ECONSTOR
Make Your Publications Visible.

A Service of

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Link, Moritz; Volkwein, Stefan

Article — Published Version

Adaptive piecewise linear relaxations for enclosure computations for nonconvex multiobjective mixed-integer quadratically constrained programs

Journal of Global Optimization

Provided in Cooperation with:
Springer Nature

Suggested Citation: Link, Moritz; Volkwein, Stefan (2023) : Adaptive piecewise linear relaxations for enclosure computations for nonconvex multiobjective mixed-integer quadratically constrained programs, Journal of Global Optimization, ISSN 1573-2916, Springer US, New York, NY, Vol. 87, Iss. 1, pp. 97-132, https://doi.org/10.1007/s10898-023-01309-5

This Version is available at:
https://hdl.handle.net/10419/307034

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

https://creativecommons.org/licenses/by/4.0/

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

Journal of Global Optimization (2023) 87:97–132
https://doi.org/10.1007/s10898-023-01309-5

Adaptive piecewise linear relaxations for enclosure computations for nonconvex multiobjective mixed-integer quadratically constrained programs

Moritz Link[1] · **Stefan Volkwein**[1]

Received: 29 August 2022 / Accepted: 20 June 2023 / Published online: 5 July 2023
© The Author(s) 2023

Abstract
In this paper, a new method for computing an enclosure of the nondominated set of multiobjective mixed-integer quadratically constrained programs 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

Mathematics Subject Classification 65K05 · 90C11 · 90C26 · 90C29 · 90C35

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 that 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 only one objective is of interest. This is also the case in our application, where we are looking for energy supply network plans which

✉ Moritz Link
moritz.link@uni-konstanz.de

Stefan Volkwein
stefan.volkwein@uni-konstanz.de

[1] Department of Mathematics and Statistics, University of Konstanz, Universitätsstraße 10, 78464 Constance, Baden-Württemberg, Germany

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. [23–26]) or an approximation (cf. [5, 10, 17, 30, 33]) 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., [47]). 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 also returns 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 2) tackles the problem more directly without being too complicated. In fact, it is a slight modification of the method presented in [23] in the way that we use the Restricted Weighted Sum Method (cf. [31, 32]) 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 programming (MILP) problems have to be solved, where powerful solvers are available (cf. [14, 29]). The accuracy of these relaxations is chosen adaptively by the method itself during the solution process. Consequently, it is possible to use different levels of accuracy in different parts of the criterion space, which can be of importance if also the relaxed problems are difficult to solve.

This approach is the first of its kind for a (MOP). In [24] 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 that do not belong to the image of the feasible set. In [25] 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 from the methods presented in this paper. Note further that due to clarity and conciseness, we restrict ourselves to nonconvex quadratically constrained problems in this paper as this is of relevance to our application. However, the presented method can be straightforwardly generalized to polynomially constrained programs using reduction techniques to decompose polynomials into quadratic and bilinear terms (cf., e.g., [45] and Remark 4.3).

The manuscript is organized as follows: After introducing preliminary notions in Sect. 2 we define enclosures and related notions as well as provide a basic approach for computing such an enclosure in Sect. 3. In Sect. 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 Sect. 5. We further prove correct and finite termination of this procedure. In Sect. 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 nonconvex multiobjective mixed-integer quadratically constrained problems. Finally, a conclusion is drawn in Sect. 7.

2 Notations and definitions

Throughout we write $x \leq y$ for $x, y \in \mathbb{R}^k$ if for any $i \in [k] := \{1, \ldots, k\}$ 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 $f_i : \mathbb{R}^{n+m} \to \mathbb{R}, i \in [k]$, of the objective function $f = (f_1, \ldots, f_k)^\top : \mathbb{R}^{n+m} \to \mathbb{R}^k$, the continuous components $h_i : \mathbb{R}^{n+m} \to \mathbb{R}$, $i \in [p]$, of the equality constraint function $h = (h_1, \ldots, h_p)^\top : \mathbb{R}^{n+m} \to \mathbb{R}^p$ and the continuous components $g_i : \mathbb{R}^{n+m} \to \mathbb{R}$, $i \in [q]$, of the inequality constraint function $g = (g_1, \ldots, g_q)^\top : \mathbb{R}^{n+m} \to \mathbb{R}^q$. For all of these functions, we only allow quadratic and bilinear terms. The corresponding multiobjective optimization problem is then given by

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

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. For a survey on solution methods for general versions of (MOP) we refer the reader to [22]. Single objective versions of (MOP) we denote by MIQCP (or more general MINLP) problems. Interested readers are referred to [1, 6, 9, 36, 43, 52].

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

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

which we assume to be nonempty, as well as its possibly nonconvex 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

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

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

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

describing the feasible and box-feasible solutions corresponding to $\hat{x}_I$, respectively. Notice that $S_{\hat{x}_I} = \emptyset$ holds, provided that $\hat{x}_I \in \mathbb{Z}^m \backslash S_I$. 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 $B = [z^\ell, z^u] \subseteq \mathbb{R}^k$ with $f(S) \subseteq \text{int}(B)$ for some $z^\ell, z^u \in \mathbb{R}^k$, $z^\ell \leq z^u$, $z^\ell \neq z^u$.

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. [20]).

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

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

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

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

(2) Let $y \in \mathbb{R}^k$ and $N \subset \mathbb{R}^k$. 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. [16, 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. [49]):

- **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 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., [23–26]. 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 Sect. 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-non-do-mi-na-ted 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 Sects. 3 and 6.

3 Enclosures and local bounds

In [23, 24, 26] 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 $\mathcal{N} \subseteq L + \mathbb{R}^k_+$ 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 [23–26] 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 $s(\ell, u) := \min\{u_i - \ell_i \mid i \in [k]\}$ represents the shortest edge length of a given box $[\ell, u]$. The justification for the choice of this width measure is given by the following lemma.

Lemma 3.2 (cf. [26, 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 [23, 24], whereas in the branch-and-bound based method from [26] 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 [23–25]) of the concept of Local Lower and Local

Upper Bounds (LLBs and LUBs, respectively) which is an extension of the work in [15, 34] and also [21].

In [15] 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 $y_i^1 < y_i^2$ and $y_j^2 < y_j^1$. One can immediately see that for any (MOP) the corresponding nondominated set $\mathcal{N}$ is stable. We start with the definition of local upper bounds as introduced in [34].

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

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

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

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

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

(i) $s(N) = \bigcup_{u \in U(N)} c(u)$,
(ii) $c\left(u^1\right) \not\subseteq c\left(u^2\right)$ 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 [23, 24] the authors extended the concept of LUBs to 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

$$S(N) := \left\{ y \in \operatorname{int}(B) \mid y' \not\geq y \text{ for every } y' \in N \right\},$$

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

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

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) $C\left(\ell^1\right) \not\subseteq C\left(\ell^2\right)$ 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 [26] 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 [23] relates the concept of local lower/upper bounds to lower/upper bounds as introduced in Definition 3.1.

Lemma 3.5 (cf. [23, 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 [34, Algorithm 3], [23, Algorithm 2]. In Algorithm 1 we provide the scheme for updating a local upper bound set. Similar to [34] we use the following notation: for $y \in \mathbb{R}^k$, $c \in \mathbb{R}$ and $i \in [k]$ we define

$$y_{-i} := (y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_k)^\top \text{ and}$$

Algorithm 1 Updating a Local Upper Bound Set

Require: Local upper bound set $U(N)$ and update point $y \in f(S)$
1: $A = \{u \in U(N) \mid y < u\}$
2: **for** $i \in [k]$ **do**
3: $\quad B_i = \left\{u \in U(N) \mid y_i = u_i \text{ and } y_{-i} < u_{-i}\right\}$
4: $\quad P_i = \emptyset$
5: **end for**
6: **for** $i \in [k]$ **do**
7: $\quad$ **for** $u \in A$ **do**
8: $\quad\quad P_i = P_i \cup \left\{\left(y_i, u_{-i}\right)\right\}$
9: $\quad$ **end for**
10: **end for**
11: **for** $i \in [k]$ **do**
12: $\quad P_i = \left\{u \in P_i \mid u \nleq u' \text{ for all } u' \in P_i \cup B_i,\ u' \neq u\right\}$
13: **end for**
14: $U(N \cup \{y\}) = (U(N) \setminus A) \cup \bigcup_{i \in [k]} P_i$
return Updated local upper bound set $U(N \cup \{y\})$

$$(c, y_{-i}) := (y_1, \ldots, y_{i-1}, c, y_{i+1}, \ldots, y_k)^\top.$$

In order to obtain an algorithm for updating a local lower bound set $L(N)$ w.r.t. an update point $y \in \operatorname{int}(B)$, one can modify Algorithm 1 in the following way. We replace $U(N)$ by $L(N)$, and change the following:

- Step 1 to $A = \{\ell \in L(N) \mid y > \ell\}$,
- Step 3 to $B_i = \{\ell \in L(N) \mid y_i = \ell_i \text{ and } y_{-i} > \ell_{-i}\}$,
- Step 12 to $P_i = \left\{\ell \in P_i \mid \ell \ngeq \ell' \text{ for all } \ell' \in P_i \cup B_i,\ \ell' \neq \ell\right\}$.

Furthermore, by analyzing the update procedure of $U(N)$ w.r.t. a point $y \in f(S)$, 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 [23, Lemma 3.8] *Let $u \in U(N \cup \{y\})$ be a local upper bound. Then its parent $u' \in U(N)$ is unique.*

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 [15, 34] the authors present a general scheme for computing an approximation of the nondominated set of a multiobjective optimization problem (cf., e.g., [15, Algorithm 2]). In [23] 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.

For this paper, we use a slightly modified approach of these as a basic scheme which is written in Algorithm 2.

In Algorithm 2, we loop through the set U_{loop}, i.e. the set of local upper bounds at the beginning of the iteration. If the current local upper bound u is part of a not sufficiently small box (see Step 5), we try to improve this local upper bound, i.e. find a nondominated point in the search region determined by u. In order to do so, we solve the weighted-sum problem

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

Algorithm 2 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 $f(S) \subseteq \text{int}(B)$, termination tolerance $\varepsilon > 0$ and off-set factor $\delta > 0$

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

for some weight vector $\alpha \in \text{int}(\mathbb{R}^k_+)$. 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. [4, Lemma 1.5.2],[20, p. 214f]), i.e. $\bar{y} = f(\bar{x}) \in \mathcal{N}$ and $\bar{y} < u - \delta e$. Furthermore, we 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 [48]. 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 2 can be seen in the context of the so-called *Adaptive Weighted Sum Method* as presented in [31, 32].

Remark 3.7 For a clear definition of defining points we refer to [34]. Furthermore, in [34] 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 [34, Algorithm 5]). In fact, [34, Algorithm 5] is used in any implementation of the present paper instead of Algorithm 1. We waived to present [34, Algorithm 5] in detail to keep things more concise.

If (WSP $(\alpha; u)$) is feasible, it is solved to optimality and its solution $\bar{y} \in \mathcal{N}$ is used to update the local upper and lower bound sets. Afterward, it proceeds to the next iteration. If

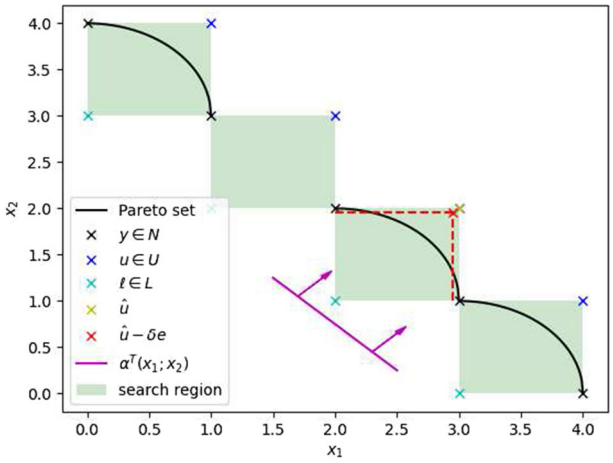

Fig. 1 Exemplary configuration during Algorithm 2 tackling an instance of Example 3.14. $\hat{u} \in U$ denotes the current local upper bound of interest. The moving hyperplane $\sigma^T(x_1; x_2)$ together with the red dashed line mimics the behavior of the (WSP $(\alpha; u)$)

otherwise (WSP $(\alpha; u)$) is not feasible, we know that there is no nondominated point y of (MOP) satisfying $y < u - \delta e$ and therefore particularly $y < u - \varepsilon e$. Therefore, we can use $u - \delta e$ to update the set of local lower bounds (see Step 10).

An exemplary situation during Algorithm 2 is depicted in Fig. 1, where $\hat{u} \in U_{\text{loop}}$ denotes the current local upper bound of interest. The green boxes represent the current enclosure based on the set of local upper bounds U and local lower bounds L, which depend on the current approximation N of the Pareto set $\mathcal{N}$. Note that the enclosure simultaneously functions as the leftover search zone, i.e. the area where nondominated points can lie. The dashed red line mimics the restrictions coming from (WSP $(\alpha; u)$), i.e. we are only looking for nondominated points in the lower left quadrant w.r.t. $\hat{u} - \delta e$. Clearly, if in this area there is no part of the Pareto set, we can update the lower bound set w.r.t. $\hat{u} - \delta e$ and declare the search region determined by $\hat{u}$ as well-enough explored. Furthermore, in that picture one can imagine 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.

Remark 3.8 We briefly describe the method from [23] as it is closely related to ours from Algorithm 2.

The method from [23] iterates through the elements of the list of local lower bounds L in an outer loop and through the elements of the list of local upper bounds 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,

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

Problem (PSP (ℓ, u)) can be interpreted as Pascoletti-Serafini problem (cf. [46]) 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. [46]). If $\bar{y}$ satisfies some specific criterion (cf. [23, 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. We decided to avoid solving problems like (PSP (ℓ, u)) since preliminary numerical experiments pointed in the direction that solving (PSP (ℓ, u)) is computationally not as efficient as solving (WSP $(\alpha; u)$) for our test instances. Furthermore, by doing so, one avoids the procedure for ensuring strict nondominance of the update points.

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

$$\left(u^{\text{new}} - \ell^{\text{new}}\right)_i \leq \max\left\{\varepsilon,\ 0.5\left(u^{\text{old}} - \ell^{\text{old}}\right)_i\right\} \tag{3}$$

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. [23, 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. [23, Lemma 4.3] and [23, Theorem 4.5]). Due to the modifications in our method, we obtain the decrease estimate for the boxes (4) which differs from (3).

We proceed with proving correctness and finiteness of Algorithm 2.

Lemma 3.9 *Let U be the local upper bound set at some arbitrary point in Algorithm 2. Then U is an upper bound set in the sense of Definition 3.1.*

Proof We initialize the set $U = \{z^u\}$. During Algorithm 2 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 3.10 *Let L be the local lower bound set at some arbitrary point in Algorithm 2. Then L is a lower bound set in the sense of Definition 3.1.*

Proof We initialize the set $L = \{z^\ell\}$. During Algorithm 2 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 $y = \hat{u} - \delta e \notin f(S) + \text{int}(\mathbb{R}^k_+)$. Hence, by correctness of the modified version of Algorithm 1 for local lower bounds we obtain that L is a local lower bound set w.r.t. some set $N_2 \subseteq \text{int}(B)\backslash(f(S) + \text{int}(\mathbb{R}^k_+))$. 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 2. 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 $N_2 \subseteq \text{int}(B)\backslash(f(S) + \text{int}(\mathbb{R}^k_+))$ are finite. If we show that Algorithm 2 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 2 we start with a 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 [23]. However, as some adaptions have to be made, we present the adapted proofs in detail.

Theorem 3.11 *Let $\varepsilon > \delta > 0$ and $z^\ell, z^u \in \mathbb{R}^k$ be the input parameters of Algorithm 2. Moreover, let L^{start} and U^{start} be the local lower and upper bound sets at the beginning of some iteration in Algorithm 2, 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*

$$\left(u^e - \ell^e\right)_j < \max\left\{\left(u^s - \ell^s\right)_j - \delta, \varepsilon\right\}. \tag{4}$$

Proof Let $\ell^e \in L^{\text{end}}$ and $u^e \in U^{\text{end}}$ with $\ell^e \leq u^e$. 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

$$\left|P\left(\ell^e\right) \cap L\right| = 1 \text{ and } \left|P\left(u^e\right) \cap U\right| = 1 \tag{5}$$

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 1 and its adaption for local lower bounds, 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

$$\ell^s \leq \ell \leq \ell^e \text{ for all } \ell \in P\left(\ell^e\right),\ u^e \leq u \leq u^s \text{ for all } u \in P\left(u^e\right), \tag{6}$$

and therefore (1) is satisfied.

We now 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

$$\left(u^e - \ell^e\right)_i \geq \max\left\{(u - \ell)_i - \delta, \varepsilon\right\}. \tag{7}$$

In particular, (7) 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 2, 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

$$s\left(\ell', u^s\right) \geq \varepsilon \tag{8}$$

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 (5). Say $u^j \in U_{\text{updated}}$. Using (6), we compute

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

a contradiction to (7).

Case A.2 $y \not< u'$. Then there exists $j \in [k]$ with $u'_j \leq y_j$. Again, using (6), we compute

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

a contradiction to (7).

(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 1 for local lower bounds, 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 (5). Say $\ell^j \in L_{\text{updated}}$. Using (6), we compute

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

a contradiction to (7).

Case B.2 $\ell' \not< y$. Then there exists $j \in [k]$ with $\ell'_j \geq y_j = u^s_j - \delta$, i.e. particularly $s(\ell', u^s) \leq \delta < \varepsilon$, a contradiction to (8).

Obtaining a contradiction in all possible cases shows that our assumption (7) cannot be true and therefore statement (2) is also true, which completes the proof. □

Knowing that in every iteration of Algorithm 2 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 2 terminates after finitely many iterations.

Theorem 3.12 *Let $\varepsilon > \delta > 0$ and $z^\ell, z^u \in \mathbb{R}^k$ be the input parameters of Algorithm 2. We define*

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

Then the number of iterations of Algorithm 2, i.e. the number of iterations of the `while`-loop, is bounded by $\max\{1, \kappa\}$. *Hence, Algorithm 2 is finite.*

Proof If $\kappa \leq 1$, then $\Delta \leq \varepsilon$ and Algorithm 2 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 3.11, 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

$$\left(u^l - \ell^l\right)_{i^l} < \max\left\{\left(u^{l-1} - \ell^{l-1}\right)_{i^l} - \delta, \varepsilon\right\}. \tag{9}$$

Suppose now that Algorithm 2 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

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

As $\kappa > k\lceil\frac{\Delta-\varepsilon}{\delta}\rceil$ there exists at least one index $j \in [k]$ with $n(j) \geq \lceil\frac{\Delta-\varepsilon}{\delta}\rceil + 1$. Iterative use of (9) yields that

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

a contradiction to $s(\ell^\kappa, u^\kappa) \geq \varepsilon$. Hence, we have that $w(E_\kappa) < \varepsilon$ and Algorithm 2 terminates. □

Corollary 3.13 *Let $\varepsilon > \delta > 0$ and $z^\ell, z^u \in \mathbb{R}^k$ be the input parameters of Algorithm 2. 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 3.12 tells us that Algorithm 2 terminates after at most κ calls of the `while`-loop, i.e. the output set E_κ satisfies $w(E_\kappa) < \varepsilon$. By Lemma 3.9 and Lemma 3.10 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 2, 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 performing 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 3.12 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 3.14 We consider the optimization problem:

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

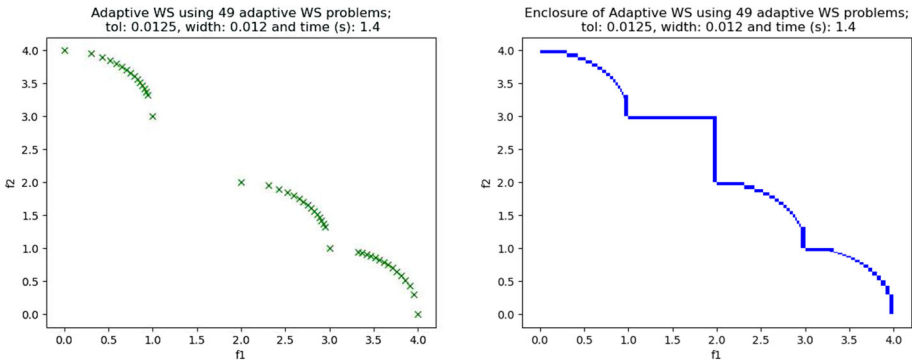

Fig. 2 Computational results on first instance in Example 3.14. Left: nondominated points of (Circles) obtained by Algorithm 2 with WSM. Right: enclosure given by the LLBs and LUBs

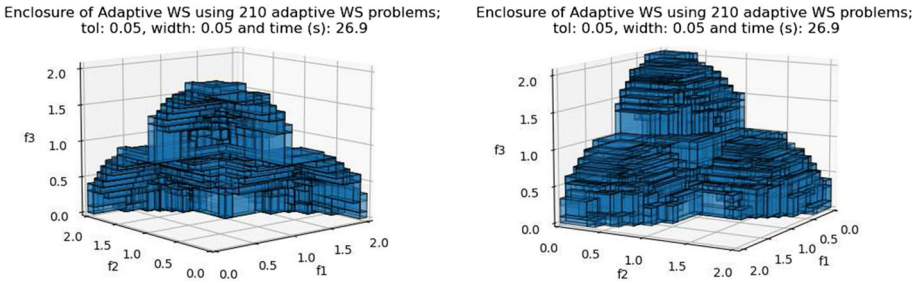

Fig. 3 Computational results on second instance in Example 3.14. Two different viewpoints of the enclosure computed by Algorithm 2 with WSM

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 2 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 Fig. 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 2 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 Fig. 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. [41]) of the SCIP optimization suite (cf. [7]) and are carried out on a machine with Intel Core i7-8565U processor and 32GB of RAM. In Fig. 2 (Right) and Fig. 3 the enclosure obtained by the method is depicted. One can observe a significant increase in computational time going from two dimensions to three. In Fig. 2 (Left) the approximation of the nondominated set is depicted.

4 Piecewise linear relaxations and lower bounding

As we have seen, using Algorithm 2 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 2 are still MINLP problems in general, there may arise inconvenience while tackling, e.g., larger instances (see Sect. 6). One idea to cope with this issue is to bypass the non-linearity of the problems, e.g., trying to solve just MILP problems.

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 from single objective optimization in regard to solving MINLP problems. In [13, 40, 45] several methods for solving MINLP problems, 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., [6, 12, 13, 28, 38, 44, 53]). 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 [45] on the one hand and [13, 40] 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 problem 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 [45] 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 problem 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 problem the resulting NLP problem 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 improved 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 problem 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 at 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 [13] to preserve the theoretically ensured convergence of the method, but also add the upper bounding part as well as the gap termination criterion from

[45] to avoid unnecessary computations. Furthermore, the relaxation techniques used in [13] are able to handle general nonlinear functions, whereas the techniques in [45] are based on the so-called McCormick relaxations for bilinear and multilinear terms (cf. [42]) 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 Sect. 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 by 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 Sect. 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., [11, 13, 28]).

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 $\mathcal{R}_h = (h_u, h_o)$, 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

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

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 [11], we measure the quality of a given piecewise linear relaxation $\mathcal{R}_h$ of a nonlinear term h by

- the *overestimation error*

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

- the *underestimation error*

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

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

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

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

Similar for the *underestimation error*. The *overall relaxation error* is then given by $\varepsilon_{rel}^{\mathcal{R}} := \max\{\varepsilon_o^{\mathcal{R}}, \varepsilon_u^{\mathcal{R}}\}$. 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. [42]).

Remark 4.3 As already mentioned, we put the focus in this paper on the ingredients necessary for our application. As there appear only bilinear and quadratic constraints, we restrict ourselves to the description of exactly this case. However, by decomposing general polynomials into bilinear and quadratic terms one can apply the method from this work to general polynomially constrained problems. For example, the polynomial x^3y can be decomposed into

$$\begin{aligned} x_1 &= x^2, \\ x_2 &= xy, \\ x_3 &= x_1x_2, \end{aligned}$$

where x_i, $i = 1, 2, 3$, are additional variables.

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 $i \in \{0, \ldots, r\}$ with corresponding intervals $[x_{i-1}, x_i]$ and $[y_{i-1}, y_i]$ for $i \in [r]$.

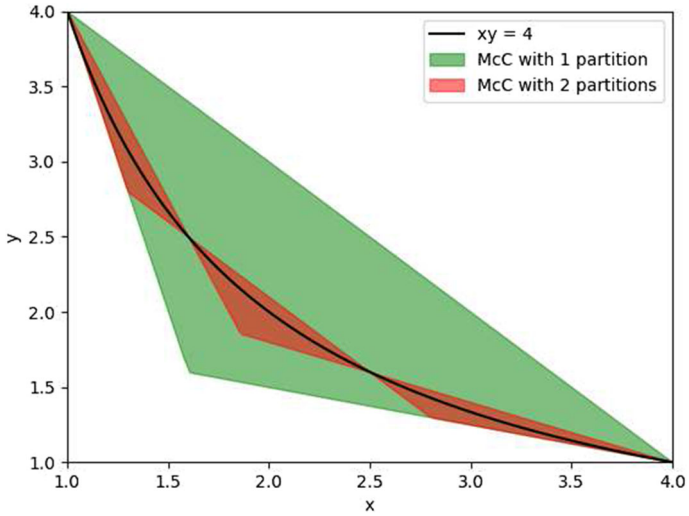

Fig. 4 McCormick relaxations of the bilinear term $x\,y$ using one partition per variable (green) and two partitions per variable (red). (Color figure online)

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 $D_h^{ij} := [x_{i-1}, x_i] \times [y_{j-1}, y_j]$ is given by

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

and the linear overestimator is given by

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

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

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

and linear overestimator is given by

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

The resulting relaxation error corresponding to r equidistant partitions is

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

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 $\varepsilon_{rel}^{(h,\mathcal{R}_h^r)} < \varepsilon$.

Remark 4.4 In [45] the authors provide an adaptive partitioning scheme, i.e. there is no requirement for equidistant intervals. New breakpoints are added in parts of the variable domains close to the value of the previous solution in order to refine the relaxation. The idea is to only partition on regions of the variable domain that appear to influence optimality the most. In contrast, using uniformly distributed break points, one runs the risk of refining in uninteresting regions of the variable domain and therefore unnecessarily blowing up the problem. The scheme presented in [45] could also be incorporated[2] into the methods of the present paper. But for simplicity of presentation and since the equidistant procedure works very well for our application, we restrict ourselves to the equidistant partitioning scheme presented in this paper.

After we have introduced the relaxations of interest for the present 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 $f(\tilde{S}) \subseteq \text{int}(B)$. We obtain the following lemma.

Lemma 4.5 *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. $f(\tilde{x}) \in \text{int}(B) \backslash (f(S) + \text{int}(\mathbb{R}_+^k))$.*

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 [45], 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 3.

[2] Note that the argument showing that the refinement technique presented in this paper satisfies Assumption 5.1 heavily relies on the equidistant partitioning scheme.

Algorithm 3 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: Set $N = N \setminus \left\{y' \in N \mid y \leq y'\right\}$
2: **if** y is nondominated w.r.t. N **then**
3: Set $N = N \cup \{y\}$
4: **end if**
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 4.6 *Let N_1 be a stable input set and let $y \in f(S)$. Then Algorithm 3 returns a stable set N_2. Furthermore, either $N_1 \subseteq N_2$ or there exists $y' \in N_1$ such that y dominates y'.*

Proof We have to distinguish two base cases. Firstly, assume there exists no $y' \in N_1$ with $y \leq y'$. Then either y is not nondominated w.r.t. N_1 and we have that $N_1 = N_2$, or y is nondominated w.r.t. N_1 and we have that $N_1 \subset N_2 = N_1 \cup \{y\}$. In both cases, we have that N_2 is stable.

Secondly, assume that there exist $y^i \in N_1$ with $y \leq y^i$, $i \in [s]$ for some $s \in \mathbb{N}$. Due to the stability of N_1 we have that y is nondominated w.r.t. $N_1 \backslash \{y^i \mid i \in [s]\}$. Hence, $N_2 = (N_1 \backslash \{y^i \mid i \in [s]\}) \cup \{y\}$ and N_2 is stable. Further, if $y = y^1$ we have that $N_1 = N_2$. If otherwise, we have that y dominates y^1. □

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 $\hat{y}$ dominates $\tilde{y}$. Note that if y' dominates $\tilde{y}$ 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 4.

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

We obtain the analogue to Lemma 4.6. Note that by Lemma 4.5 we know that $\tilde{N}$ is nondominated w.r.t. $\mathcal{N}$, i.e. $\tilde{N}$ actually approximates $\mathcal{N}$ from below.

Lemma 4.7 *Let $\tilde{N}_1$ be a stable input set and let $y \in \operatorname{int}(B) \backslash (f(S) + \operatorname{int}(\mathbb{R}^k_+))$. Then Algorithm 4 returns a stable set $\tilde{N}_2$. Furthermore, either $\tilde{N}_1 \subseteq \tilde{N}_2$ or there exists $\tilde{y} \in \tilde{N}_1$ such that $\tilde{y}$ dominates y.*

Proof We have to distinguish two base cases. Firstly, assume there exists no $\tilde{y} \in \tilde{N}_1$ with $\tilde{y} \leq y$. Then either $\tilde{N}_1$ is not nondominated w.r.t. y and we have that $\tilde{N}_1 = \tilde{N}_2$, or $\tilde{N}_1$ is nondominated w.r.t. y and we have that $\tilde{N}_1 \subset \tilde{N}_2 = \tilde{N}_1 \cup \{y\}$. In both cases, we have that $\tilde{N}_2$ is stable.

Secondly, assume that there exist $\tilde{y}^i \in \tilde{N}_1$ with $\tilde{y}^i \leq y$, $i \in [s]$ for some $s \in \mathbb{N}$. Due to the stability of $\tilde{N}_1$ we have that $\tilde{N}_1 \backslash \{\tilde{y}^i \mid i \in [s]\}$ is nondominated w.r.t. y. Hence, $\tilde{N}_2 = (\tilde{N}_1 \backslash \{\tilde{y}^i \mid i \in [s]\}) \cup \{y\}$ and $\tilde{N}_2$ is stable. Further, if $y = \tilde{y}^1$ we have that $\tilde{N}_1 = \tilde{N}_2$. If otherwise, we have that $\tilde{y}^1$ dominates y. □

5 General scheme

We have seen at the end of Sect. 3 that Algorithm 2 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 2 are MINLP problems. At the end of Sect. 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 MINLP problems increases, the run time of solvers like SCIP for computing a solution to such an MINLP problem 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 Sect. 4 we have presented ideas and concepts from single objective optimization of MINLP problems 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., [47, 51] for the biobjective linear case and [24] for the multiobjective convex case, or even Algorithm 2 or [23]. 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 [13] 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. Note that one could additionally incorporate adaptivity in the variable domains into the refinement procedure (see Remark 4.4).

In the following, we present an algorithm similar to Algorithm 2 which makes use of these ideas in order to compute an enclosure of the nondominated set without solving scalarized MINLP problems, but only MILP and NLP problems (see Algorithm 5).

Algorithm 5 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 $f(S) \subseteq \text{int}(B)$, termination tolerance $\varepsilon_{\text{encl}} > 0$, off-set 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 $\mathcal{R}_{\text{current}} = \min\left\{\mathcal{R}^u, \mathcal{R}'\right\}$, where $\left(u, \mathcal{R}^u\right) \in \mathcal{D}(U)$ 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. 4 and 1 for local lower bounds 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. 3 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. 3 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 ??
          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 $\mathcal{N}_{\varepsilon_{\text{encl}}}$

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

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

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 $\mathcal{R}^I \supsetneq \mathcal{R}^1 \supsetneq \mathcal{R}^2 \supsetneq \ldots$ 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 $l \geq s$.

Remark 5.2 We should mention that there is a wide range of alternative relaxation techniques in the literature, including the use of convex underestimators (cf. [2, 3, 37, 50]) instead of piecewise linear ones. One could then either solve the resulting convex MINLP problems or combine them with outer approximation techniques (cf. [18, 27, 35, 39, 54]). However, here we restrict ourselves to the case of piecewise linear relaxations since the relaxation error computation and especially convergence in the sense of Assumption 5.1 is straightforward. For the above-mentioned approaches, one has to ensure that both of these requirements are fulfilled.

Furthermore, if $\mathcal{R} \in \Omega$ we consider any feasible point $\tilde{x} \in \tilde{S}$ of the corresponding relaxed problem (RMOP$_{\tilde{S}}$) based on the feasible set $\tilde{S} = S^{\mathcal{R}}$ as a feasible point of (MOP), i.e. $\tilde{x} \in S$. Consequently, by Lemma 4.5 for any efficient point $\tilde{x} \in \tilde{S}$ of (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 (RMOP$_{\tilde{S}}$) as a nondominated point of (MOP). For 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 Sect. 4.

In Step 3, we initialize the set

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

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 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

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

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 5 since the method also refines the relaxations if

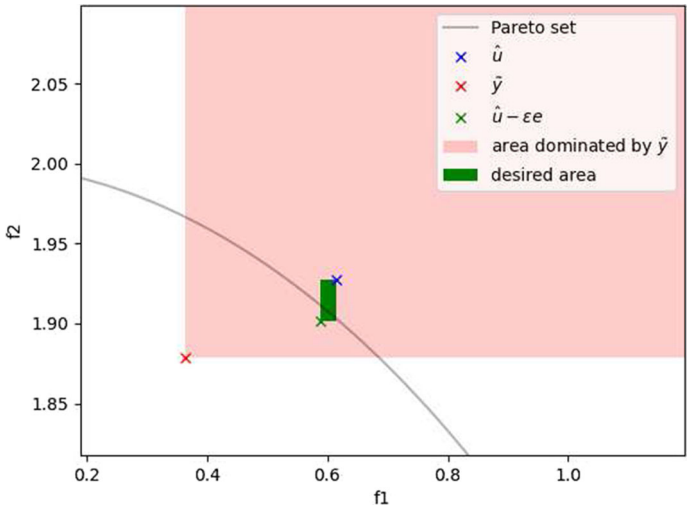

Fig. 5 A strict refinement of the relaxation is needed in order to obtain solutions closer to the desired area as it is dominated by $\tilde{y}$

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 Fig. 5. The possible negative effect of not forcing strictness can be seen in the comparison of Figs. 7 and 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 losing 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 2, 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 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 [13], i.e. $\mathcal{R}_{\text{current}} \in \Omega$, we consider $\tilde{y}$ as a 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 [45], 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.

Remark 5.3 Note that it is not necessary to solve the (WSP $(\alpha; u)$) corresponding to (redMOP($\tilde{x}_I$)) to global optimality. Since we aim to find a feasible point $x \in S$ satisfying $f(x) < \hat{u} - \delta e$—and this is already incorporated in the constraints—it suffices to find a feasible point for (WSP $(\alpha; u)$). Furthermore, even solving to global optimality would not guarantee finding an efficient solution $f(x) \in \mathcal{N}$ for (MOP) since we are restricted to a specific integer assignment. However, it may speed up the algorithm investing the time in computing global solutions of the possibly nonconvex (WSP $(\alpha; u)$) depending on the structure and complexity of (MOP).

We turn now to proving correctness and finiteness of Algorithm 5.

Lemma 5.4 *Let $\tilde{N}$ be the set of best relaxed solutions at some arbitrary point in Algorithm 5. Then $\tilde{N} \subseteq \text{int}(B)\backslash(f(S) + \text{int}(\mathbb{R}^k_+))$ and $\tilde{N}$ is stable.*

Proof We initialize the set $\tilde{N} = \emptyset$. During Algorithm 5 the set $\tilde{N}$ is updated only in Step 14. The update takes place w.r.t. a point $\tilde{y} \in \text{int}(B)\backslash(f(S) + \text{int}(\mathbb{R}^k_+))$ by Lemma 4.5. Thus, by correctness of Algorithm 4 we obtain that $\tilde{N} \subseteq \text{int}(B)\backslash(f(S) + \text{int}(\mathbb{R}^k_+))$ and $\tilde{N}$ is stable. □

Lemma 5.5 *Let N be the set of potentially nondominated points at some arbitrary point in Algorithm 5. Then $N \subseteq f(S)$ and N is stable.*

Proof We initialize the set $N = \emptyset$. During Algorithm 5 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 3 we obtain that $N \subseteq f(S)$ and N is stable. □

Similar to Sect. 3 we prove correctness of the sets U and L.

Lemma 5.6 *Let U be the local upper bound set at some arbitrary point in Algorithm 5. Then U is an upper bound set in the sense of Definition 3.1.*

Proof We initialize the set $U = \{z^u\}$. During Algorithm 5 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 5.7 *Let L be the local lower bound set at some arbitrary point in Algorithm 5. Then L is a lower bound set in the sense of Definition 3.1.*

Proof We initialize the set $L = \{z^\ell\}$. During Algorithm 5 the set L is updated only in Steps 16 and 20. If updated in Step 16, 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 20, 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 20, 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 1 adapted to local lower bounds we obtain that L is a local lower bound set w.r.t. some set $N' \subseteq \text{int}(B)\backslash(f(S) + \text{int}(\mathbb{R}^k_+))$. Note that $\tilde{N} \subseteq N'$. The claim follows by Lemma 3.5. □

After proving that at any point in Algorithm 5 the sets N, $\tilde{N}$, L and U satisfy the requirements, we proceed by showing termination of Algorithm 5 after finitely many iterations of the outer `while`-loop. For that purpose, we first prove that the inner `while`-loop is terminated after a finite number of iterations.

Lemma 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 5. Moreover, let $\hat{u} \in U_{loop}$ be the local upper bound chosen in the* `for`*-loop at some arbitrary point of Algorithm 5. Assume that there exists $\ell \in L$ with $\ell \leq \hat{u}$ and $s\left(\ell, \hat{u}\right) \geq \varepsilon_{encl}$. 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 $\varepsilon_{rel}^{\mathcal{R}^s} < \varepsilon_{rel}$. Hence, in the sth 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 4.5 we have that $\tilde{N} \subseteq \text{int}(B) \backslash (f(S) + \text{int}(\mathbb{R}^k_+))$ 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 to Algorithm 2 we can prove some decrease in some edge lengths after any iteration of the outer `while`-loop.

Theorem 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 5. Moreover, let L^{start} and U^{start} be the local lower and upper bound sets at the beginning of some iteration in Algorithm 5, 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*

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

Proof The proof of part (1) works exactly the same as in the proof of Theorem 3.11. 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

$$\left(u^e - \ell^e\right)_i \geq \max\left\{(u - \ell)_i - \delta,\ \varepsilon_{\text{encl}}\right\}. \tag{10}$$

In particular, (10) 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 5, 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 3.11. 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

$$s\left(\ell', u^s\right) \geq \varepsilon_{\text{encl}} \tag{11}$$

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

$$\ell^s \leq \ell \leq \ell^e \quad \text{and} \quad u^e \leq u \leq u^s \tag{12}$$

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 at 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 5.8 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 (5). Say $u^j \in U_{\text{updated}}$. Using (12), we compute

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

a contradiction to (10).
Case A.2 $\tilde{y} \not< u'$. Then there exists $j \in [k]$ with $u'_j \leq \tilde{y}_j$. Again, using (12), we compute

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

a contradiction to (10).

(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 1 adapted to local lower bounds, 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 (5). Say $\ell^j \in L_{\text{updated}}$. Using (12), we compute

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

a contradiction to (10).
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 (11).

Obtaining a contradiction in all possible cases shows that our assumption (10) cannot be true and therefore statement (2) is true, which completes the proof. □

Similar as in the case of Algorithm 2, Theorem 5.9 enables us to prove finiteness of Algorithm 5.

Theorem 5.10 *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 5. We define*

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

Then the number of iterations of Algorithm 5, i.e. the number of iterations of the outer `while`*-loop, is bounded by* $\max\{1, \kappa\}$*. Furthermore, Algorithm 5 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 3.12. 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 5.8. □

Corollary 5.11 *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 5. 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 5.10 tells us that Algorithm 5 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 5.6 and Lemma 5.7 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 5. 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 5.12 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 3.14. Again, as for Algorithm 2, we did not exactly implement the procedure given in Algorithm 5 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. $\hat{u} \in \{u \in U \mid \exists \ell \in L : s(\ell, u) = w(E)\}$. Again, finiteness of that implementation is not anymore guaranteed by Theorem 5.10. 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 Sect. 3. Furthermore, we use a relative relaxation error calculation. For example, given a quadratic term $h(x) = x^2$ on the interval $x^\ell \le x \le x^u$ we compute the relative relaxation error via

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

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 3.14 as in Sect. 3. In the Figs. 6, 7 and 8, the respective enclosure computed by Algorithm 5 is depicted on the left together with the computational time and number of WSM problems solved, where we count the solution of (RMOP$_{\tilde{S}}$) and the corresponding (redMOP($\tilde{x}_I$)) as one. On the respective right-hand side, for each of these solved relaxed problems, we count the corresponding degree of relaxation—in our case the number of equidistant partitions.

- 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 5 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 Fig. 6 (Left). In Fig. 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

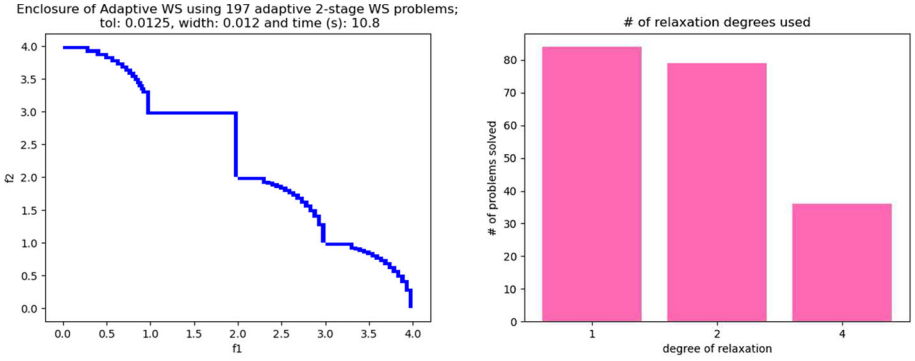

Fig. 6 Computational results on first instance in Example 3.14. Left: enclosure given by LLBs and LUBs of (Circles) obtained by Algorithm 5 with WSM. Right: counter of used degrees of relaxations

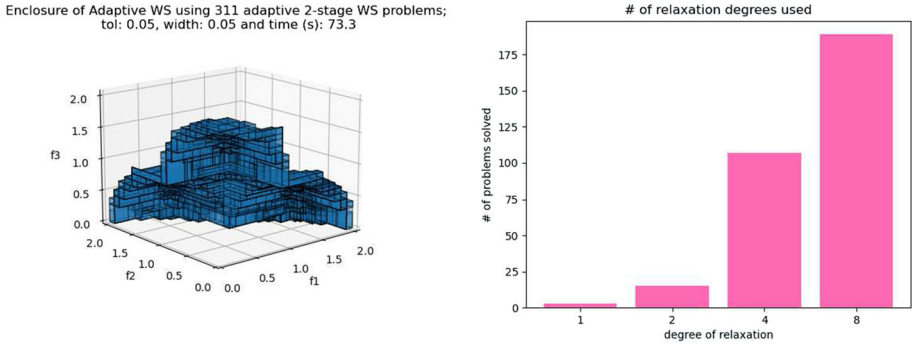

Fig. 7 Computational results on second instance in Example 3.14. Left: enclosure given by LLBs and LUBs of (Circles) obtained by Algorithm 5 with WSM. Right: counter of used degrees of relaxations

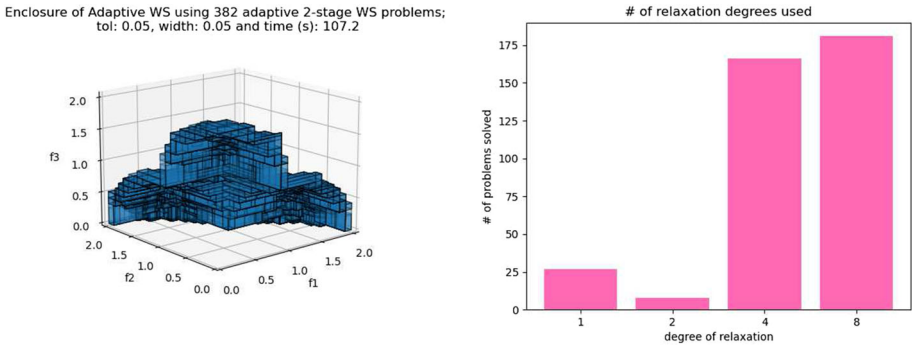

Fig. 8 Computational results on second instance in Example 3.14 with no strict inclusion in Step 10 of Algorithm 5. Left: enclosure given by LLBs and LUBs of (Circles) obtained by Algorithm 5 with WSM. Right: counter of used degrees of relaxations

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, i.e. a degree of relaxation equal to five, 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 5 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 Fig. 7 (Left). In Fig. 7 (Right) the use counters of each relaxation are 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 anymore. 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. In fact, the method decides early in the solution process that coarse relaxations do not suffice to terminate the method. Together with the fact that the degree of relaxation is inherited in the parent history of local upper bounds, this yields that only a few problems are solved with degrees of relaxation 1 and 2. Then the method sticks some time with degree of relaxation 4, while it needs relaxation degree 8 for a large number of problems. In Fig. 8 we have the same setting as in Fig. 7, but do not require a strict inclusion in Step 10 of Algorithm 5. One can see the effect in an increase in computational time and more problems to be solved. Furthermore, on the right one can observe that the algorithm is trying to stick longer with a coarse relaxation before actually going for the refinement step (see the increase of problems with relaxation degrees 1 and 4).

However, comparing the number of problems to be solved as well as especially the computational time with the ones from Sect. 3, one can see that the power of available solvers like SCIP dealing with quadratically constrained problems 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 a MIQCP problem. The general network structure is a graph, 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 contain bilinear or quadratic terms the resulting optimization problem has the mentioned MIQCP 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 [38] and more recently in [19].

For the present paper, we consider three network instances of such decentralized energy supply networks, namely: If e.g., we set up a single objective optimization problem with

	`network 1`	⊂	`network 2`	⊂	`network 3`
# Nodes	12		20		39
# Binaries	108		189		360
# Variables	484		829		1570
# Constraints	620		1064		2014

cost minimization as objective function and put the carbon emissions to the constraints, we obtain the following computational times

- `network 1`: 0.57 s
- `network 2`: 3.35 s
- `network 3`: > 3 h,

using the SCIP solver with the standard settings[3] from the `pyscipopt`-package (cf. [41]).

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

$$\mathrm{R}^{\mathrm{e}}_{i,j} f^{\mathrm{e}}_{\mathrm{in},i,j} = a^{\mathrm{e}} u_j \bar{u}_{i,j}, \tag{13}$$

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

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 (13). 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

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

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 a 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 [8]. By doing so, we obtain for instance

$$\mathrm{R}^{\mathrm{g}}_{i,j} \bar{q}^2_{ij} \leq \bar{p}_{\max} y_{i,j}, \tag{14}$$

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 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

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

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 a 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 5, but chose a relaxation with $r \geq 10$ and then used 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.

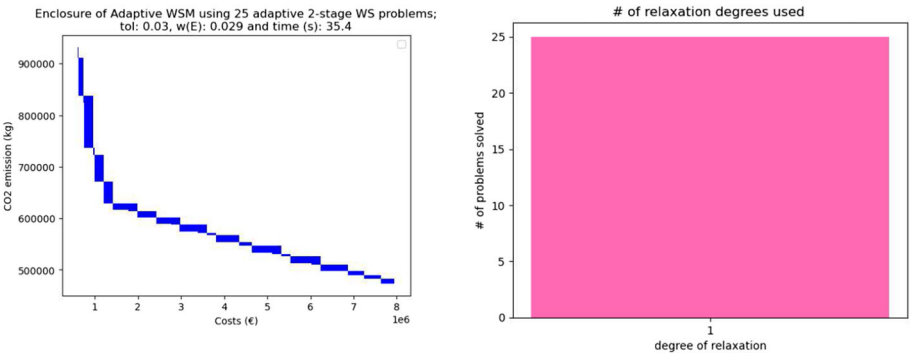

Fig. 9 Computational results on `network 3`. Left: enclosure given by LLBs and LUBs of (Circles) obtained by Algorithm 5 with WSM. Right: counter of used degrees of relaxations

Looking at the results for `network 3` (see Fig. 9; the results for `network 1` and `network 2` are similar) 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.

7 Conclusion

In the present work, a general MIQCP 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., [11]. 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.

Acknowledgements This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) via the German Excellence Strategy. We are also very thankful for the support by Piet Hensel and Dirk König from Rechenzentrum für Versorgungsnetze Wehr GmbH (http://www.rzvn.de) for their support for setting up the numerical model in Sect. 6.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The datasets used for computations in Sect. 6 are available from the corresponding author on reasonable request.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Achterberg, T., Wunderling, R.: Mixed integer programming: analyzing 12 years of progress. In: Facets of Combinatorial Optimization, pp. 449–481. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38189-8_18
2. Adjiman, C.S., Dallwig, S., Floudas, C.A., Neumaier, A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs—I. Theoretical advances. Comput. Chem. Eng. **22**(9), 1137–1158 (1998). https://doi.org/10.1016/S0098-1354(98)00027-1
3. Androulakis, I.P., Maranas, C.D., Floudas, C.A.: αBB: a global optimization method for general constrained nonconvex problems, vol. 7, pp. 337–363 (1995). https://doi.org/10.1007/BF01099647. State of the art in global optimization: computational methods and applications (Princeton, NJ, 1995). https://doi.org/10.1007/BF01099647
4. Banholzer, S.: Rom-Based Multiobjective Optimization with PDE Constraints. Ph.D. thesis, Universität Konstanz, Konstanz (2021)
5. Banholzer, S., Gebken, B., Dellnitz, M., Peitz, S., Volkwein, S.: ROM-based multiobjective optimization of elliptic PDEs via numerical continuation. In: Non-smooth and Complementarity-based Distributed Parameter Systems—Simulation and Hierarchical Optimization. Int. Ser. Numer. Math., vol. 172, pp. 43–76. Birkhäuser/Springer, Cham (2022). https://doi.org/10.1007/978-3-030-79393-7_3
6. Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., Mahajan, A.: Mixed-integer nonlinear optimization. Acta Numer. **22**, 1–131 (2013). https://doi.org/10.1017/S0962492913000032
7. Bestuzheva, K., Besançon, M., Chen, W.-K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., Gottwald, L., Graczyk, C., Halbig, K., Hoen, A., Hojny, C., van der Hulst, R., Koch, T., Lübbecke, M., Maher, S.J., Matter, F., Mühmer, E., Müller, B., Pfetsch, M.E., Rehfeldt, D., Schlein, S., Schlösser, F., Serrano, F., Shinano, Y., Sofranac, B., Turner, M., Vigerske, S., Wegscheider, F., Wellner, P., Weninger, D., Witzig, J.: The SCIP Optimization Suite 8.0. ZIB-Report 21-41, Zuse Institute Berlin (2021). http://nbn-resolving.de/urn:nbn:de:0297-zib-85309
8. Borraz-Sánchez, C., Bent, R., Backhaus, S., Hijazi, H., Van Hentenryck, P.: Convex relaxations for gas expansion planning. INFORMS J. Comput. **28**(4), 645–656 (2016). https://doi.org/10.1287/ijoc.2016.0697
9. Boukouvala, F., Misener, R., Floudas, C.A.: Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. Eur. J. Oper. Res. **252**(3), 701–727 (2016). https://doi.org/10.1016/j.ejor.2015.12.018
10. Burachik, R.S., Kaya, C.Y., Rizvi, M.M.: Algorithms for generating pareto fronts of multi-objective integer and mixed-integer programming problems. Eng. Optim. (2021). https://doi.org/10.1080/0305215X.2021.1939695
11. Burlacu, R.: Adaptive mixed-integer refinements for solving nonlinear problems with discrete decisions. Ph.D. thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (2019)
12. Burlacu, R.: On refinement strategies for solving MINLPs by piecewise linear relaxations: a general red refinement. Optim. Lett. (2021). https://doi.org/10.1007/s11590-021-01740-1
13. Burlacu, R., Geißler, B., Schewe, L.: Solving mixed-integer nonlinear programmes using adaptively refined mixed-integer linear programmes. Optim. Methods Softw. **35**(1), 37–64 (2020). https://doi.org/10.1080/10556788.2018.1556661
14. Cplex, I.I.: V12. 1: user's manual for CPLEX. Int. Bus. Mach. Corp. **46**(53), 157 (2009)
15. Dächert, K., Klamroth, K., Lacour, R., Vanderpooten, D.: Efficient computation of the search region in multi-objective optimization. Eur. J. Oper. Res. **260**(3), 841–855 (2017). https://doi.org/10.1016/j.ejor.2016.05.029
16. De Santis, M., Eichfelder, G., Niebling, J., Rocktäschel, S.: Solving multiobjective mixed integer convex optimization problems. SIAM J. Optim. **30**(4), 3122–3145 (2020). https://doi.org/10.1137/19M1264709
17. Diessel, E.: An adaptive patch approximation algorithm for bicriteria convex mixed-integer problems. Optimization **0**(0), 1–46 (2021). https://doi.org/10.1080/02331934.2021.1939699
18. Duran, M.A., Grossmann, I.E.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. **36**(3), 307–339 (1986). https://doi.org/10.1007/BF02592064
19. Eggen, C., Huynh, T.-V., Link, M., Stephan, P., Volkwein, S.: An MINLP model for designing decentralized energy supply networks. Technical report. arXiv:2212.06527 (2022)
20. Ehrgott, M.: Multicriteria Optimization, 2nd edn., p. 323. Springer, Berlin (2005)
21. Ehrgott, M., Gandibleux, X.: Bound sets for biobjective combinatorial optimization problems. Comput. Oper. Res. **34**(9), 2674–2694 (2007). https://doi.org/10.1016/j.cor.2005.10.003
22. Eichfelder, G.: Twenty years of continuous multiobjective optimization in the twenty-first century. EURO J. Comput. Optim. **9**, 100014 (2021). https://doi.org/10.1016/j.ejco.2021.100014

23. Eichfelder, G., Warnow, L.: An approximation algorithm for multi-objective optimization problems using a box-coverage. J. Global Optim. (2021)
24. Eichfelder, G., Warnow, L.: A hybrid patch decomposition approach to compute an enclosure for multi-objective mixed-integer convex optimization problems (2021)
25. Eichfelder, G., Stein, O., Warnow, L.: A deterministic solver for multiobjective mixed-integer convex and nonconvex optimization (2022)
26. Eichfelder, G., Kirst, P., Meng, L., Stein, O.: A general branch-and-bound framework for continuous global multiobjective optimization. J. Global Optim. **80**(1), 195–227 (2021). https://doi.org/10.1007/s10898-020-00984-y
27. Fletcher, R., Leyffer, S.: Solving mixed integer nonlinear programs by outer approximation. Math. Program. **66**(3, Ser. A), 327–349 (1994). https://doi.org/10.1007/BF01581153
28. Geißler, B., Martin, A., Morsi, A., Schewe, L.: Using piecewise linear functions for solving MINLPs. In: Mixed Integer Nonlinear Programming. IMA Vol. Math. Appl., vol. 154, pp. 287–314. Springer, New York (2012)
29. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2022). https://www.gurobi.com
30. Iapichino, L., Trenz, S., Volkwein, S.: Reduced-order multiobjective optimal control of semilinear parabolic problems. In: Numerical Mathematics and Advanced Applications—ENUMATH 2015. Lect. Notes Comput. Sci. Eng., vol. 112, pp. 389–397. Springer, Cham (2016)
31. Kim, I.Y., de Weck, O.: Adaptive weighted sum method for bi-objective optimization: Pareto front generation. Struct. Multidiscip. Optim. **29**, 149–158 (2005). https://doi.org/10.1007/s00158-004-0465-1
32. Kim, I.Y., de Weck, O.: Adaptive weighted sum method for multiobjective optimization: a new method for Pareto front generation. Struct. Multidiscip. Optim. **31**(2), 105–116 (2006). https://doi.org/10.1007/s00158-005-0557-6
33. Kirlik, G., Sayın, S.: Bilevel programming for generating discrete representations in multiobjective optimization. Math. Program. **169**(2), 585–604 (2018). https://doi.org/10.1007/s10107-017-1149-0
34. Klamroth, K., Lacour, R., Vanderpooten, D.: On the representation of the search region in multi-objective optimization. Eur. J. Oper. Res. **245**(3), 767–778 (2015). https://doi.org/10.1016/j.ejor.2015.03.031
35. Kronqvist, J., Lundell, A., Westerlund, T.: The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. J. Global Optim. **64**(2), 249–272 (2016). https://doi.org/10.1007/s10898-015-0322-3
36. Lee, J., Leyffer, S. (eds.): Mixed Integer Nonlinear Programming. The IMA Volumes in Mathematics and its Applications, vol. 154, p. 690. Springer, New York (2012). https://doi.org/10.1007/978-1-4614-1927-3. Selected papers based on the IMA Hot Topics Workshop "Mixed-Integer Nonlinear Optimization: Algorithmic Advances and Applications" held in Minneapolis, MN, November 17–21, 2008. https://doi.org/10.1007/978-1-4614-1927-3
37. Liberti, L.: Reformulation and convex relaxation techniques for global optimization. Q. J. Belg. Fr. Ital. Oper. Res. Soc. **2**, 255–258 (2004). https://doi.org/10.1007/s10288-004-0038-6
38. Lu, J.: Mixed-Integer Nonlinear Modeling and Optimization of Designing Decentralized Energy Supply Networks. Ph.D. thesis, Universität Konstanz, Konstanz (2023)
39. Lundell, A., Kronqvist, J.: Polyhedral approximation strategies for nonconvex mixed-integer nonlinear programming in SHOT. J. Global Optim. **82**(4), 863–896 (2022). https://doi.org/10.1007/s10898-021-01006-1
40. Lundell, A., Skjäl, A., Westerlund, T.: A reformulation framework for global optimization. J. Global Optim. **57**(1), 115–141 (2013). https://doi.org/10.1007/s10898-012-9877-4
41. Maher, S., Miltenberger, M., Pedroso, J.P., Rehfeldt, D., Schwarz, R., Serrano, F.: PySCIPOpt: mathematical programming in python with the SCIP optimization suite. In: Mathematical Software—ICMS 2016, pp. 301–307. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42432-3_37
42. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs. I. Convex underestimating problems. Math. Program. **10**(2), 147–175 (1976). https://doi.org/10.1007/BF01580665
43. Misener, R., Floudas, C.A.: Global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCQP) through piecewise-linear and edge-concave relaxations. Math. Program. **136**(1, Ser. B), 155–182 (2012). https://doi.org/10.1007/s10107-012-0555-6
44. Morsi, A., Geißler, B., Martin, A.: Mixed integer optimization of water supply networks. In: Mathematical Optimization of Water Networks. Internat. Ser. Numer. Math., vol. 162, pp. 35–54. Birkhäuser/Springer Basel AG, Basel (2012). https://doi.org/10.1007/978-3-0348-0436-3_3
45. Nagarajan, H., Lu, M., Wang, S., Bent, R., Sundar, K.: An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs. J. Global Optim. **74**(4), 639–675 (2019). https://doi.org/10.1007/s10898-018-00734-1
46. Pascoletti, A., Serafini, P.: Scalarizing vector optimization problems. J. Optim. Theory Appl. **42**(4), 499–524 (1984). https://doi.org/10.1007/BF00934564

47. Perini, T., Boland, N., Pecin, D., Savelsbergh, M.: A criterion space method for biobjective mixed integer programming: the boxed line method. INFORMS J. Comput. **32**(1), 16–39 (2020). https://doi.org/10.1287/ijoc.2019.0887
48. Ryu, N., Min, S.: Multiobjective optimization with an adaptive weight determination scheme using the concept of hyperplane. Int. J. Numer. Methods Eng. **118**(6), 303–319 (2019). https://doi.org/10.1002/nme.6013
49. Sayın, S.: Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. Math. Program. **87**(3, Ser. A), 543–560 (2000). https://doi.org/10.1007/s101070050128
50. Skjäl, A.: On the use of convex under estimators in global optimization. Ph.D. thesis, Abo Akademi University (2014)
51. Stidsen, T., Andersen, K.A.: A hybrid approach for biobjective optimization. Discrete Optim. **28**, 89–114 (2018). https://doi.org/10.1016/j.disopt.2018.02.001
52. Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-integer Nonlinear Programming. Nonconvex Optimization and its Applications, vol. 65, p. 475. Kluwer Academic Publishers, Dordrecht (2002). https://doi.org/10.1007/978-1-4757-3532-1. Theory, algorithms, software, and applications
53. Vielma, J.P., Ahmed, S., Nemhauser, G.: Mixed-integer models for nonseparable piecewise-linear optimization: unifying framework and extensions. Oper. Res. **58**(2), 303–315 (2010). https://doi.org/10.1287/opre.1090.0721
54. Westerlund, T., Pettersson, F.: An extended cutting plane method for solving convex MINLP problems. Comput. Chem. Eng. **19**, 131–136 (1995). https://doi.org/10.1016/0098-1354(95)87027-X. European Symposium on Computer Aided Process Engineering 3–5

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.