Lemma 3.6 holds also for any local lower bound $\ell \in L(N \cup \{y\})$.

In [11, 26] the authors present a general scheme for computing an approximation of the nondominated set of a multiobjective optimization problem (cf., e.g., [11, Algorithm 2]). In [18] the authors extend this scheme for computing an enclosure E of $\mathcal{N}$ which, in fact, consists of boxes $[\ell, u]$, where ℓ is a local lower and u a local upper bound, and satisfies $w(E) < \varepsilon$ for given $\varepsilon > 0$. Both methods make use of the search zones determined by a given local upper bound u. We briefly describe the method from [18] as it is closely related to our next step.

Given lists of local lower bounds L and local upper bounds U, the method iterates through the elements of L in an outer loop and through the elements of U in an inner loop. Given $\hat{\ell} \in L$ it iteratively chooses $\hat{u} \in U$ satisfying $\hat{\ell} \leq \hat{u}$ and $s(\hat{\ell}, \hat{u}) > \varepsilon$. After fixing a pair $(\hat{\ell}, \hat{u})$ it solves the following optimization problem, where $\ell = \hat{\ell}$ and $u = \hat{u}$ are set,

$$\min\ t \quad \text{s.t.} \quad (t, x) \in \mathbb{R} \times S \text{ and } f(x) - \ell - t\,(u - \ell) \leq 0. \tag{PSP(ℓ, u)}$$

Problem (PSP(ℓ, u)) can be interpreted as Pascoletti-Serafini problem (cf. [33]) with reference point $\hat{\ell}$ and direction $\hat{u} - \hat{\ell}$. Subsequently, it solves an optimization problem that ensures that one obtains a nondominated point $\bar{y}$. Note that for an optimal solution $(\bar{x}, \bar{t})$ of (PSP(ℓ, u)) the point $\bar{x} \in S$ is only guaranteed to be weakly efficient (cf. [33]). If $\bar{y}$ satisfies some specific criterion (cf. [18, Section 4.3]), it is used to update the sets of local lower and upper bounds. If, otherwise, $\bar{y}$ does not satisfy this criterion, the point $\hat{\ell} + \tilde{t}(\hat{u} - \hat{\ell})$ is used to update the local lower bound set, where $\tilde{t} \geq 0.5$ is computed in a way that strict nondominance of $\hat{\ell} + \tilde{t}(\hat{u} - \hat{\ell})$ w.r.t. $\mathcal{N}$ is ensured. After running through both loops, it is guaranteed that any new box $[\ell^{\text{new}}, u^{\text{new}}]$ evolving from the above described updates satisfies

$$(u^{\text{new}} - \ell^{\text{new}})_i \leq \max\left\{\varepsilon,\ 0.5\left(u^{\text{old}} - \ell^{\text{old}}\right)_i\right\}$$

for at least one index $i \in [k]$, where ℓ^{old} and u^{old} are local lower and upper bounds from the beginning of the iteration satisfying $\ell^{\text{old}} \leq \ell^{\text{new}} \leq u^{\text{new}} \leq u^{\text{old}}$ (cf. [18, Theorem 4.2]). Using that, the authors prove correctness and finiteness of the proposed method and provide an upper bound on the number of iterations until termination (cf. [18, Lemma 4.3] and [18, Theorem 4.5]).

For this paper we use a slightly modified approach as basic scheme which is written in Algorithm 3.

In fact, we only loop through the set U_{loop}, i.e. the set of local upper bounds at the begin of the iteration. Furthermore, in Step 6 we do not solve (PSP(ℓ, u)), but the weighted-sum problem

$$\min\ \alpha^\top f(x) \quad \text{s.t.} \quad x \in S \text{ and } f(x) < u - \delta e, \tag{WSP$(\alpha; u)$}$$

for some weight vector $\alpha \in \operatorname{int}(\mathbb{R}^k_+)$. Note that if the weight vector α has only strictly positive entries we know that any global solution $\bar{x}$ of (WSP$(\alpha; u)$) is efficient for (MOP) (cf. [1, Lemma 1.5.2]), i.e. $\bar{y} = f(\bar{x}) \in \mathcal{N}$ and $\bar{y} < u - \delta e$. Furthermore, we

Algorithm 3 General scheme for computing an enclosure of the nondominated set relying on a scalarization technique.

Require: box $B = \left[z^\ell, z^u\right]$ with $f(S) \subseteq \operatorname{int}(B)$, termination tolerance $\varepsilon > 0$ and off-set factor $\delta > 0$

Initialize nondominated set $N = \emptyset$, set of local lower bounds $L = \left\{z^\ell\right\}$ and set of local upper bounds $U = \{z^u\}$
while $w(E) \geq \varepsilon$ **do**
  $U_{\text{loop}} = U$
  for $u \in U_{\text{loop}}$ **do**
    if there exists $\ell \in L$ with $\ell \leq u$ and $s(\ell, u) \geq \varepsilon$ **then**
      if there exists $y \in \mathcal{N}$ with $y < u - \delta e$ **then**
        Update U w.r.t. y using Algorithm 1
        Update L w.r.t. y using Algorithm 2
      else
        Update L w.r.t. $u - \delta e$ using Algorithm 2
      end if
    end if
  end for
end while
return Enclosure $E(L, U)$ satisfying $w(E) < \varepsilon$

know that there exists $y \in \mathcal{N}$ with $y < u - \delta e$ if and only if (WSP$(\alpha; u)$) has a solution. In sum, this means that we can decide whether to enter and execute the `if`-statement in Step 6 after solving (WSP$(\alpha; u)$) for a strictly positive weight vector α. This weight vector could be chosen always the same, e.g., $\alpha = (1, \ldots, 1)^\top / k$. But one could also compute it individually for any u, as done for the numerical examples presented later in this work. For example, one could compute the weight vector α for the problem (WSP$(\alpha; u)$) using a computation technique proposed in [35]. Given the current local upper bound of interest $u \in U(N)$, where N is the current approximation of $\mathcal{N}$, for any $i \in [k]$ let $y^i \in N$ be a defining point of the i–th component of u, i.e. for any $i \in [k]$ we have $y^i_i = u_i$ and $y^i_{-i} < u_{-i}$. We then solve the system

$$\begin{pmatrix} y^1_1 & \ldots & y^1_k \\ \vdots & & \vdots \\ y^k_1 & \ldots & y^k_k \end{pmatrix} \begin{pmatrix} \tilde{\alpha}_1 \\ \vdots \\ \tilde{\alpha}_k \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \tag{2}$$

and set $\alpha := |\tilde{\alpha}| / \|\tilde{\alpha}\|_2$. Note that α is the normal vector of the hyperplane determined by the points $y^1, \ldots, y^k$. Using (WSP$(\alpha; u)$), Algorithm 3 can be seen in the context of the so-called *Adaptive Weighted Sum Method* as presented in [23, 24].

Remark 3.7. For a clear definition of defining points we refer to [26]. Furthermore, in [26] the authors present methods for updating local upper bound sets together with the sets of defining points, i.e. an algorithm doing the same as Algorithm 1, but also updating the corresponding defining points (see [26, Algorithm 5]). In fact, [26,

Algorithm 5] is used in any implementation of the present paper instead of Algorithm 1. We waived to present [26, Algorithm 5] in detail to keep things more concise.

An exemplary situation during Algorithm 3 is depicted in Figure 1, where $\hat{u} \in U_{\text{loop}}$ denotes the current local upper bound of interest. In that picture one can see the impact of enlarging δ, as the distance between the new potentially nondominated point and the old ones increases if δ gets closer to ε.

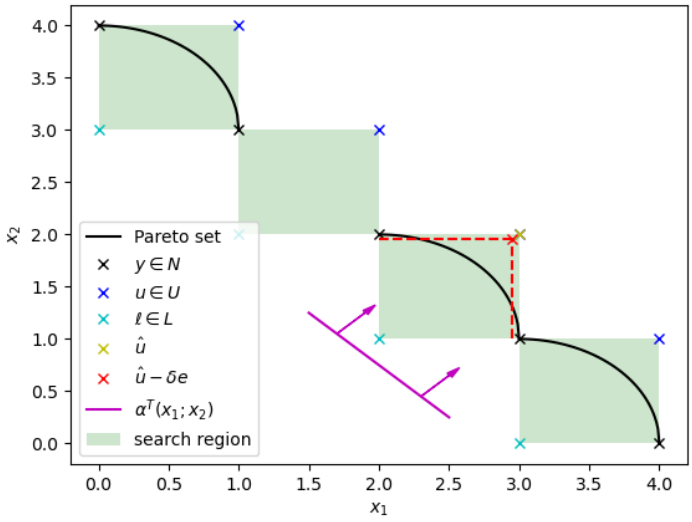

Figure 1: Exemplary configuration during Algorithm 3 tackling an instance of Example 3.13.

Let us now proceed with proving correctness and finiteness of Algorithm 3.

Lemma 3.8. *Let U be the local upper bound set at some arbitrary point in Algorithm* 3. *Then U is an upper bound set in the sense of Definition* 3.1.

Proof. We initialize the set $U = \{z^u\}$. During Algorithm 3 the set U is updated only in Step 7. The update procedure takes place w.r.t. a point $y \in \mathcal{N}$ and therefore particularly $y \in f(S)$. Thus, by correctness of Algorithm 1 we obtain that U is a local upper bound set w.r.t. some set $N_1 \subseteq f(S)$. The claim follows by Lemma 3.5. □

Lemma 3.9. *Let L be the local lower bound set at some arbitrary point in Algorithm* 3. *Then L is a lower bound set in the sense of Definition* 3.1.

Proof. We initialize the set $L = \{z^\ell\}$. During Algorithm 3 the set L is updated only in the Steps 8 and 10. If updated in Step 8, L is updated w.r.t. a point $y \in \mathcal{N}$ and therefore in particular $y \notin f(S) + \text{int}(\mathbb{R}^k_+)$. If otherwise updated in Step 10, it is

updated w.r.t. $y = \hat{u} - \delta e$ for some local upper bound $\hat{u} \in U$. Since we only enter Step 10, if there exists no $\bar{y} \in \mathcal{N}$ satisfying $\bar{y} \leq \hat{u} - \delta e$ we particularly know that $y = \hat{u} - \delta e \notin f(S) + \operatorname{int}(\mathbb{R}^k_+)$. Hence, by correctness of Algorithm 2 we obtain that L is a local lower bound set w.r.t. some set $N_2 \subseteq \operatorname{int}(B) \setminus (f(S) + \operatorname{int}(\mathbb{R}^k_+))$. The claim follows by Lemma 3.5. □

The above results show that the sets L and U are lower and upper bound sets in the sense of Definition 3.1 at any time of Algorithm 3. To show that also the output sets are sets of that type, we have to ensure that the sets $N_1 \subseteq f(S)$ and $N_2 \subseteq \operatorname{int}(B) \setminus (f(S) + \operatorname{int}(\mathbb{R}^k_+))$ are finite. If we show that Algorithm 3 is itself finite, i.e. terminates after a finite number of steps, the finiteness of N_1 and N_2 follows immediately. In order to show finiteness of Algorithm 3 we start with some result about the decrease of certain edge lengths during one iteration of the method. The following is essentially the same as the corresponding results in [18]. However, as some adaptions have to be made, we present the adapted proofs in detail.

Theorem 3.10. *Let $\varepsilon > \delta > 0$ and $z^\ell, z^u \in \mathbb{R}^k$ be the input parameters of Algorithm 3. Moreover, let L^{start} and U^{start} be the local lower and upper bound sets at the beginning of some iteration in Algorithm 3, i.e. at the begin of the `while`-loop of some iteration. Accordingly denote by L^{end}, U^{end} the sets at the end of this iteration. Then for any $\ell^e \in L^{end}$ and any $u^e \in U^{end}$ with $\ell^e \leq u^e$ there exist $\ell^s \in L^{start}$ and $u^s \in U^{start}$ such that the following hold:*

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

2) *There exists an index $j \in [k]$ such that*

$$(u^e - \ell^e)_j < \max\left\{(u^s - \ell^s)_j - \delta, \varepsilon\right\}.$$

Proof. Let $\ell^e \in L^{\text{end}}$ and $u^e \in U^{\text{end}}$ such that $\ell^e \leq u^e$. We denote by $P(\ell^e), P(u^e) \subseteq \mathbb{R}^k$ the sets containing the parent history of ℓ^e and u^e in the current iteration, i.e. their parents, their parents' parents and so on until the ancestors of ℓ^e and u^e belonging to L^{start} and U^{start}. By Lemma 3.6 we have that

$$|P(\ell^e) \cap L| = 1 \text{ and } |P(u^e) \cap U| = 1 \tag{3}$$

at any point in the current iteration, i.e. for any assignment of L and U during the `while`-loop, and therefore in particular we obtain $\ell^s \in P(\ell^e) \cap L^{\text{start}}$ and $u^s \in P(u^e) \cap U^{\text{start}}$. By the definition of parents and the corresponding update procedures given in Algorithm 2 and Algorithm 1 we know that $\ell^p \leq \ell^c$ and $u^c \leq u^p$, where ℓ^c and u^c are the children of ℓ^p and u^p, respectively. Hence, we obtain that

$$\begin{aligned} \ell^s \leq \ell \leq \ell^e \text{ for all } \ell \in P(\ell^e), \\ \text{and } u^e \leq u \leq u^s \text{ for all } u \in P(u^e), \end{aligned} \tag{4}$$

and therefore 1) is satisfied.
We now want to show that also 2) holds. We assume for a contradiction that there

exist $\ell^e \in L^{\text{end}}$ and $u^e \in U^{\text{end}}$ with $\ell^e \leq u^e$ such that for any $\ell \in L^{\text{start}}$ and $u \in U^{\text{start}}$ with $\ell \leq \ell^e \leq u^e \leq u$ and any index $i \in [k]$ we have that

$$(u^e - \ell^e)_i \geq \max\{(u - \ell)_i - \delta, \varepsilon\}. \tag{5}$$

In particular, (5) holds for the ancestors of ℓ^e and u^e in L^{start} and U^{start}, namely ℓ^s and u^s. We consider now the point in Algorithm 3, where u^s is chosen in the `for`-loop. Note that u^s might not be the first local upper bound from U_{loop} considered in the `for`-loop and therefore it might be the case that $u^s \notin U_{\text{current}}$, where U_{current} is the current assignment of U. However, since for any $i \in [k]$ we have that $(u^e - \ell^e)_i \geq \varepsilon$ we know that for any $\ell' \in P(\ell^e) \cap L_{\text{current}}$ we have that $(u^s - \ell')_i \geq \varepsilon$ for any $i \in [k]$, where L_{current} denotes the current assignment of L. Hence, we have that

$$s(\ell', u^s) \geq \varepsilon \tag{6}$$

and we therefore enter the `if`-statement in Step 5. Similarly, we fix $u' \in P(u^e) \cap U_{\text{current}}$. Note that

$$\ell^s \leq \ell' \leq \ell^e \text{ and } u^e \leq u' \leq u^s.$$

We denote by L_{updated} and U_{updated} the assignments of L and U after the current iteration of the `for`-loop is executed. We have to distinguish two main cases, namely we enter the `if`-statement in Step 6 (case **A**) or we do not (case **B**).

(A) In that case we enter the `if`-statement in Step 6, i.e. there exists $y \in \mathcal{N}$ with $y < u^s - \delta e$. We have to distinguish two cases.
Case A.1: $y < u'$. Then u' would be removed during the update of U_{current} using Algorithm 1, i.e. $u' \notin U_{\text{updated}}$. Now, we have the candidates

$$u^i = \left(y_i, u'_{-i}\right), \quad \text{for } i \in [k],$$

from which at least one belongs to U_{updated} by (3). Say $u^j \in U_{\text{updated}}$. Using (4), we compute

$$(u^e - \ell^e)_j \leq \left(u^j - \ell^e\right)_j \leq y_j - \ell^s_j < (u^s - \ell^s)_j - \delta,$$

a contradiction to (5).
Case A.2: $y \not< u'$. Then there exists $j \in [k]$ with $u'_j \leq y_j$. Again, using (4), we compute

$$(u^e - \ell^e)_j \leq (u' - \ell^e)_j \leq y_j - \ell^s_j < (u^s - \ell^s)_j - \delta,$$

a contradiction to (5).

(B) Assume now that we do not enter the `if`-statement in Step 6, i.e. there exists no $y' \in \mathcal{N}$ with $y' < u^s - \delta e =: y$. We have to distinguish two cases.
Case B.1: $\ell' < y$. Then ℓ' would be removed during the update of L_{current} using Algorithm 2, i.e. $\ell' \notin L_{\text{updated}}$. Now, we have the candidates

$$\ell^i = \left(y_i, \ell'_{-i}\right), \quad \text{for } i \in [k],$$

from which at least one belongs to L_{updated} by (3). Say $\ell^j \in L_{\text{updated}}$. Using (4), we compute

$$(u^e - \ell^e)_j \leq \left(u^e - \ell^j\right)_j \leq u_j^s - y_j = \delta < \varepsilon,$$

a contradiction to (5).

Case B.2: $\ell' \not< y$. Then there exists $j \in [k]$ with $\ell'_j \geq y_j = u_j^s - \delta$, i.e. particularly $s(\ell', u^s) \leq \delta < \varepsilon$, a contradiction to (6).

Obtaining a contradiction in all possible cases shows that our assumption (5) cannot be true and therefore statement 2) is also true, which completes the proof. □

Knowing that in every iteration of Algorithm 3 for any pair (ℓ^e, u^e) we obtain a decrease w.r.t. the edge length for at least one edge $j \in [k]$ we are able to prove that Algorithm 3 terminates after finitely many iterations.

Theorem 3.11. *Let $\varepsilon > \delta > 0$ and $z^\ell, z^u \in \mathbb{R}^k$ be the input parameters of Algorithm 3. We define*

$$\Delta \coloneqq \left\| z^u - z^\ell \right\|_\infty \quad \text{and} \quad \kappa \coloneqq k \left\lceil \frac{\Delta - \varepsilon}{\delta} \right\rceil + 1.$$

Then the number of iterations of Algorithm 3, i.e. the number of iterations of the `while`*-loop, is bounded by* $\max\{1, \kappa\}$*. Hence, Algorithm 3 is finite.*

Proof. If $\kappa \leq 1$, then $\Delta \leq \varepsilon$ and Algorithm 3 terminates without entering the `while`-loop.

Therefore, let $\kappa > 1$. We write $\ell^1 = z^\ell$ and $u^1 = z^u$ and for any $l \geq 1$ let L_l and U_l be the assignments of L and U at the begin of the l-th execution of the `while`-loop.

By Theorem 3.10, we know that for any $l \geq 2$ and any $\ell^l \in L_l$ and $u^l \in U_l$ there exist $\ell^{l-1} \in L_{l-1}$ and $u^{l-1} \in U_{l-1}$ with $\ell^{l-1} \leq \ell^l \leq u^l \leq u^{l-1}$ as well as an index $i^l \in [k]$ such that

$$\left(u^l - \ell^l\right)_{i^l} < \max\left\{\left(u^{l-1} - \ell^{l-1}\right)_{i^l} - \delta, \varepsilon\right\}. \tag{7}$$

Suppose now that Algorithm 3 has more than κ iterations. Then there exist $\ell^\kappa \in L_\kappa$ and $u^\kappa \in U_\kappa$ with $s(\ell^\kappa, u^\kappa) \geq \varepsilon$. For any $i \in [k]$ we define

$$n(i) = \left|\left\{l \in \{2, \ldots, \kappa\} \mid i = i^l\right\}\right|.$$

As $\kappa > k\left\lceil \frac{\Delta - \varepsilon}{\delta} \right\rceil$ there exists at least one index $j \in [k]$ with $n(j) \geq \left\lceil \frac{\Delta - \varepsilon}{\delta} \right\rceil + 1$. Iterative use of (7) yields that

$$\begin{aligned} s(\ell^\kappa, u^\kappa) &\leq u_j^\kappa - \ell_j^\kappa \\ &< u_j^1 - \ell_j^1 - n(j)\delta \\ &\leq z_j^u - z_j^\ell - \left\lceil \frac{\Delta - \varepsilon}{\delta} \right\rceil \delta \\ &\leq \Delta - \frac{\Delta - \varepsilon}{\delta}\delta \leq \varepsilon, \end{aligned}$$

a contradiction to $s(\ell^\kappa, u^\kappa) \geq \varepsilon$. Hence, we have that $w(E_\kappa) < \varepsilon$ and Algorithm 3 terminates. □

Corollary 3.12. *Let $\varepsilon > \delta > 0$ and $z^\ell, z^u \in \mathbb{R}^k$ be the input parameters of Algorithm 3. Then after finitely many iterations of the `while`-loop an enclosure E of the nondominated set $\mathcal{N}$ satisfying $w(E) < \varepsilon$ is returned.*

Proof. Theorem 3.11 tells us that Algorithm 3 terminates after at most κ calls of the `while`-loop, i.e. the output set E_κ satisfies $w(E_\kappa) < \varepsilon$. By Lemma 3.8 and Lemma 3.9 we know that the set L, resp. U, is a lower, resp. upper, bound set in the sense of Definition 3.1. In particular, this holds for L_κ and U_κ. Thus, E_κ is an enclosure of the nondominated set $\mathcal{N}$ satisfying $w(E_\kappa) < \varepsilon$. □

We conclude this section with an example together with the numerical results. Note that for the numerical realization we did not exactly implement Algorithm 3, but a modification in a way that we do not iterate through all local upper bounds in every call of the `while`-loop. Instead, at the begin of any `while`-loop we determine a local upper bound $\hat{u} \in U$ such that there exists $\ell \in L$ with $\ell \leq \hat{u}$ and $s(\ell, \hat{u}) = w(E)$ and then perform the loop for $\hat{u}$ only. By doing so, we avoid to perform computations for local upper bounds $u \in U_{\text{loop}}$ which do not belong to the current U anymore. However, finiteness of this procedure is not guaranteed by Theorem 3.11 anymore. But if it terminates after finitely many steps, then we still have $w(E) < \varepsilon$. Furthermore, we use a normalized shortest edge calculation, i.e. instead of $s(\ell, u) = \min_{i \in [k]} \{u_i - \ell_i\}$ we compute it via:

$$s(\ell, u) = \min_{i \in [k]} \left\{ \frac{u_i - \ell_i}{z_i^u - z_i^\ell} \right\}.$$

Consequently, the termination tolerance ε is not an absolute measure anymore, but a relative one. This normalized approach guarantees that differences in the magnitude of the different objective functions are taken into account.

Example 3.13. We consider the optimization problem:

$$\begin{aligned} \min x = (x_1, \ldots, x_n)^\top \quad \text{s.t.} \quad & 0 = b_i \left(\|x - m^i\|_2^2 - r^2 \right), && i \in [m] \\ & 0 \leq b_i \left(x_j - m_j^i \right), && i \in [m],\ j \in [n], \\ & 1 = \sum_{i=1}^{m} b_i, \\ & b_i \in \{0, 1\}, && i \in [m], \\ & 0 \leq x. \end{aligned} \tag{Circles}$$

We consider two different instances of (Circles):

- The first one is determined by $n = 2$, $r = 1$, $m = 3$ as well as $m^1 = (3, 0)^\top$, $m^2 = (2, 1)^\top$ and $m^3 = (0, 3)^\top$. The result of Algorithm 3 with relative tolerance

$\varepsilon = 0.0125$ (this is the equivalent of an absolute tolerance of $\varepsilon = 0.05$) and off-set factor $\delta = 0.95\varepsilon$ is depicted in Figure 2. One can observe that the gap in the nondominated set is realized in the enclosure via the edge with corner at $(2, 2)^\top$. The corresponding local upper bound is not dominated by any point in the nondominated set and therefore the method updates the local lower bounds with the point $u - \delta e$.

- The second instance is determined by $n = 3$, $r = 1$, $m = 3$ as well as $m^1 = (1, 0, 0)^\top$, $m^2 = (0, 1, 0)^\top$ and $m^3 = (0, 0, 1)^\top$. The result of Algorithm 3 with relative tolerance $\varepsilon = 0.05$ (this is the equivalent of the absolute tolerance of $\varepsilon = 0.1$) and off-set factor $\delta = 0.95\varepsilon$ is depicted in Figure 3. Again, similar to the two dimensional case, we have a region in the image space with local upper bounds not dominated by any nondominated point of the problem. We observe the same behavior.

All numerical computations in this paper are realized using the `Python`-package (cf. [29]) of the SCIP optimization suite (cf. [4]) and are carried out on a machine with Intel Core i7-8565U processor and 32GB of RAM.

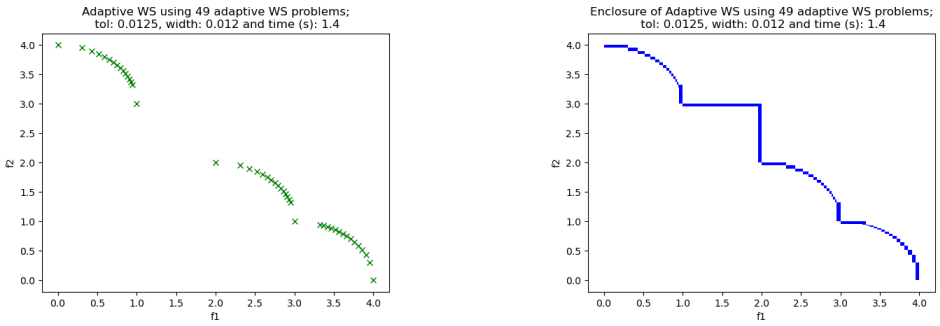

Figure 2: Computational results on first instance in Example 3.13. **Left:** nondominated points of (Circles) obtained by Algorithm 3 with WSM. **Right:** enclosure given by the LLBs and LUBs.

4 Piecewise linear relaxations and lower bounding

As we have seen, using Algorithm 3 we are able to compute an enclosure of the nondominated set of a given (MOP). However, as all the weighted-sum problems that have to be solved during Algorithm 3 are still MINLPs in general, there may arise problems while tackling, e.g., larger instances (see Section 6). One idea to cope with this issue is to bypass the non-linearity of the problems, e.g., trying to solve just MILPs.
We have seen before that the idea of computing enclosures instead of a finite approximation of the nondominated set is somehow lifted from the single objective setting to the multiobjective one. In the same spirit we are going to use an idea coming

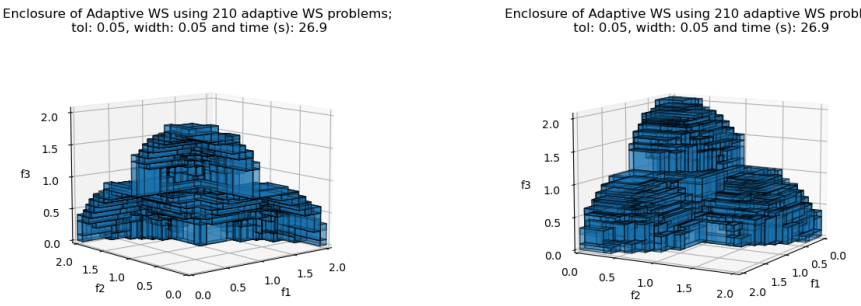

Figure 3: Computational results on second instance in Example 3.13. Two different viewpoints of the enclosure computed by Algorithm 3 with WSM.

from single objective optimization in regard of solving MINLPs. In [9, 28, 32] several methods for solving MINLPs, e.g., a (MOP) with $k = 1$, by iteratively solving adaptively refined convex or linear MIP relaxations of the original problem are presented. Note that there is plenty of literature and ongoing research in the area of efficiently computing linear or convex relaxations (or approximations) of nonlinear problems, cf., e.g., [3, 8, 9, 20, 27, 31, 38]. Although the basic idea – but not the realization – of consequently refining the relaxations and therefore guaranteeing convergence of the sequence of optimal values of the relaxed problems to the optimal value of the original problem is the same, the respective termination criteria of the algorithms from [32] on the one hand and [9, 28] on the other hand differ. In the latter, both algorithms terminate if the maximal possible constraint violation of the relaxed problem in comparison to the original problem is below an a-priori given, but arbitrarily small error bound. Morally speaking, we say that the original MINLP is solved to optimality if we computed an optimal solution of a relaxation violating the nonlinear constraints only in an acceptable manner. The authors prove that under certain assumptions on the chosen relaxation technique the proposed methods terminate after a finite number of steps.
In [32] the authors terminate their algorithm by the gap criterion which is known from classical branch-and-bound methods. The algorithm uses parts of the optimal solutions of the relaxed problems to fix variable values in the original problem, e.g., one can fix all integer variables in the MINLP to take the corresponding values of the relaxed solution and obtain an NLP problem which – if it is feasible – might be easier to solve and then gives an upper bound on the optimal value. In case of fixing the integer variables of the relaxed solution $(\tilde{x}_C, \tilde{x}_I)$ in the MINLP the resulting NLP has the following form

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

Furthermore, as before, the lower bound on the optimal value is consequently im-

proved by refining the relaxations. In every step the smallest available upper bound and the best lower bound is used to sandwich the optimal value of the original MINLP and terminate the algorithm if the gap between these bounds is small enough. However, it is not obvious that the resulting NLP problems become feasible any time during the procedure and therefore there is no guarantee that any upper bound is available for the gap criterion at all. Nevertheless, in practice it is rather unlikely to produce exclusively infeasible integer assignments with the solution of the relaxed problems, in particular with increasing accuracy.
In this paper we want to merge both ideas as the use of upper bounds together with the gap criterion may cause termination even if the maximal possible constraint violation of the relaxation is not below the tolerance and therefore accelerate the computations. In fact, we use the scheme presented in [9] to preserve theoretically ensured convergence of the method, but also add the upper bounding part as well as the gap termination criterion from [32] to avoid unnecessary computations. Furthermore, the relaxation techniques used in [9] are able to handle general nonlinear functions, whereas the techniques in [32] are based on the so-called McCormick relaxations for bilinear and multilinear terms (cf. [30]) and are therefore only able to handle general polynomial terms. For this paper we restrict ourselves to the explicit handling of quadratic and bilinear terms as these are the only ones appearing in our application (see Section 6). However, the method presented in the remainder of this paper is capable of handling general nonlinearities if an adequate relaxation technique is available. We start with introducing the necessary notions.

Definition 4.1. Let (MOP) be given. Further let $\tilde{S} \subseteq \mathbb{R}^{n+m}$ be a set and $\tilde{f}\colon \mathbb{R}^{n+m} \to \mathbb{R}^k$ a k-vector-valued function such that $S \subseteq \tilde{S}$ and $\tilde{f}(x) \leq f(x)$ for any $x \in S$. Then we call

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

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

Note that, if $k = 1$, we have that the optimal value $\tilde{f}(\tilde{x}^*)$ of (RMOP$_{\tilde{S}}$) is a lower bound on the optimal value $f(x^*)$ of (MOP). For brevity we assume without loss of generality that the objective f in (MOP) is linear. This is also motivated by our energy supply network considered in Section 6. Consequently, f needs no further relaxation and for the remainder of that paper a relaxation (RMOP$_{\tilde{S}}$) is characterized by its feasible set $\tilde{S}$. Given two different relaxations $\tilde{S}$ and $\hat{S}$, we call the relaxation $\tilde{S}$ *finer* than $\hat{S}$ if $\tilde{S} \subseteq \hat{S}$. Again, if $k = 1$, for two relaxations $\tilde{S} \subseteq \hat{S}$ we have that the optimal value $f(\hat{x}^*)$ of the relaxation $\hat{S}$ is a lower bound on the optimal value $f(\tilde{x}^*)$ of the relaxation $\tilde{S}$. There is a plenty of possibilities to obtain relaxations of a given (MOP). One way is to use the concept of piecewise linear under- and overestimators (cf., e.g., [7, 9, 20]).

Definition 4.2. Let $h(x_1, \ldots, x_n)$ be a nonlinear real-valued function with compact domain $D_h \subset \mathbb{R}^n$ appearing in (MOP), e.g., as a constraint function.

- We call a continuous piecewise linear function $h_u \colon D_h \to \mathbb{R}$ a *piecewise linear underestimator* of h if $h_u(x) \leq h(x)$ for all $x \in D_h$.
- We call a continuous piecewise linear function $h_o \colon D_h \to \mathbb{R}$ a *piecewise linear overestimator* of h if $h(x) \leq h_o(x)$ for all $x \in D_h$.
- We call a tuple $\mathcal{R}_h = (h_u, h_o)$, where h_u is a piecewise linear underestimator and h_o a piecewise linear overestimator of h, a *piecewise linear relaxation* of h.

Given a piecewise linear relaxation $\mathcal{R}_h$ of a term h, we obtain a relaxation in the sense of Definition 4.1 by replacing any appearance of h by an additional variable $\hat{h}$ and adding the constraints $h_u(x) \leq \hat{h}$ and $\hat{h} \leq h_o(x)$. Indeed, this yields a relaxation in the sense of Definition 4.1 by the fact that

$$\big\{(h(x), x) \in \mathbb{R}^{1+n} \mid x \in D_h\big\} \subseteq \big\{ \left(\hat{h}, x\right) \in \mathbb{R}^{1+n} \mid h_u(x) \leq \hat{h} \leq h_o(x) \text{ for all } x \in D_h\big\}.$$

Thus, given (MOP) with nonlinear terms $h_i, i \in \mathcal{I}$, for some finite index set $\mathcal{I}$ together with piecewise linear relaxations $\mathcal{R}_{h_i}, i \in \mathcal{I}$, and proceeding as above, we obtain a relaxation of (MOP) with feasible set denoted by $\tilde{S}_{\mathcal{R}}$ or simply by $\mathcal{R}$, where $\mathcal{R}$ is the collection of all $\mathcal{R}_{h_i}$. For the remainder of that paper we only consider relaxations coming from piecewise linear relaxations of all appearing nonlinear terms of (MOP). One can see immediately that given two relaxations $\mathcal{R}$ and $\mathcal{R}'$ we have that $\mathcal{R}' \subseteq \mathcal{R}$ if and only if for any nonlinear term h appearing in (MOP) we have $\mathcal{R}'_h \subseteq \mathcal{R}_h$, i.e. $h_u(x) \leq h'_u(x)$ and $h'_o(x) \leq h_o(x)$ for all $x \in D_h$. Consequently, by improving the piecewise linear under- and overestimators we refine the corresponding relaxation. Accordingly with [7], we measure the quality of a given piecewise linear relaxation $\mathcal{R}_h$ of a nonlinear term h by

- the *overestimation error*

$$\varepsilon_o^{(h,\mathcal{R}_h)} \coloneqq \max_{x \in D_h} \; h_o(x) - h(x),$$

- the *underestimation error*

$$\varepsilon_u^{(h,\mathcal{R}_h)} \coloneqq \max_{x \in D_h} \; h(x) - h_u(x),$$

- and the *overall relaxation error* $\varepsilon_{rel}^{(h,\mathcal{R}_h)} \coloneqq \max\left\{\varepsilon_u^{(h,\mathcal{R}_h)}, \varepsilon_o^{(h,\mathcal{R}_h)}\right\}$.

Clearly, the relaxation error decreases while refining relaxations, i.e. for two relaxations $\mathcal{R}'_h \subseteq \mathcal{R}_h$ we have that $\varepsilon_{rel}^{(h,\mathcal{R}'_h)} \leq \varepsilon_{rel}^{(h,\mathcal{R}_h)}$.

Naturally, we can also extend this concept to measure the quality of a relaxation $\mathcal{R}$ of a given (MOP). In fact, the *overestimation error* of $\mathcal{R}$ is defined by

$$\varepsilon_o^{\mathcal{R}} \coloneqq \max\left\{\varepsilon_o^{\left(h_i,\mathcal{R}_{h_i}\right)} \mid h_i \text{ appears in (MOP) and } \mathcal{R}_{h_i} \in \mathcal{R} \text{ for } i \in \mathcal{I}\right\}.$$

Similar for the *underestimation error*. The *overall relaxation error* is then given by $\varepsilon_{rel}^{\mathcal{R}} \coloneqq \max\{\varepsilon_o^{\mathcal{R}}, \varepsilon_u^{\mathcal{R}}\}$. Similar as before, if $\mathcal{R}' \subseteq \mathcal{R}$ for two relaxations $\mathcal{R}$ and $\mathcal{R}'$ we have that

$$\varepsilon_{rel}^{\mathcal{R}'} \leq \varepsilon_{rel}^{\mathcal{R}}.$$

Let us now briefly introduce the relaxation technique of interest for that paper, namely the so-called McCormick piecewise linear relaxations (cf. [30]). We start with the bilinear case, i.e. $h(x,y) = xy$, with compact domain

$$D_h = \left\{(x,y) \in \mathbb{R}^2 \mid x^\ell \leq x \leq x^u, y^\ell \leq y \leq y^u\right\}.$$

The McCormick piecewise linear relaxation depends on a so-called partition (or triangulation) of the domain D_h. For our purpose it is enough to consider simple partitions of the intervals $[x^\ell, x^u]$ and $[y^\ell, y^u]$ in r equidistant intervals. Consequently, we obtain gridpoints $x_i = x^\ell + i(y^u - x^\ell)/r$ and $y_i = y^\ell + i(y^u - y^\ell)/r$ for $i \in \{0, \dots, r\}$ with corresponding intervals $[x_{i-1}, x_i]$ and $[y_{i-1}, y_i]$ for $i \in [r]$. For any $i, j \in [r]$ and any $(x,y) \in [x_{i-1}, x_i] \times [y_{j-1}, y_j]$ the linear underestimator function on $D_h^{ij} \coloneqq [x_{i-1}, x_i] \times [y_{j-1}, y_j]$ is given by

$$h_u^{ij}(x,y) \coloneqq \max\left\{x_{i-1}y + y_{j-1}x - x_{i-1}y_{j-1}, x_i y + y_j x - x_i y_j\right\},$$

and the linear overestimator is given by

$$h_o^{ij}(x,y) \coloneqq \min\left\{x_{i-1}y + y_j x - x_{i-1}y_j, x_i y + y_{j-1}x - x_i y_{j-1}\right\}.$$

The piecewise linear under- and overestimator functions h_u and h_o are then given by concatenation as shown in Figure 4.

Note that h_u and h_o are continuous piecewise linear functions and $h_u(x,y) \leq h(x,y) \leq h_o(x,y)$ for all $(x,y) \in D_h$, i.e. h_u is a piecewise linear underestimator and h_o is a piecewise linear overestimator of h in the sense of Definition 4.2. We denote the piecewise linear relaxation based on r equidistant partitions for each variable by $\mathcal{R}_{xy}^r$. The resulting relaxation error is given by

$$\varepsilon_{rel}^{(h,\mathcal{R}_h^r)} = \frac{\left(x^u - x^\ell\right)\left(y^u - y^\ell\right)}{4r^2}.$$

Although the quadratic case is just a special case of the bilinear case we give the explicit formulation. Therefore, let $h(x) = x^2$ with $D_h = \{x \in \mathbb{R} \mid x^\ell \leq x \leq x^u\}$. We consider again the partition in r equidistant intervals, i.e. $D_h^i = [x_{i-1}, x_i]$ for $i \in [r]$. For any $i \in [r]$ and any $x \in D_h^i$ the linear underestimator is given by

$$h_u^i(x) \coloneqq \max\left\{2x_{i-1}x - x_{i-1}^2, 2x_i x - x_i^2\right\},$$

and linear overestimator is given by

$$h_o^i(x) \coloneqq (x_{i-1} + x_i)\, x - x_{i-1}x_i.$$

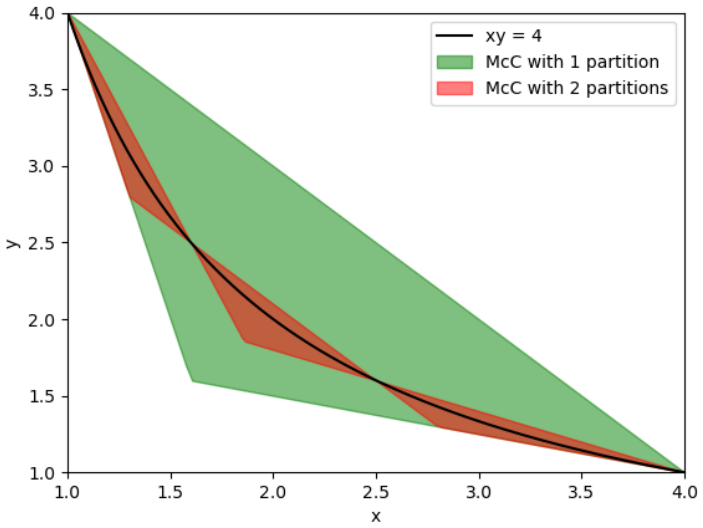

Figure 4: McCormick relaxations of the bilinear term xy using one partition per variable (green) and two partitions per variable (red).

The resulting relaxation error corresponding to r equidistant partitions is given by

$$\varepsilon_{rel}^{(h,\mathcal{R}_h^r)} = \frac{\left(x^u - x^\ell\right)^2}{4r^2}.$$

Clearly, for the McCormick piecewise linear relaxations we have that $\varepsilon_{rel}^{(h,\mathcal{R}_h^r)} \to 0$ for $r \to \infty$, i.e. using the McCormick piecewise linear relaxation technique, for any $\varepsilon > 0$ we find $r \in \mathbb{N}$ such that $\varepsilon_{rel}^{(h,\mathcal{R}_h^r)} < \varepsilon$.

After we have introduced the relaxations of interest for that paper, we now relate relaxations (RMOP$_{\tilde{S}}$) and lower bounds of the nondominated set $\mathcal{N}$ of (MOP). For the ease of notation we subsequently assume that for a given relaxation $\tilde{S}$ we have that $f(\tilde{S}) \subseteq \operatorname{int}(B)$. We obtain the following lemma.

Lemma 4.3. *Let $f(\tilde{x}) \in \mathcal{N}^{\tilde{S}}$ for some $\tilde{x} \in \tilde{S}$ and some relaxation $\tilde{S}$ of* (MOP)*. Then $f(\tilde{x})$ is nondominated by $\mathcal{N}$, i.e. $f(\tilde{x}) \in \operatorname{int}(B) \setminus (f(S) + \operatorname{int}(\mathbb{R}_+^k))$.*

Proof. Assume for a contradiction that there is some $\bar{y} \in S$ such that $f(\bar{y})$ dominates $f(\tilde{x})$. This contradicts $\bar{y} \in S \subseteq \tilde{S}$ together with the nondominance of $f(\tilde{x})$ w.r.t. (RMOP$_{\tilde{S}}$). □

The above lemma tells us that any stable set $\tilde{N}$ consisting of nondominated points of possibly different relaxations of a given (MOP) is nondominated w.r.t. $\mathcal{N}$. Therefore,

by employing Lemma 3.5, we obtain a lower bound set $L(\tilde{N})$ of the nondominated set $\mathcal{N}$. If we have furthermore any potentially nondominated – and therefore particularly stable – set $N \subseteq f(S)$ available, we obtain an enclosure $E(L(\tilde{N}), U(N))$ of the nondominated set in the sense of Definition 3.1 again by Lemma 3.5. That is the core idea of the method presented in the next section.
We finalize this section with some aspects concerning the sets N and $\tilde{N}$. We want to make use of the idea from [32], where the optimal value is sandwiched between lower bounds coming from relaxed problems and upper bounds coming from solutions of the reduced problems. In the multiobjective setting the lower bound is not a single value anymore, but a stable set consisting of optimal solutions of possibly different relaxations, namely $\tilde{N}$. In the same spirit, also the upper bound is not a single value but a stable set consisting of potentially nondominated points, namely N. As we want to somehow *decrease* this set N towards $\mathcal{N}$, it is updated w.r.t. nondominance, i.e. if we compute a new potentially nondominated point y, we add it to N if y is nondominated w.r.t. N. Furthermore, we discard any points in N which are dominated by y. This is written in Algorithm 4.

Algorithm 4 Updating the set N w.r.t. a point $y \in f(S)$

Require: Stable set N and update point $y \in f(S)$
1: **if** y is nondominated w.r.t. N **then**
2: Set $N = N \cup \{y\}$
3: **end if**
4: Set $N = N \setminus \{y' \in N \mid y < y'\}$
5: **return** Updated set N

By doing so, we ensure that N stays stable and consequently improves towards $\mathcal{N}$ as shown in the next lemma. Note that since $N \subseteq f(S)$ we have that $\mathcal{N}$ is nondominated w.r.t. N, i.e. N actually approximates $\mathcal{N}$ from above.

Lemma 4.4. *Let N_1 be a stable input set and let $y \in f(S)$. Then Algorithm 4 returns a stable set N_2. Furthermore, either $N_1 \subseteq N_2$ or there exist $y_1 \in N_1$ and $y_2 \in N_2$ such that $y_2 < y_1$.*

Proof. We have to distinguish two base cases. Firstly, if y is dominated w.r.t. N_1 Algorithm 4 returns the set $N_2 = N_1$ which is clearly stable and we have that $N_1 \subseteq N_2$. Secondly, if y is nondominated w.r.t. N_1 Algorithm 4 adds y to N_1. If additionally y dominates no point from N_1 we have that $N_2 = N_1 \cup \{y\}$. Clearly, N_2 is stable and $N_1 \subseteq N_2$. On the other hand, if y dominates some points from N_1, say one of these is y_1, then these points are not contained in N_2 and hence N_2 is stable and we have that $y = y_2 < y_1$, as required. □

Let us now turn to the set $\tilde{N}$ which is meant to approach $\mathcal{N}$ from below, i.e. we only want to use the best relaxed solutions available. In that setting we call a nondominated point $\tilde{y}$ of a relaxation $\tilde{S}$ better than a nondominated point $\hat{y}$ of a relaxation $\hat{S}$, if we have that $\hat{y} < \tilde{y}$. Note that if $y' < \tilde{y}$ we know that $\tilde{S} \subsetneq S'$ holds for the corresponding

relaxations. In a similar but reversed way as for N we update the set $\tilde{N}$ as written in Algorithm 5.

Algorithm 5 Updating the set $\tilde{N}$ w.r.t. a point $y \in \operatorname{int}(B) \setminus \left(f(S) + \operatorname{int}\left(\mathbb{R}^k_+\right)\right)$

Require: Stable set $\tilde{N}$ and update point $y \in \operatorname{int}(B) \setminus \left(f(S) + \operatorname{int}\left(\mathbb{R}^k_+\right)\right)$

1: **if** $\tilde{N}$ is nondominated w.r.t. y **then**
2: Set $\tilde{N} = \tilde{N} \cup \{y\}$
3: **end if**
4: Set $\tilde{N} = \tilde{N} \setminus \left\{\tilde{y} \in \tilde{N} \mid \tilde{y} < y\right\}$
5: **return** Updated set $\tilde{N}$

We obtain the analogue to Lemma 4.4. Note that by Lemma 4.3 we know that $\tilde{N}$ is nondominated w.r.t. $\mathcal{N}$, i.e. $\tilde{N}$ actually approximates $\mathcal{N}$ from below.

Lemma 4.5. *Let $\tilde{N}_1$ be a stable input set and let $y \in \operatorname{int}(B) \setminus (f(S) + \operatorname{int}(\mathbb{R}^k_+))$. Then Algorithm 5 returns a stable set $\tilde{N}_2$. Furthermore, either $\tilde{N}_1 \subseteq \tilde{N}_2$ or there exist $y_1 \in \tilde{N}_1$ and $y_2 \in \tilde{N}_2$ such that $y_1 < y_2$.*

Proof. We have to distinguish two base cases. Firstly, if $\tilde{N}_1$ is dominated w.r.t. y Algorithm 5 returns the set $\tilde{N}_2 = \tilde{N}_1$ which is clearly stable and we have that $\tilde{N}_1 \subseteq \tilde{N}_2$. Secondly, if $\tilde{N}_1$ is nondominated w.r.t. y Algorithm 5 adds y to $\tilde{N}_1$. If additionally y is not dominated by any point from $\tilde{N}_1$ we have that $\tilde{N}_2 = \tilde{N}_1 \cup \{y\}$. Clearly, $\tilde{N}_2$ is stable and $\tilde{N}_1 \subseteq \tilde{N}_2$. On the other hand, if y is dominated by some points from $\tilde{N}_1$, say one of these is y_1, then these points are not contained in $\tilde{N}_2$ and hence $\tilde{N}_2$ is stable and we have that $y_1 < y_2 = y$, as required. □

5 General scheme

We have seen at the end of Section 3 that Algorithm 3 is able to compute an enclosure as well as an approximation of the nondominated set of a given (MOP). However, if nonlinear constraint functions are present in (MOP), the scalarized problems arising during Algorithm 3 are MINLP problems. In the end of Section 3 we have also seen that if the used solver, like, e.g., SCIP, is capable of handling the occurring nonlinear constraints one could solve the scalarized single objective problems directly. Nevertheless, if the complexity of (MOP) and therefore the one of the resulting MINLPs increases, the run time of solvers like SCIP for computing a solution to such an MINLP may increase, too. Note that for any computed nondominated point in our approximation we have to solve at least one such MINLP problem, so even a small increase of computational time per problem may cause a tremendous upturn of run time of the whole procedure.
In Section 4 we have presented ideas and concepts from single objective optimization of MINLPs which are meant to reduce complexity and therefore facilitate the computations while solving a scalar MINLP problem. Now one could think of choosing a specific relaxation $\mathcal{R}$ and then using one of the present algorithms for computing

a representation (or enclosure) of the nondominated set of the relaxed mixed-integer linear (or convex) problem, see, e.g., [34, 37] for the biobjective linear case and [19] for the multiobjective convex case, or even Algorithm 3 or [18]. One could argue that if the relaxation $\mathcal{R}$ satisfies some quality criterion, e.g., a small enough estimation error $\varepsilon_{rel}^{\mathcal{R}}$, the approximation (or enclosure) of $\mathcal{N}^{\mathcal{R}}$ can be considered to be an approximation (or enclosure) of $\mathcal{N}$, similar as proposed in [9] for the single objective case. Note that convergence then only relies on the theory of the used multiobjective method.
However, computing such a relaxation and solving the arising problem using an available solution method may be very time consuming as the complexity even of the relaxed problems may increase with ongoing refinement – in particular, as the number of integer variables increases while tightening the relaxations. One strategy for avoiding this, is trying to use *cheap* relaxations whenever possible and refining them only when necessary, e.g., only in specific parts of the image space. This idea of adaptively refining the relaxations while computing an enclosure of the nondominated set of (MOP) is the core of this work.
In the following we present an algorithm similar to Algorithm 3 which makes use of these ideas in order to compute an enclosure of the nondominated set without solving scalarized MINLP, but only MILP and NLP problems (see Algorithm 6).

Before starting the procedure we fix a relaxation technique guaranteeing that the relaxation error quality criterion, namely $\varepsilon_{rel}^{\mathcal{R}} < \varepsilon_{rel}$, is satisfied after a finite number of refinement steps. We introduce the set consisting of all such relaxations

$$\Omega \coloneqq \left\{ \mathcal{R} \mid \varepsilon_{rel}^{\mathcal{R}} < \varepsilon_{rel} \right\},$$

and assume the following for the remainder of the paper.

Assumption 5.1. *For any relaxation technique and any initial relaxation $\mathcal{R}^I$. Let $\mathcal{R}^I \supsetneq \mathcal{R}^1 \supsetneq \mathcal{R}^2 \supsetneq \dots$ be a chain of strictly decreasing relaxations. Then for any $\varepsilon_{rel} > 0$ there exists an $s \in \mathbb{N}$ with $\varepsilon_{rel}^{\mathcal{R}^s} < \varepsilon_{rel}$, i.e. $\mathcal{R}^l \in \Omega$ for all $l \geq s$.*

Furthermore, if $\mathcal{R} \in \Omega$ we consider any feasible point $\tilde{x} \in \tilde{S}$ of the corresponding relaxed problem ($\text{RMOP}_{\tilde{S}}$) based on the feasible set $\tilde{S} = S^{\mathcal{R}}$ as feasible point of (MOP), i.e. $\tilde{x} \in S$. Consequently, by Lemma 4.3 for any efficient point $\tilde{x} \in \tilde{S}$ of ($\text{RMOP}_{\tilde{S}}$) we have that $f(\tilde{x}) \in \mathcal{N}$. This means, that for a relaxation $\mathcal{R} \in \Omega$ we consider any nondominated point of ($\text{RMOP}_{\tilde{S}}$) as nondominated point of (MOP). For the ease of notation we write $\tilde{x} \in S$ if $\tilde{x} \in S^{\mathcal{R}}$ for some $\mathcal{R} \in \Omega$ for the remainder of that paper. Note that Assumption 5.1 holds for the McCormick piecewise linear relaxations introduced in Section 4.
In Step 3, we initialize the set

$$\mathcal{D}(U) \coloneqq \{(u, \mathcal{R}) \mid u \in U,\ \mathcal{R} \text{ caused the computation of } u\},$$

consisting of any present local upper bound together with the relaxation $\mathcal{R}$ which was needed to obtain this specific local upper bound. We say that the relaxation $\mathcal{R}$ caused the computation of a local upper bound u if u entered the set of local upper bounds after it was updated w.r.t. a point y whose computation relied on $\mathcal{R}$. This could be

Algorithm 6 General scheme for computing an enclosure of the nondominated set relying on relaxation and scalarization techniques.

Require: box $B = \left[z^\ell, z^u\right]$ with $f(S) \subseteq \operatorname{int}(B)$, termination tolerance $\varepsilon_{\text{encl}} > 0$, offset factor $\varepsilon_{\text{encl}} > \delta > 0$, initial relaxation $\mathcal{R}^I$, estimation error tolerance $\varepsilon_{rel} > 0$

Initialize potentially nondominated set $N = \emptyset$ and set of local upper bounds $U = \{z^u\}$
Initialize set of best relaxed solutions $\tilde{N} = \emptyset$ and set of local lower bounds $L = \{z^\ell\}$
Initialize the sets $E = E(L, U)$, $\mathcal{D}(U) = \left\{\left(z^u, \mathcal{R}^I\right)\right\}$ and $\mathcal{D}(\tilde{N}) = \emptyset$
while $w(E) \geq \varepsilon_{\text{encl}}$ **do**
  $U_{\text{loop}} = U$
  for $u \in U_{\text{loop}}$ **do**
    if there exists $\ell \in L$ with $\ell \leq u$ and $s(\ell, u) \geq \varepsilon_{\text{encl}}$ **then**
      `done` = false
      while `done` = false **do**
        Set $\mathcal{R}_{\text{current}} = \min\{\mathcal{R}^u, \mathcal{R}'\}$, where $(u, \mathcal{R}^u) \in \mathcal{D}(U)$ and $\mathcal{R}' \subsetneq \min\left\{\mathcal{R}^{\tilde{y}} \mid \left(\tilde{y}, \mathcal{R}^{\tilde{y}}\right) \in \mathcal{D}\left(\tilde{N}\right) \text{ and } \tilde{y} < u - \varepsilon_{\text{encl}} e\right\}$
        Set relaxation $\tilde{S} = S^{\mathcal{R}_{\text{current}}}$
        if there exists $\tilde{y} = f(\tilde{x}) \in \mathcal{N}^{\tilde{S}}$ with $\tilde{y} < u - \delta e$ **then**
          if $\tilde{N}$ is nondominated w.r.t. $\tilde{y}$ **then**
            Update $\tilde{N}$ and L w.r.t. $\tilde{y}$ using Alg. 5 and 2 and set $\mathcal{R}^{\tilde{y}} = \mathcal{R}_{\text{current}}$
            if $\varepsilon_{rel}^{\mathcal{R}_{\text{current}}} < \varepsilon_{rel}$ **then**
              Update N and U w.r.t. $\tilde{y}$ using Alg. 4 and 1 and set $\mathcal{R}^u = \mathcal{R}_{\text{current}}$ for any new local upper bound u
              `done` = true
            else
              if there exists solution y to (redMOP($\tilde{x}_I$)) with $y < u - \delta e$ **then**
                Update N and U w.r.t. y using Alg. 4 and 1 and set $\mathcal{R}^u = \mathcal{R}_{\text{current}}$ for any new local upper bound u
                `done` = true
              else
                Choose $\mathcal{R}$ with $\mathcal{R}_{\text{current}} \supsetneq \mathcal{R}$ and set $\mathcal{R}^u = \mathcal{R}$
              end if
            end if
          else
            Choose $\mathcal{R}$ with $\mathcal{R}_{\text{current}} \supsetneq \mathcal{R}$ and set $\mathcal{R}^u = \mathcal{R}$
          end if
        else
          Update L w.r.t. $u - \delta e$ using Algorithm 2
          `done` = true
        end if
      end while
    end if
  end for
end while
return Enclosure $E(L, U)$ satisfying $w(E) < \varepsilon_{\text{encl}}$ and approximation N of $\mathcal{N}_{\varepsilon_{\text{encl}}}$

either the case if y is the solution of the relaxed problem corresponding to $\mathcal{R}$ and $\mathcal{R} \in \Omega$ or if y is a nondominated point of (redMOP($\tilde{x}_I$)), where $\tilde{x}_I$ was computed using the relaxation $\mathcal{R}$. Furthermore, we initialize the set

$$\mathcal{D}\left(\tilde{N}\right) \coloneqq \left\{(\tilde{y}, \mathcal{R}) \mid \tilde{y} \in \tilde{N},\ \mathcal{R} \text{ caused the computation of } \tilde{y}\right\},$$

consisting of solutions of relaxed problems together with their corresponding relaxation $\mathcal{R}$.

Suppose now we are at the beginning of the l-th call of the outer `while`-loop in Step 4 and we have that $w(E) \geq \varepsilon_{\text{encl}}$. We fix the set U_{loop} to be the current assignment of the set of local lower bounds U and start the `for`-loop in Step 6. In that `for`-loop, let $\hat{u} \in U_{\text{loop}}$ such that there exists $\ell \in L$ with $\ell \leq \hat{u}$ and $s(\ell, \hat{u}) \geq \varepsilon_{\text{encl}}$, i.e. the search zone determined by ℓ and $\hat{u}$ is not yet well enough explored and we set `done` = false. We use the indicator `done` to determine whether we achieved an improvement w.r.t. $\hat{u}$, i.e. the inner `while`-loop ensures that we concentrate on $\hat{u}$ until we made some improvement. We say that we improved $\hat{u}$ if we entered one of the `if`-statements in the Steps 15 and 19 or the `else`-statement in Step 29 as in these steps either a potentially nondominated point y with $y < \hat{u} - \delta e$ is found or the search region $c(\hat{u})$ is declared to be well enough explored.
However, given the current local upper bound $\hat{u}$ we have to choose an appropriate relaxation $\mathcal{R}_{\text{current}}$ for executing our computations. This is realized in Step 10. We choose a relaxation at least as fine as the relaxation which led to $\hat{u}$, i.e. $\mathcal{R}^{\hat{u}} \supseteq \mathcal{R}_{\text{current}}$. Furthermore, if there exists some relaxed solution $\tilde{y} \in \tilde{N}$ with $\tilde{y} < \hat{u} - \varepsilon_{\text{encl}} e$ we choose a strictly finer relaxation than $\mathcal{R}^{\tilde{y}}$, i.e. we ensure that $\mathcal{R}^{\tilde{y}} \supsetneq \mathcal{R}_{\text{current}}$. Note that the strictness of the inclusion is not necessary for the convergence of Algorithm 6 since the method also refines the relaxations if the incumbent relaxed solution did not lead to an improvement of the lower bound set. However, for some problems it may be of advantage to refine the relaxations more aggressively instead of solving *cheaper* problems that do not have a high chance of leading to a significant improvement of the lower bound set. In fact, not forcing the inclusion $\mathcal{R}_{\text{current}} \subseteq \mathcal{R}^{\tilde{y}}$ to be strict makes it impossible to find a relaxed solution y' dominating $\tilde{y}$, and therefore satisfying $\hat{u} - \varepsilon_{\text{encl}} e \leq y'$, as can be seen in Figure 5. The possibly negative effect of not forcing strictness can be seen in the comparison of Figure 7 and Figure 8, where we can observe an increase in runtime as well as in the number of problems to be solved. However, refining too aggressively may also be a problem as it may result in solving harder problems than necessary. From our first observations it is a good strategy to take the coarsest possible relaxation without loosing the strictness of the inclusion – at least with using our basic refinement strategy.

After that we initialize the relaxation $\tilde{S} = S^{\mathcal{R}_{\text{current}}}$, solve (WSP($\alpha; u$)) with $u = \hat{u}$ for some $\alpha \in \text{int}(\mathbb{R}^k_+)$ and feasible set $\tilde{S}$ and then decide whether we are able to enter the `if`-statement in Step 12. If (WSP($\alpha; u$)) is infeasible, we declare the current search region $c(\hat{u})$ as well enough explored by the same arguments as in Algorithm 3, and set `done` = true. If, otherwise, there exists a solution $\tilde{y}$ to (WSP($\alpha; u$)) we check if $\tilde{N}$ is nondominated w.r.t. $\tilde{y}$, i.e. if $\tilde{y}$ improves the set $\tilde{N}$. If that is not the case, i.e. if there exists $y' \in \tilde{N}$ with $\tilde{y} \leq y'$, we have to restart the inner `while`-loop with a

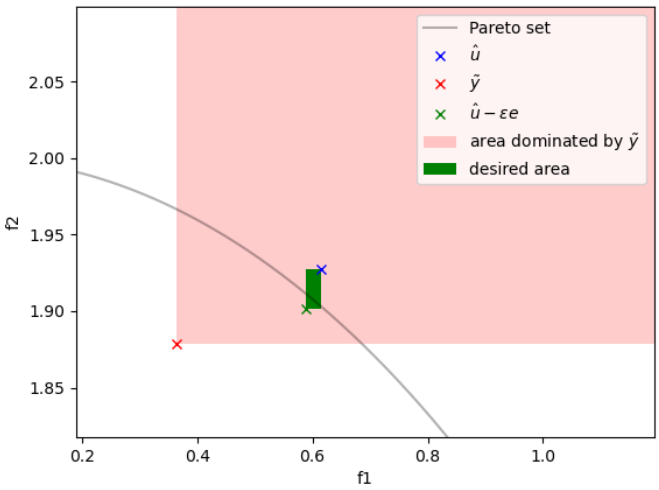

Figure 5: A strict refinement of the relaxation is needed in order to obtain solutions closer to the desired area as it is dominated by $\tilde{y}$.

finer relaxation as the current one did not lead to any improvement. If otherwise, $\tilde{N}$ is nondominated w.r.t. $\tilde{y}$, it is reasonable to move on as $\tilde{y}$ suggests an improvement of $\hat{u}$. Consequently, we update the sets $\tilde{N}$ and L w.r.t. $\tilde{y}$ and save the corresponding relaxation. If now the relaxation is fine enough in the sense of [9], i.e. $\mathcal{R}_{\text{current}} \in \Omega$, we consider $\tilde{y}$ as nondominated point of (MOP), update the sets N and U w.r.t. $\tilde{y}$ and set `done` = true. If otherwise the relaxation is not yet fine enough we try to make use of the idea from [32], i.e. using parts of the relaxed solution to set up the reduced problem (redMOP($\tilde{x}_I$)). We solve the corresponding (WSP($\alpha; u$)) and if it has a solution we obtain a potentially nondominated point, i.e. update the sets N and U and set `done` = true. If it is infeasible, we have to restart with a finer relaxation, since $\mathcal{R}_{\text{current}}$ suggested wrongly that we would find a potentially nondominated point.

We turn now to proving correctness and finiteness of Algorithm 6.

Lemma 5.2. *Let $\tilde{N}$ be the set of best relaxed solutions at some arbitrary point in Algorithm 6. Then $\tilde{N} \subseteq \operatorname{int}(B) \setminus (f(S) + \operatorname{int}(\mathbb{R}^k_+))$ and $\tilde{N}$ is stable.*

Proof. We initialize the set $\tilde{N} = \emptyset$. During Algorithm 6 the set $\tilde{N}$ is updated only in Step 14. The update takes place w.r.t. a point $\tilde{y} \in \operatorname{int}(B) \setminus (f(S) + \operatorname{int}(\mathbb{R}^k_+))$ by Lemma 4.3. Thus, by correctness of Algorithm 5 we obtain that $\tilde{N} \subseteq \operatorname{int}(B) \setminus (f(S) + \operatorname{int}(\mathbb{R}^k_+))$ and $\tilde{N}$ is stable. □

Lemma 5.3. *Let N be the set of potentially nondominated points at some arbitrary point in Algorithm 6. Then $N \subseteq f(S)$ and N is stable.*

Proof. We initialize the set $N = \emptyset$. During Algorithm 6 the set N is updated only in Steps 16 and 20. In both cases the update takes place w.r.t. a point $y \in f(S)$ – note that in Step 16 we have $\mathcal{R}_{\text{current}} \in \Omega$ and therefore $\tilde{y} \in f(S)$. Thus, by correctness of Algorithm 4 we obtain that $N \subseteq f(S)$ and N is stable. □

Similar as in Section 3 we prove correctness of the sets U and L.

Lemma 5.4. *Let U be the local upper bound set at some arbitrary point in Algorithm* 6. *Then U is an upper bound set in the sense of Definition* 3.1.

Proof. We initialize the set $U = \{z^u\}$. During Algorithm 6 the set U is updated only in Steps 16 and 20. In both cases U is updated w.r.t. a point $y \in f(S)$ – note that in Step 16 we have $\mathcal{R}_{\text{current}} \in \Omega$ and therefore $\tilde{y} \in f(S)$. Thus, by correctness of Algorithm 1 we obtain that U is a local upper bound set w.r.t. the set $N \subseteq f(S)$. The claim follows by Lemma 3.5. □

Lemma 5.5. *Let L be the local lower bound set at some arbitrary point in Algorithm* 6. *Then L is a lower bound set in the sense of Definition* 3.1.

Proof. We initialize the set $L = \{z^\ell\}$. During Algorithm 6 the set L is updated only in Steps 14 and 30. If updated in Step 14, L is updated w.r.t. a point $\tilde{y} \in \text{int}(B)\backslash(f(S)+\text{int}(\mathbb{R}^k_+))$. If otherwise updated in Step 30, it is updated w.r.t. $y = \hat{u}-\delta e$ for some local upper bound $\hat{u} \in U$. Since we only enter Step 30, if there exists no $\bar{y} \in \mathcal{N}^{\tilde{S}}$ satisfying $\bar{y} \leq \hat{u} - \delta e$ we particularly know that $y = \hat{u} - \delta e \notin f(S) + \text{int}(\mathbb{R}^k_+)$. Hence, by correctness of Algorithm 2 we obtain that L is a local lower bound set w.r.t. some set $N' \subseteq \text{int}(B) \setminus (f(S) + \text{int}(\mathbb{R}^k_+))$. Note that $\tilde{N} \subseteq N'$. The claim follows by Lemma 3.5. □

After proving that at any point in Algorithm 6 the sets N, $\tilde{N}$, L and U satisfy the requirements, we proceed with showing termination of Algorithm 6 after a finite number of iterations of the outer `while`-loop. In order to do so, we first prove that the inner `while`-loop is terminated after a finite number of iterations.

Lemma 5.6. *Let $\varepsilon_{encl} > \delta > 0$, $\varepsilon_{rel} > 0$, $\mathcal{R}^I$ and $z^\ell, z^u \in \mathbb{R}^k$ be the input parameters of Algorithm* 6. *Moreover, let $\hat{u} \in U_{loop}$ be the local upper bound chosen in the `for`-loop at some arbitrary point of Algorithm* 6. *Assume that there exists $\ell \in L$ with $\ell \leq \hat{u}$ and $s(\ell, \hat{u}) \geq \varepsilon_{encl}$. Then after finitely many iterations of the inner `while`-loop we have that `done` = true.*

Proof. We have to show that after finitely many iterations we either enter one of the `if`-statements in Steps 15 or 19 or we enter the `else`-statement in Step 29. For a contradiction we assume that none of them is entered after finitely many iterations of the inner `while`-loop. Then, every iteration is executed with a relaxation strictly finer than the one before. This is due to the fact that any already executed iteration of the inner `while`-loop terminated either with Step 23 or Step 27. Thus, we have an infinite chain of relaxations $\mathcal{R}^1 \supsetneq \mathcal{R}^2 \supsetneq \ldots \supsetneq \mathcal{R}^l \supsetneq \ldots$, where $\mathcal{R}_l$ denotes the relaxation corresponding to the l-th iteration of the inner `while`–loop. Now, by Assumption 5.1 there exists $s \in \mathbb{N}$ such that $\mathcal{R}^s \in \Omega$ and therefore $\varepsilon^{\mathcal{R}^s}_{rel} < \varepsilon_{rel}$. Hence, in the s-th

iteration of the inner `while`-loop we either enter the `if`-statement in Step 12 or the `else`-statement in Step 29. By our assumption we do not enter the `else`-statement. By Lemma 4.3 we have that $\tilde{N} \subseteq \operatorname{int}(B) \setminus (f(S) + \operatorname{int}(\mathbb{R}^k_+))$ and therefore $\tilde{N}$ is nondominated w.r.t. $\mathcal{N}$. By the fact that $\mathcal{R}^s \in \Omega$ we have that $\tilde{y} \in \mathcal{N}$ and therefore we enter the `if`-statement in Step 13 and subsequently the one in Step 15, a contradiction. □

Similar as for Algorithm 3 we can prove some decrease in some edge lengths after any iteration of the outer `while`-loop.

Theorem 5.7. *Let $\varepsilon_{encl} > \delta > 0$, $\varepsilon_{rel} > 0$, $\mathcal{R}^I$ and $z^\ell, z^u \in \mathbb{R}^k$ be the input parameters of Algorithm 6. Moreover, let L^{start} and U^{start} be the local lower and upper bound sets at the beginning of some iteration in Algorithm 6, i.e. at the begin of the outer `while`-loop of some iteration. Accordingly denote by L^{end}, U^{end} the sets at the end of this iteration. Then for any $\ell^e \in L^{end}$ and any $u^e \in U^{end}$ with $\ell^e \leq u^e$ there exist $\ell^s \in L^{start}$ and $u^s \in U^{start}$ such that the following hold:*

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

2) *There exists an index $j \in [k]$ such that*

$$(u^e - \ell^e)_j < \max\left\{(u^s - \ell^s)_j - \delta, \varepsilon_{encl}\right\}.$$

Proof. The proof of part 1) works exactly the same as in the proof of Theorem 3.10. We therefore directly show part 2). Assume for a contradiction that there exist $\ell^e \in L^{\text{end}}$ and $u^e \in U^{\text{end}}$ such that for any $\ell \in L^{\text{start}}$ and $u \in U^{\text{start}}$ with $\ell \leq \ell^e \leq u^e \leq u$ and any index $i \in [k]$ we have that

$$(u^e - \ell^e)_i \geq \max\left\{(u - \ell)_i - \delta,\ \varepsilon_{\text{encl}}\right\}. \tag{8}$$

In particular, (8) holds for the ancestors of ℓ^e and u^e in L^{start} and U^{start}, namely ℓ^s and u^s. We consider now the point in Algorithm 6, where u^s is chosen in the `for`-loop in Step 6. Recall that we have introduced the sets $P(\ell^e)$ and $P(u^e)$ in the proof of Theorem 3.10. Note that u^s might not be the first local upper bound from U_{loop} considered in the `for`-loop and therefore it might be the case that $u^s \notin U_{\text{current}}$, where U_{current} is the current assignment of U. However, since for any $i \in [k]$ we have that $(u^e - \ell^e)_i \geq \varepsilon_{\text{encl}}$ we know that for $\ell' \in P(\ell^e) \cap L_{\text{current}}$ we have that $(u^s - \ell')_i \geq \varepsilon_{\text{encl}}$ for any $i \in [k]$, where L_{current} denotes the current assignment of L. Hence, we have that

$$s(\ell', u^s) \geq \varepsilon_{\text{encl}} \tag{9}$$

and we therefore enter the `if`-statement in Step 7. Similarly, we fix $u' \in P(u^e) \cap U_{\text{current}}$. Note that

$$\ell^s \leq \ell \leq \ell^e \quad \text{and} \quad u^e \leq u \leq u^s \tag{10}$$

for any $\ell \in P(\ell^e) \cap L$ and $u \in P(u^e) \cap U$ and for any L and U. We denote by L_{updated} and U_{updated} the assignments of L and U in the end of that iteration of the `for`-loop.

As the exit from the inner `while`-loop can happen in three different ways, we have to distinguish three main cases, namely if we set `done` = true in Step 17 (case **A**), if we set `done` = true in Step 21 (case **B**) or if we set `done` = true in Step 31 (case **C**). Note that Lemma 5.6 guarantees that after finitely many iterations of the inner `while`-loop one of the three above options is actually chosen.

(A) In that case we entered the `if`-statement in Step 15, i.e. there exists $\tilde{y} \in f(S)$ with $\tilde{y} < u^s - \delta e$. We have to distinguish two cases.
Case A.1: $\tilde{y} < u'$. Then u' would be removed during the update of U_{current} using Algorithm 1, i.e. $u' \notin U_{\text{updated}}$. We have the candidates

$$u^i = \left(\tilde{y}_i, u'_{-i}\right), \quad \text{for } i \in [k],$$

from which at least one belongs to U_{updated} by (3). Say $u^j \in U_{\text{updated}}$. Using (10), we compute

$$(u^e - \ell^e)_j \leq \left(u^j - \ell^e\right)_j \leq \tilde{y}_j - \ell^s_j < (u^s - \ell^s)_j - \delta,$$

a contradiction to (8).
Case A.2: $\tilde{y} \not< u'$. Then there exists $j \in [k]$ with $u'_j \leq \tilde{y}_j$. Again, using (10), we compute

$$(u^e - \ell^e)_j \leq (u' - \ell^e)_j \leq \tilde{y}_j - \ell^s_j < (u^s - \ell^s)_j - \delta,$$

a contradiction to (8).

(B) In that case we entered the `if`-statement in Step 19, i.e. there exists $y \in f(S)$ with $y < u^s - \delta e$. Again, we have to distinguish two cases, which both work exactly the same as the ones from (A).

(C) In that case we entered the `else`-statement in Step 29, i.e. there exists no $\tilde{y} \in \mathcal{N}^{\tilde{S}}$ with $\tilde{y} < u^s - \delta e =: y$. Therefore, the set L is updated w.r.t. y. We have to distinguish two cases.
Case C.1: $\ell' < y$. Then ℓ' would be removed during the update of L_{current} using Algorithm 2, i.e. $\ell' \notin L_{\text{updated}}$. We have the candidates

$$\ell^i = \left(y_i, \ell'_{-i}\right), \quad \text{for } i \in [k],$$

from which at least one belongs to L_{updated} by (3). Say $\ell^j \in L_{\text{updated}}$. Using (10), we compute

$$(u^e - \ell^e)_j \leq \left(u^e - \ell^j\right)_j \leq u^s_j - y_j = \delta < \varepsilon_{\text{encl}},$$

a contradiction to (8).
Case C.2: $\ell' \not< y$. Then there exists $j \in [k]$ with $\ell' \geq y_j = u^s_j - \delta$, i.e. particularly $s(\ell', u^s) \leq \delta < \varepsilon_{\text{encl}}$, a contradiction to (9).

Obtaining a contradiction in all possible cases shows that our assumption (8) cannot be true and therefore statement 2) is true, which completes the proof. □

Similar as in the case of Algorithm 3, Theorem 5.7 enables us to prove finiteness of Algorithm 6.

Theorem 5.8. *Let $\varepsilon_{encl} > \delta > 0$, $\varepsilon_{rel} > 0$, $\mathcal{R}^I$ and $z^\ell, z^u \in \mathbb{R}^k$ be the input parameters of Algorithm 6. We define*

$$\Delta \coloneqq \left\| z^u - z^\ell \right\|_\infty \quad \text{and} \quad \kappa \coloneqq k \left\lceil \frac{\Delta - \varepsilon_{encl}}{\delta} \right\rceil + 1.$$

Then the number of iterations of Algorithm 6, i.e. the number of iterations of the outer `while`-loop, is bounded by $\max\{1, \kappa\}$. Furthermore, Algorithm 6 terminates after finitely many steps.

Proof. The proof for bounding the number of calls of the outer `while`-loop works exactly the same as the one of Theorem 3.11. Termination after finitely many steps follows by the fact that in every iteration of the outer `while`-loop, the inner `while`-loop is only called finitely many times as shown in Lemma 5.6. □

Corollary 5.9. *Let $\varepsilon_{encl} > \delta > 0$, $\varepsilon_{rel} > 0$, $\mathcal{R}^I$ and $z^\ell, z^u \in \mathbb{R}^k$ be the input parameters of Algorithm 6. Then after finitely many iterations of the outer `while`-loop an enclosure E of the nondominated set $\mathcal{N}$ satisfying $w(E) < \varepsilon_{encl}$ is returned.*

Proof. Theorem 5.8 tells us that Algorithm 6 terminates after at most κ iterations of the outer `while`-loop, i.e. the output set E_κ satisfies $w(E_\kappa) < \varepsilon_{\text{encl}}$. By Lemma 5.4 and Lemma 5.5 we know that the set L, resp. U, is a lower, resp. upper, bound set in the sense of Definition 3.1 at any point of Algorithm 6. In particular, this holds for L_κ and U_κ. Thus, E_κ is an enclosure of the nondominated set $\mathcal{N}$ satisfying $w(E_\kappa) < \varepsilon_{\text{encl}}$. □

Remark 5.10. Note that one could also use two different off-set factors such that $0 < \delta < \tilde{\delta} < \varepsilon_{\text{encl}}$. The idea is that for a relaxed solution $\tilde{y}$ we expect the eligible solutions y of (redMOP($\tilde{x}_I$)) to satisfy $\tilde{y} + \rho e < y$ for some sufficiently small $\rho > 0$. If now $\tilde{y}$ satisfies $\tilde{y} < u - \delta e$ too tightly, e.g., if $\tilde{y} + \rho e \not< u - \delta e$, we do not find any point y with the desired property in Step 19 and therefore would refine the relaxation. This is not a problem in general, as we expect $\rho \to 0$ for finer relaxations, i.e. at some point we either do not find any admissible points $\tilde{y}$ anymore and therefore declare the search region $c(u)$ for well enough explored, or we find admissible points y. However, this may be very time consuming as we might have to refine the relaxations and repeat the computations a few times. Thus, using two different off-set factors δ and $\tilde{\delta}$ may help in terms of run time by being more restrictive for $\tilde{y}$ by requiring $\tilde{y} < u - \tilde{\delta} e$ in Step 12 in comparison to y where we require $y < u - \delta e$ in Step 19, where $0 < \delta < \tilde{\delta}$.

We conclude this section with the performance of the described method on the problem of Example 3.13. Again, as for Algorithm 3, we did not exactly implement the procedure given in Algorithm 6 but a slight modification. In fact, we do not iterate through all local upper bounds in every iteration of the outer `while`-loop, but choose only one local upper bound $\hat{u}$ where the current width is attained, i.e. $\hat{u} \in \{u \in U \mid \exists \ell \in L \colon s(\ell, u) = w(E)\}$. Again, finiteness of that implementation is not anymore guaranteed by Theorem 5.8. But if it terminates after finitely many steps,

we still have that $w(E) < \varepsilon_{\text{encl}}$. Again we use a relative shortest edge calculation as described in Section 3. Furthermore, we use a relative relaxation error calculation. For example, given a quadratic term $h(x) = x^2$ on the interval $x^\ell \leq x \leq x^u$ we compute the relative relaxation error via

$$\varepsilon_{rel}^{(h,\mathcal{R}_h^r)} = \frac{\left(x^u - x^\ell\right)^2}{4r^2} \frac{1}{\max\{x^2 \mid x^\ell \leq x \leq x^u\}},$$

where $r \in \mathbb{N}$ is the number of considered equidistant partitions. Note that $\varepsilon_{rel}^{(h,\mathcal{R}_h^r)} \to 0$ for $r \to \infty$, i.e. refining relaxations by increasing the number of equidistant partitions satisfies Assumption 5.1. Furthermore, note that the relaxation $\mathcal{R}$ only depends on the number of equidistant partitions, i.e. we identify $\mathcal{R}$ by r in the following.
We consider the same instances of Example 3.13 as in Section 3:

- The first one is determined by $n = 2$, $r = 1$, $m = 3$ as well as $m^1 = (3, 0)^\top$, $m^2 = (2, 1)^\top$ and $m^3 = (0, 3)^\top$. The enclosure computed by Algorithm 6 with relative tolerance $\varepsilon_{\text{encl}} = 0.0125$ (this is the equivalent of a standard tolerance of $\varepsilon_{\text{encl}} = 0.05$) and unique off–set factor $\delta = 0.95\varepsilon_{\text{encl}}$ is depicted in Figure 6 (Left). In Figure 6 (Right) the usage counter of each degree of relaxation is depicted. The relaxation degree equal to 1 corresponds to the McCormick relaxation with no extra partition of the intervals, i.e. the intervals are not partitioned at all. The relaxation degree equal to 2 corresponds to one additional breaking point per considered interval, i.e. the original intervals are partitioned into two intervals. One can see that the method uses maximally four partitions per interval, but mostly one or two. We used a tolerance for the relative relaxation error of $\varepsilon_{rel} = 0.01$. This yields the necessity of at least five partitions per variable to satisfy the relaxation error criterion. We can see that the method does not need to go all the way it is allowed to since four partitions seem to be enough. Of course, if we increase the tolerance of the relative relaxation error, the number of needed partitions decreases.

- The second one is determined by $n = 3$, $r = 1$, $m = 3$ as well as $m^1 = (1, 0, 0)^\top$, $m^2 = (0, 1, 0)^\top$ and $m^3 = (0, 0, 1)^\top$. The enclosure computed by Algorithm 6 with relative tolerance $\varepsilon_{\text{encl}} = 0.05$ (this is equivalent to an absolute tolerance of $\varepsilon_{\text{encl}} = 0.1$) and unique off–set factor $\delta = 0.95\varepsilon_{\text{encl}}$ is depicted in Figure 7 (Left). In Figure 7 (Right) the use counters of each relaxation is depicted. Again we used a relative relaxation error tolerance of $\varepsilon_{rel} = 0.01$, which yields a minimum of five partitions per variable to obtain an error smaller than the tolerance. In particular that means that if the current number of partitions is greater than five, we do not need to refine any more. We can see that the method makes extensive use of that, i.e. in most of the iterations a relaxation satisfying the relaxation error criterion is chosen.

However, comparing the number of problems to be solved as well as especially the computational time with the ones from Section 3, one can see that the power of available solvers like SCIP dealing with nonlinear – and especially nonconvex – constraints

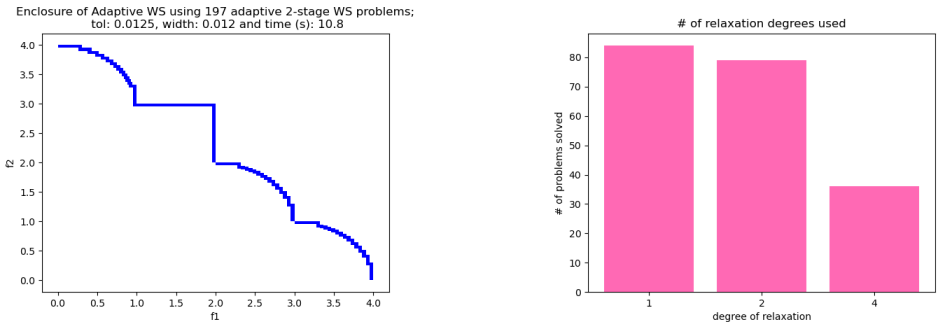

Figure 6: Computational results on first instance in Example 3.13. **Left:** enclosure given by LLBs and LUBs of (Circles) obtained by Algorithm 6 with WSM. **Right:** counter of used degrees of relaxations.

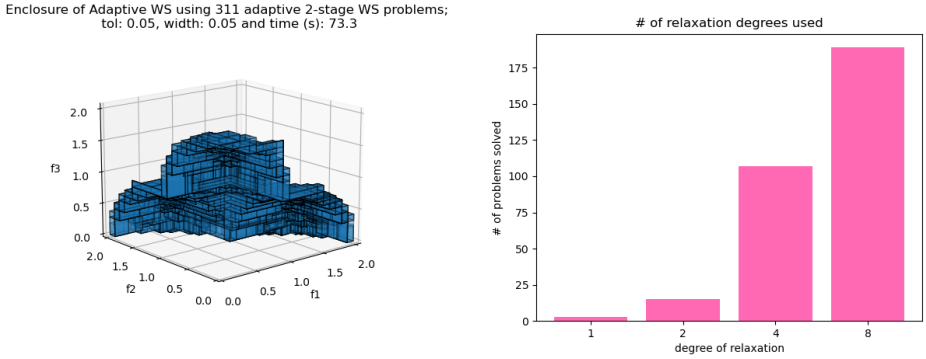

Figure 7: Computational results on second instance in Example 3.13. **Left:** enclosure given by LLBs and LUBs of (Circles) obtained by Algorithm 6 with WSM. **Right:** counter of used degrees of relaxations.

makes the use of relaxations redundant in some cases, as e.g. the ones considered above. However, with increasing complexity of the problems one might have an advantage by only considering relaxations instead of the original problem, as can be seen in the next section.

6 Application to the multiobjective optimization of decentralized energy supply networks

In this section we present numerical results of the described method on some network optimization problem. In fact, aiming to model a decentralized energy supply network we obtain an MINLP optimization problem. The general network structure is a graph,

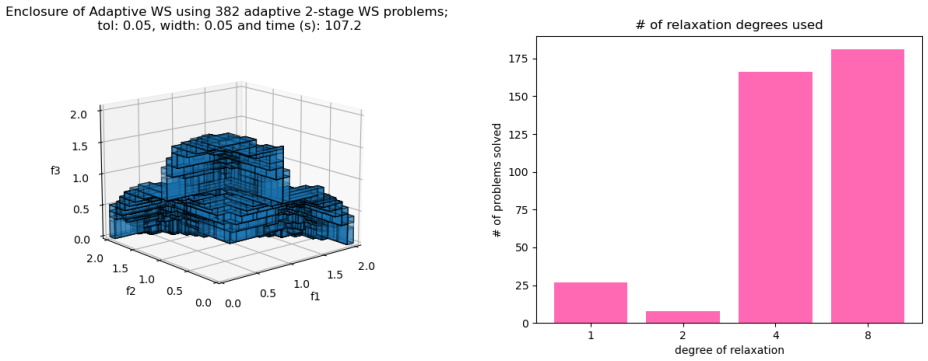

Figure 8: Computational results on second instance in Example 3.13 with no strict inclusion in Step 10 of Algorithm 6. **Left:** enclosure given by LLBs and LUBs of (Circles) obtained by Algorithm 6 with WSM. **Right:** counter of used degrees of relaxations.

where the nodes represent individual consumers and the edges connect the consumer nodes with the so-called source node, where energy is supplied. The mixed-integer character is coming from certain decision options available in the optimization process, e.g., if a gas pipe is laid at some edge or not. Furthermore, we take stationary models of energy flow into account, namely an equation based on the Ohmic law for the electricity flow as well as the Darcy-Weisbach equation for gas flow. As both of them are nonlinear the resulting optimization problem has the mentioned MINLP structure. As objective functions we use the overall costs for realizing a given network plan on the one hand and the carbon emissions of that network plan on the other hand. Naturally, a cheap network plan results in high carbon emissions and a low carbon emission can be obtained by, e.g., investing in energy-efficient house renovation which results in higher costs. Thus, we have a classical (MOP) with two conflicting objective functions. Details regarding the modeling aspects can be found in [27].
For the present paper we consider three network instances of such decentralized energy supply networks, namely:

	`network 1` $\subset$	`network 2` $\subset$	`network 3`
# nodes	12	20	39
# binaries	108	189	360
# variables	484	829	1570
# constraints	620	1064	2014

If, e.g., we set up a single objective optimization problem with cost minimization as objective function, we obtain the following computational times

- `network 1`: 0.57 s
- `network 2`: 3.35 s

- `network 3`: > 3h,

using the SCIP solver with the standard settings[2] from the `pyscipopt`-package (cf. [29]).
For testing the new method we use a relative width tolerance $\varepsilon_{\text{encl}} = 0.03$ as well as two off-set factors $\tilde{\delta} = 0.95\varepsilon_{\text{encl}}$ and $\delta = 0.8\varepsilon_{\text{encl}}$. In the network models the present nonlinearities are of the following form:

- For modeling the low-voltage energy flow we use for instance

$$\mathrm{R}^{\mathrm{e}}_{i,j} f^{\mathrm{e}}_{\mathrm{in},i,j} = a^{\mathrm{e}} u_j \bar{u}_{i,j}, \tag{11}$$

  where $\mathrm{R}^{\mathrm{e}}_{i,j} > 0$ denotes the resistance of the underlying cable at arc (i,j), $a^{\mathrm{e}} > 0$ the calorific multiplier of three-phase electric power flows, $f^{\mathrm{e}}_{\mathrm{in},i,j}$ is a variable representing the electric power flow on arc (i,j) into j. The variable u_i denotes the electrical voltage at node i and the variable $\bar{u}_{i,j} = u_i - u_j$ the voltage drop on arc (i,j). Consequently, we have a quadratic term u_i^2 and a bilinear term $u_i u_j$ appearing in (11). For the computation of the corresponding relative relaxation errors the box constraints $360 \leq u_i, u_j \leq 440$ are relevant. Thus, if we partition the corresponding intervals into r equidistant intervals, i.e. use the relaxation $\mathcal{R}^r$, we obtain

$$\varepsilon^{\mathcal{R}^r}_{rel} = \frac{80^2}{4r^2} \frac{1}{\max\{x^2 \,|\, 360 \leq x \leq 440\}},$$

  and therefore the number of partitions of each interval to fall below a given tolerance ε_{rel} is given by

$$r = \left\lceil \frac{40}{440} \frac{1}{\sqrt{\varepsilon_{rel}}} \right\rceil.$$

  Thus, if we require an relative relaxation error $\varepsilon_{rel} = 0.01$ we have to partition the corresponding intervals into at least $r = 1$ partitions, i.e. we do not have to partition at all.

- For modeling low-pressure gas supply we use a reformulation of the Darcy-Weisbach equation avoiding the use of the sign-function as proposed in [5]. By doing so, we obtain for instance

$$\mathrm{R}^{\mathrm{g}}_{i,j} \bar{q}^2_{ij} \leq \bar{p}_{\max} y_{i,j}, \tag{12}$$

  where $\mathrm{R}^{\mathrm{g}}_{i,j} > 0$ denotes the resistance constant of the underlying gas pipeline on arc (i,j), $\bar{q}_{ij}$ the gas flow on arc (i,j), $\bar{p}_{\max} > 0$ the maximal pressure loss allowed in the network as well as a binary decision variable $y_{i,j}$ indicating if a

[2] Note that the computational time needed for solving `network 3` is drastically reduced if one increases the tolerance for termination.

gas pipe is laid at arc (i, j). The relevant box constraints are $-150 \leq \bar{q}_{ij} \leq 150$ and partitioning into r intervals, i.e. using relaxation $\mathcal{R}^r$, we obtain

$$\varepsilon_{rel}^{\mathcal{R}^r} = \frac{300^2}{4r^2} \frac{1}{\max\{x^2 \mid -150 \leq x \leq 150\}},$$

and therefore the number of partitions of each interval to fall below a given tolerance ε_{rel} is given by

$$r = \left\lceil \frac{1}{\sqrt{\varepsilon_{rel}}} \right\rceil.$$

Thus, if we require an relative relaxation error $\varepsilon_{rel} = 0.01$ we have to partition the corresponding intervals into at least $r = 10$ partitions. Note that even if we just require a relative relaxation error $\varepsilon_{rel} = 0.03$ we still have to partition into at least $r = 6$ intervals.

In sum, this yields that – if we use r equidistant partitions for any variable appearing in any nonlinear term of our problem – we fall below a relaxation error tolerance of $\varepsilon_{rel} = 0.01$ as soon as we use a relaxation $\mathcal{R}^r$ with $r \geq 10$. Note that if we did not use the adaptive approach given in Algorithm 6, but choosing a relaxation with $r \geq 10$ and then using a method for solving multiobjective linear mixed-integer problems we would have to solve a problem with at least $10 \cdot |\text{Edges}|$ additional integer variables, i.e. in the case of `network 3` about 400 if we just use the ones for the quadratic terms. Looking at the results on the specific networks (see Figure 9 to Figure 11) we can see that the method uses only the relaxation $\mathcal{R}^r$ with $r = 1$, i.e. the coarsest relaxation possible using the McCormick relaxations. This shows the power of the proposed method dealing with the considered large network instances.

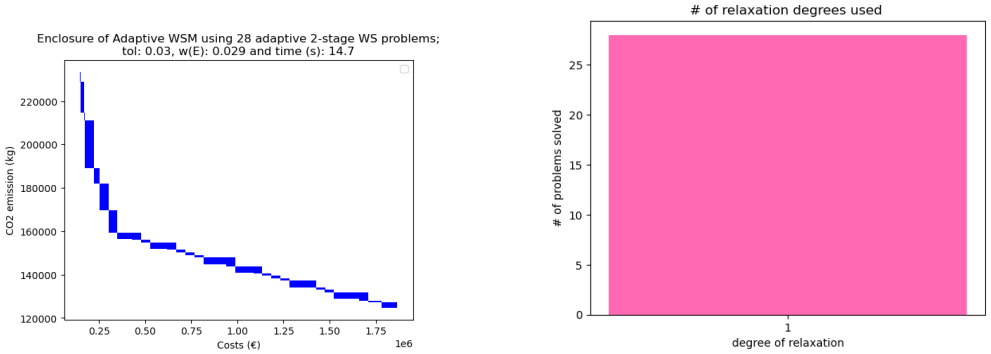

Figure 9: Computational results on `network 1`. **Left:** enclosure given by LLBs and LUBs of (Circles) obtained by Algorithm 6 with WSM. **Right:** counter of used degrees of relaxations.

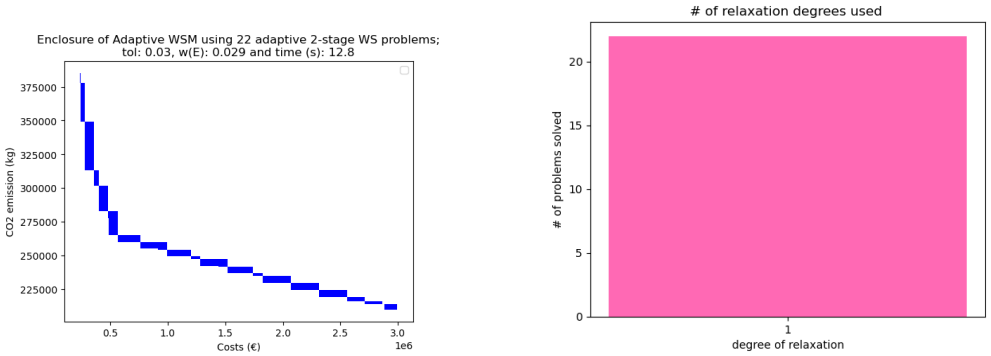

Figure 10: Computational results on `network 2`. **Left:** enclosure given by LLBs and LUBs of (Circles) obtained by Algorithm 6 with WSM. **Right:** counter of used degrees of relaxations.

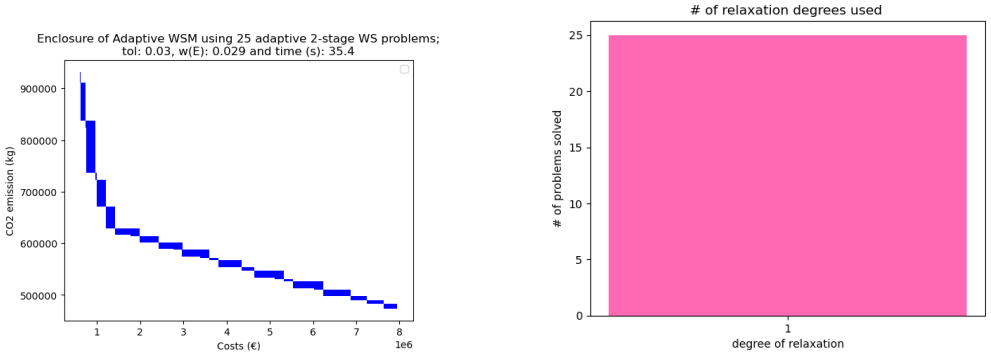

Figure 11: Computational results on `network 3`. **Left:** enclosure given by LLBs and LUBs of (Circles) obtained by Algorithm 6 with WSM. **Right:** counter of used degrees of relaxations.

7 Conclusion

In the present work a general MINLP problem is considered and two novel methods for computing an enclosure of the nondominated set are presented. For both of them we proved correct and finite termination as well as demonstrated the respective advantages and disadvantages. The implementation of the second approach is currently only able to deal with bilinear and quadratic terms. However, one could handle general polynomial terms still relying on McCormick relaxations. For general nonlinear terms one has to go for a more elaborate relaxation technique as presented in, e.g., [7]. However, these are only implementation issues. As long as the relaxation technique satisfies Assumption 5.1 the theoretical results presented in this paper still apply.

References

[1] Stefan Banholzer. *ROM-Based Multiobjective Optimization with PDE Constraints*. PhD thesis, Universität Konstanz, Konstanz, 2021.

[2] Stefan Banholzer, Bennet Gebken, Michael Dellnitz, Sebastian Peitz, and Stefan Volkwein. ROM-based multiobjective optimization of elliptic PDEs via numerical continuation. In *Non-smooth and complementarity-based distributed parameter systems—simulation and hierarchical optimization*, volume 172 of *Internat. Ser. Numer. Math.*, pages 43–76. Birkhäuser/Springer, Cham, 2022.

[3] Pietro Belotti, Christian Kirches, Sven Leyffer, Jeff Linderoth, James Luedtke, and Ashutosh Mahajan. Mixed-integer nonlinear optimization. *Acta Numer.*, 22:1–131, 2013.

[4] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. ZIB-Report 21-41, Zuse Institute Berlin, December 2021.

[5] Conrado Borraz-Sánchez, Russell Bent, Scott Backhaus, Hassan Hijazi, and Pascal Van Hentenryck. Convex relaxations for gas expansion planning. *INFORMS J. Comput.*, 28(4):645–656, 2016.

[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. *Engineering Optimization*, 0(0):1–13, 2021.

[7] Robert Burlacu. *Adaptive Mixed-Integer Refinements for Solving Nonlinear Problems with Discrete Decisions*. PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2019.

[8] Robert Burlacu. On refinement strategies for solving minlps by piecewise linear relaxations: a general red refinement. *Optimization Letters*, jun 2021.

[9] Robert Burlacu, Björn Geiß ler, and Lars Schewe. Solving mixed-integer nonlinear programmes using adaptively refined mixed-integer linear programmes. *Optim. Methods Softw.*, 35(1):37–64, 2020.

[10] IBM ILOG Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[11] Kerstin Dächert, Kathrin Klamroth, Renaud Lacour, and Daniel Vanderpooten. Efficient computation of the search region in multi-objective optimization. *European J. Oper. Res.*, 260(3):841–855, 2017.

[12] Marianna De Santis, Gabriele Eichfelder, Julia Niebling, and Stefan Rocktäschel. Solving multiobjective mixed integer convex optimization problems. *SIAM J. Optim.*, 30(4):3122–3145, 2020.

[13] Erik Diessel. An adaptive patch approximation algorithm for bicriteria convex mixed-integer problems. *Optimization*, 0(0):1–46, 2021.

[14] Matthias Ehrgott. *Multicriteria optimization.* Springer-Verlag, Berlin, second edition, 2005.

[15] Matthias Ehrgott and Xavier Gandibleux. Bound sets for biobjective combinatorial optimization problems. *Comput. Oper. Res.*, 34(9):2674–2694, 2007.

[16] Gabriele Eichfelder, Peter Kirst, Laura Meng, and Oliver Stein. A general branch-and-bound framework for continuous global multiobjective optimization. *J. Global Optim.*, 80(1):195–227, 2021.

[17] Gabriele Eichfelder, Oliver Stein, and Leo Warnow. A deterministic solver for multiobjective mixed-integer convex and nonconvex optimization. 2022.

[18] Gabriele Eichfelder and Leo Warnow. An approximation algorithm for multiobjective optimization problems using a box-coverage. *J. Global Optim.*, 2021.

[19] Gabriele Eichfelder and Leo Warnow. A hybrid patch decomposition approach to compute an enclosure for multiobjective mixed-integer convex optimization problems. 2021.

[20] Björn Geißler, Alexander Martin, Antonio Morsi, and Lars Schewe. Using piecewise linear functions for solving MINLPs. In *Mixed integer nonlinear programming*, volume 154 of *IMA Vol. Math. Appl.*, pages 287–314. Springer, New York, 2012.

[21] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.

[22] Laura Iapichino, Stefan Trenz, and Stefan Volkwein. Reduced-order multiobjective optimal control of semilinear parabolic problems. In *Numerical mathematics and advanced applications—ENUMATH 2015*, volume 112 of *Lect. Notes Comput. Sci. Eng.*, pages 389–397. Springer, [Cham], 2016.

[23] Il Y. Kim and Oliver de Weck. Adaptive weighted sum method for bi-objective optimization: Pareto front generation. *Struct. Multidiscip. Optim.*, 29:149–158, 2005.

[24] Il Y. Kim and Oliver de Weck. Adaptive weighted sum method for multiobjective optimization: a new method for Pareto front generation. *Struct. Multidiscip. Optim.*, 31(2):105–116, 2006.

[25] Gokhan Kirlik and Serpil Sayın. Bilevel programming for generating discrete representations in multiobjective optimization. *Math. Program.*, 169(2, Ser. A):585–604, 2018.

[26] Kathrin Klamroth, Renaud Lacour, and Daniel Vanderpooten. On the representation of the search region in multi-objective optimization. *European J. Oper. Res.*, 245(3):767–778, 2015.

[27] Jianjie Lu. *Mixed-Integer Nonlinear Modeling and Optimization of Designing Decentralized Energy Supply Networks*. PhD thesis, Universität Konstanz, 2021.

[28] Andreas Lundell, Anders Skjäl, and Tapio Westerlund. A reformulation framework for global optimization. *J. Global Optim.*, 57(1):115–141, 2013.

[29] Stephen Maher, Matthias Miltenberger, João P. Pedroso, Daniel Rehfeldt, Robert Schwarz, and Felipe Serrano. PySCIPOpt: Mathematical programming in python with the SCIP optimization suite. In *Mathematical Software – ICMS 2016*, pages 301–307. Springer International Publishing, 2016.

[30] Garth P. McCormick. Computability of global solutions to factorable nonconvex programs. I. Convex underestimating problems. *Math. Programming*, 10(2):147–175, 1976.

[31] Antonio Morsi, Björn Geiß ler, and Alexander Martin. Mixed integer optimization of water supply networks. In *Mathematical optimization of water networks*, volume 162 of *Internat. Ser. Numer. Math.*, pages 35–54. Birkhäuser/Springer Basel AG, Basel, 2012.

[32] Harsha Nagarajan, Mowen Lu, Site Wang, Russell Bent, and Kaarthik Sundar. An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs. *J. Global Optim.*, 74(4):639–675, 2019.

[33] Adriano Pascoletti and Paolo Serafini. Scalarizing vector optimization problems. *J. Optim. Theory Appl.*, 42(4):499–524, 1984.

[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):16–39, 2020.

[35] Namhee Ryu and Seungjae Min. Multiobjective optimization with an adaptive weight determination scheme using the concept of hyperplane. *Internat. J. Numer. Methods Engrg.*, 118(6):303–319, 2019.

[36] Serpil Sayın. Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Math. Program.*, 87(3, Ser. A):543–560, 2000.

[37] Thomas Stidsen and Kim Allan Andersen. A hybrid approach for biobjective optimization. *Discrete Optim.*, 28:89–114, 2018.

[38] Juan Pablo Vielma, Shabbir Ahmed, and George Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: unifying framework and extensions. *Oper. Res.*, 58(2):303–315, 2010.