Eichfelder, Gabriele; Warnow, Leo

An approximation algorithm for multi-objective optimization problems using a box-coverage

Journal of Global Optimization

Provided in Cooperation with:
Springer Nature

Suggested Citation: Eichfelder, Gabriele; Warnow, Leo (2021) : An approximation algorithm for multi-objective optimization problems using a box-coverage, Journal of Global Optimization, ISSN 1573-2916, Springer US, New York, NY, Vol. 83, Iss. 2, pp. 329-357, https://doi.org/10.1007/s10898-021-01109-9

Journal of Global Optimization (2022) 83:329–357
https://doi.org/10.1007/s10898-021-01109-9

An approximation algorithm for multi-objective optimization problems using a box-coverage

Gabriele Eichfelder[1] · **Leo Warnow**[1]

Received: 29 October 2020 / Accepted: 29 October 2021 / Published online: 24 November 2021

Abstract
For a continuous multi-objective optimization problem, it is usually not a practical approach to compute all its nondominated points because there are infinitely many of them. For this reason, a typical approach is to compute an approximation of the nondominated set. A common technique for this approach is to generate a polyhedron which contains the nondominated set. However, often these approximations are used for further evaluations. For those applications a polyhedron is a structure that is not easy to handle. In this paper, we introduce an approximation with a simpler structure respecting the natural ordering. In particular, we compute a box-coverage of the nondominated set. To do so, we use an approach that, in general, allows us to update not only one but several boxes whenever a new nondominated point is found. The algorithm is guaranteed to stop with a finite number of boxes, each being sufficiently thin.

Keywords Multi-objective optimization · Approximation algorithm · Nondominated set · Enclosure · Box-coverage

Mathematics Subject Classification 90C26 · 90C29 · 90C59 · 65K05

1 Introduction

The aim of multi-objective optimization is to minimize not only one but multiple objectives at the same time. Usually, it is not possible to find a feasible point that minimizes all objectives as these are conflicting. Hence, a commonly used approach is to find so-called nondominated points in the criterion space which belong to so-called efficient solutions in the decision space. The set of all these nondominated points is called nondominated set or Pareto front. Given an efficient solution, it is not possible to find a feasible point that leads to an improvement for any

✉ Leo Warnow
leo.warnow@tu-ilmenau.de

Gabriele Eichfelder
gabriele.eichfelder@tu-ilmenau.de

[1] Technische Universität Ilmenau, P.O. Box 10 05 65, 98684 Ilmenau, Germany

objective without deteriorating another. For an introduction to multi-objective optimization see [9,30,32].

In general, there is an infinite number of nondominated points for a continuous multi-objective optimization problem. Thus, a common technique is to compute a (finite) approximation of the nondominated set. In [34] a survey of such techniques including a classification can be found. There are basically two kinds of approximations. On the one hand, there are approaches which compute a finite number of nondominated points to represent the whole nondominated set which we refer to as representation approach (e.g., [4,16,42]). On the other hand, there are approaches which compute a set (instead of a finite number of points) that contains the nondominated set which we refer to as coverage approach. A common technique for coverage approaches is to combine inner and outer approximations (e.g., [11,26,36]) . This is sometimes referred to as sandwiching, see [1].

Those sandwiching techniques using inner and outer approximations lead to a polyhedron which contains the nondominated set. For representing such polyhedra, usually their vertices are used. Hence, updating such polyhedral approximations is a constant change of computing inner and outer approximations and recomputing at least some of the vertices from a hyperplane representation, see for instance [8,11].

Another approach are coverages that consist of boxes. Boxes can be easily represented by their corners, which we refer to as lower and upper bound. Thus, one can expect that updating (a collection of) boxes requires less effort than updating a polyhedron. Moreover, boxes respect the natural ordering. This is an advantage for applications that perform further computation based on the approximation.

For example, one could think of fields as mixed-integer multi-objective optimization where the nondominated set for the mixed-integer problem can be computed by comparing the nondominated sets of the multi-objective optimization problems that arise by fixing the values of the integer variables.

Another field is set optimization where values of set-valued objective functions have to be compared, see [24]. Robust approaches for handling uncertainties in multi-objective optimization lead to such set optimization problems, see [21]. For instance, for the upper-type set relation comparing compact sets corresponds to comparing Pareto fronts, for which coverages can be used, see for example [13].

Thereby, boxes can be compared more easily than general polyhedra. Moreover, in case these coverages are not exact enough, it is important to be able to improve those iteratively. In view of this, we are able to present such a guarantee for our coverage approach which we call Halving Theorem.

An example for a box-coverage is presented in [19] for bi-objective problems and has been extended in [27] to tri-objective problems. The approach in [27] is to split a given box into seven subboxes using update points that are computed using a lexicographic ε-constraint scalarization. Boxes are removed if they do not contain any feasible points. Otherwise they are split again until their maximum width is smaller than a given tolerance. In [5] an algorithm to generate a box-coverage for bi-objective integer programs is presented. This algorithm also uses the approach to divide a given box into (two) subboxes. The update point, which decides where the box is split, is computed using an approach related to the Pascoletti–Serafini scalarization. It also takes into account the integrality of the problem to avoid working with infeasible subboxes. To the best of our knowledge there is no generalization of the approaches from [27] and [5] for an arbitrary number of objective functions.

Another approach for a box-based approximation, mainly focused on discrete tri-objective problems, is given in [6]. It is different from the approaches described above as all boxes have the same lower bound. In other words, only the upper bounds of the boxes are updated.

Moreover, the boxes are allowed to intersect. As a result, a single update point, i.e., a nondominated point, can lead to a split of multiple boxes and, consequently, redundant subboxes. However, approaches how to remove redundancy are presented as well.

In [25] a concept of search regions related to those from [6] and so-called local upper bounds are presented. They can be used for any number of objective functions and in particular for (non-discrete) multi-objective optimization problems. In [38] and very recently in [7] representation approaches that inherently also generate a box approximation have been presented, where [7] demonstrates the use of such approaches in radiotherapy planning as a real world application.

For completeness, we want to mention that branch–and–bound algorithms usually use boxes in the decision space and some of these algorithms also generate boxes in the criterion space, see [14,15,31,35]. However, we want to focus here on working in the criterion space without creating any substructure in the decision space. One reason for our focus on a criterion space based method is that the computation time of branch-and-bound approaches in the decision space increases quite fast when the number of decision variables increases. Since our algorithm works in the criterion space, its computation time depends more on the number of objectives than on the number of decision variables. Hence, our algorithm focuses on such cases where there are more decision variables than objective functions whereas branch–and–bound approaches as those from [14,15,31,35] are alternatives in case that the dimension of the decision space is relatively small. Some of these methods like [14,35] are also limited to purely box-constrained optimization problems. Our approach works for an arbitrary feasible set as long as it is compact.

In this paper, we introduce a new approximation algorithm for multi-objective optimization problems using the bound concepts from [25] to compute a box-coverage of the nondominated set. To the best of our knowledge, we are the first to present a box approximation concept for an arbitrary dimension of the criterion space involving both upper and lower bound improvements with an exact bound on the number of iterations needed. Moreover, we show that in every iteration a certain improvement of the approximation can be guaranteed, which we refer to as Halving Theorem. While we recommend our algorithm most of all for convex multi-objective optimization problems, i.e., problems where all objective and constraint functions are smooth and convex, the theoretical results presented in this paper still hold when assuming continuous objective functions and a compact feasible set. In particular, the presented algorithm needs to solve a large number of single-objective optimization problems that are derived from the original multi-objective optimization problem. It is crucial for the performance of the algorithm that a fast and reliable solver for these single-objective subproblems is available. A well-known class of optimization problems for which such a solver is available are smooth convex problems. In case the problems are nonconvex, a suitable global solver needs to be used. Hence, the reader should be aware that while in theory the algorithm presented in this paper works even under weaker assumptions the best solvers for the subproblems exist for smooth convex optimization problems.

The remaining paper is structured as follows. We start in Sect. 2 with some notations and definitions. We also present the problem formulation (MOP) and characterize the kind of approximation that we aim for. In Sect. 3 we discuss the approach to characterize the boxes of our approximation using lower and upper bounds based on the concepts from [25]. Then, in Sect. 4 we introduce our new algorithm to compute the box-based approximation of the nondominated set including a detailed discussion of its properties, such as finiteness. Finally, in Sect. 5 some numerical results for using our algorithm to compute an approximation are presented.

2 Notations and definitions

For a positive integer $n \in \mathbb{N}$ we use the notation $[n] := \{1, \ldots, n\}$. All relations in this paper are meant to be read component-wise, i.e., for $x, x' \in \mathbb{R}^n$ it is

$$x \leq x' \Leftrightarrow x_i \leq x'_i \text{ for all } i \in [n],$$
$$x < x' \Leftrightarrow x_i < x'_i \text{ for all } i \in [n].$$

For $l, u \in \mathbb{R}^n$ with $l \leq u$ we denote by $[l, u] := \left\{ y \in \mathbb{R}^n \mid l \leq y \leq u \right\}$ the box with lower bound l and upper bound u. As already mentioned in the introduction, we focus on multi-objective optimization problems. We denote by $f_i : \mathbb{R}^n \to \mathbb{R}, i \in [m]$ the objective functions and by $S \subseteq \mathbb{R}^n$ the feasible set. We also write $f = (f_1, \ldots, f_m) : \mathbb{R}^n \to \mathbb{R}^m$. Then, our multi-objective optimization problem is given as

$$\min_x \ f(x) \quad \text{s.t.} \quad x \in S \tag{MOP}$$

where all functions $f_i, i \in [m]$ are assumed to be continuous and S is assumed to be a nonempty, compact set. Since $f(S)$ is bounded, it holds that

$$\exists\, z, Z \in \mathbb{R}^m : \ f(S) \subseteq \text{int}(B) \text{ with } B := [z, Z]. \tag{2.1}$$

We assume in the following that such a box B is known. While there is no need to assume convexity of the objective functions $f_i, i \in [m]$ and the set S for the theoretical results in this paper, one needs to be able to solve a single-objective subproblem related to (MOP), see (SUP(l, u)) on page 13. Fast and reliable solvers for such optimization problems are available for example when assuming convexity of the objective functions $f_i, i \in [m]$ and the set S. The corresponding subproblems are then single-objective convex optimization problems where every locally optimal solution is also a globally optimal solution. Hence, from a practical point of view, we recommend to use our algorithm first of all for smooth convex multi-objective optimization problems. We discuss this in more detail in Sect. 5. However, for the theoretical results of our paper we stick with the weaker assumptions of continuous objective functions and a compact feasible set S.

As the different objective functions of (MOP) are usually competing with each other, in general it is not possible to find a feasible point that minimizes all objectives at the same time. Thus, we use the following optimality concepts.

Definition 2.1 A point $\bar{x} \in S$ is called an efficient solution for (MOP) if there exists no $x \in S$ with $f_i(x) \leq f_i(\bar{x})$ for all $i \in [m]$ and with $f_j(x) < f_j(\bar{x})$ for at least one $j \in [m]$. It is called a weakly efficient solution for (MOP) if there exists no $x \in S$ with $f_i(x) < f_i(\bar{x})$ for all $i \in [m]$.

For a given $\varepsilon > 0$ we call $\bar{x} \in S$ an ε-efficient solution for (MOP) if there exists no $x \in S$ with $f_i(x) \leq f_i(\bar{x}) - \varepsilon$ for all $i \in [m]$ and with $f_j(x) < f_j(\bar{x}) - \varepsilon$ for at least one $j \in [m]$. It is called a weakly ε-efficient solution for (MOP) if there exists no $x \in S$ with $f_i(x) < f_i(\bar{x}) - \varepsilon$ for all $i \in [m]$.

We use a related concept in the criterion space, called dominance.

Definition 2.2 Let $y^1, y^2 \in \mathbb{R}^m$ and $\preceq \in \{\leq, \geq\}$. Then y^2 is dominated by y^1 with respect to $\preceq$ if $y^1 \neq y^2, \ y^1 \preceq y^2$. For a set $N \subseteq \mathbb{R}^m$ a vector $y \in \mathbb{R}^m$ is dominated given N with respect to $\preceq$ if

$$\exists\, \hat{y} \in N : \ \hat{y} \neq y, \ \hat{y} \preceq y.$$

If y is not dominated given N w.r.t. $\preceq$, it is called nondominated given N with respect to $\preceq$. Analogously, for $\prec \in \{<, >\}$ we say y^2 is strictly dominated by y^1 with respect to $\prec$ if $y^1 \prec y^2$ and a vector $y \in \mathbb{R}^m$ is strictly dominated given a set $N \subseteq \mathbb{R}^m$ with respect to $\prec$ if

$$\exists\, \hat{y} \in N \colon \hat{y} \prec y.$$

If y is not strictly dominated given N w.r.t. $\prec$, it is called weakly nondominated given N with respect to $\prec$.

In general, the specification of the relation $\preceq/\prec$ is left out if it is known by context. As the images $f(\bar{x})$ of efficient solutions $\bar{x} \in S$ are nondominated given $f(S)$ w.r.t. $\leq$, they are called nondominated points of (MOP). We denote by $\mathcal{E}$ the set of efficient solutions (also efficient set) and by $\mathcal{N}$ the set of nondominated points (also nondominated set) of (MOP), i.e., $\mathcal{N} := \left\{ y \in \mathbb{R}^m \;\middle|\; y = f(x),\ x \in \mathcal{E} \right\} \subseteq \mathbb{R}^m$. Also, for an arbitrary $\varepsilon > 0$, we denote the ε-nondominated set for (MOP) by

$$\mathcal{N}_\varepsilon := \left\{ y \in \mathbb{R}^m \;\middle|\; y = f(x),\ x \text{ is an } \varepsilon\text{-efficient solution for } (\textit{MOP}) \right\}$$

and the weakly ε-nondominated set for (MOP) by

$$\mathcal{N}_\varepsilon^w := \left\{ y \in \mathbb{R}^m \;\middle|\; y = f(x),\ x \text{ is a weakly } \varepsilon\text{-efficient solution for } (\textit{MOP}) \right\}.$$

In this paper we focus on the criterion space and hence, on finding an approximation of the set $\mathcal{N}$. As already mentioned in the introduction, we aim for a box-based coverage of $\mathcal{N}$. The concept of an enclosure, as presented in [12], realizes such a box-based coverage.

Definition 2.3 Let $L, U \subseteq \mathbb{R}^m$ be two finite sets with

$$\mathcal{N} \subseteq L + \mathbb{R}^m_+ \text{ and } \mathcal{N} \subseteq U - \mathbb{R}^m_+. \tag{2.2}$$

Then L is called lower bound set, U is called upper bound set, and the set $\mathcal{A}$ which is given as

$$\mathcal{A} = \mathcal{A}(L, U) := (L + \mathbb{R}^m_+) \cap (U - \mathbb{R}^m_+) = \bigcup_{l \in L} \bigcup_{\substack{u \in U, \\ l \preceq u}} [l, u] \tag{2.3}$$

is called approximation or enclosure of the nondominated set $\mathcal{N}$ given L and U.

For an illustration of this concept, see Fig. 1. In this figure, the nondominated set $\mathcal{N}$ is given in orange and the sets $L = \{l^1, l^2\}$ and $U = \{u^1, u^2\}$ are lower and upper bound sets as in Definition 2.3. The box structure of the corresponding approximation $\mathcal{A}$ can also be seen.

We aim for an approximation of certain quality. For this reason, we use a quality criterion that is presented in [12]. There, the authors suggest to generalize the quality citerion given by the interval length $u - l$ of enclosing intervals $[l, u]$ from single-objective optimization to the so-called width $w(\mathcal{A})$ of the enclosure $\mathcal{A}$ with respect to the direction of the all-ones vector e, i.e., to define $w(\mathcal{A})$ as the optimal value of

$$\sup_{y,t} \frac{\|(y + te) - y\|}{\sqrt{m}} \quad \text{s.t. } y, y + te \in \mathcal{A},\ t \in \mathbb{R}_+. \tag{2.4}$$

This definition arises quite naturally, which we want to explain briefly. By Definition 2.3, we have $\mathcal{N} \subseteq \mathcal{A}$. Besides that, it is also reasonable to aim for an approximation $\mathcal{A}$ that only consists of points that are at least approximately nondominated. For example, we can demand that for some $\varepsilon > 0$ we have that $y \in \mathcal{N}_\varepsilon$ for all $y \in \mathcal{A} \cap f(S)$. A sufficient criterion for this to hold would be that for any $y \in \mathcal{A}$ we have that $y - \varepsilon e \notin \mathcal{A}$. In other words, the quality of

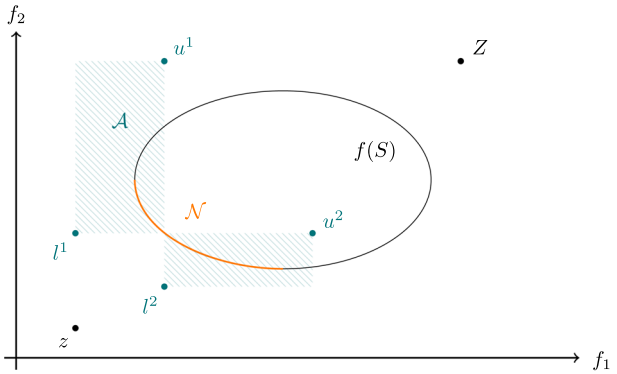


Fig. 1 Illustration of approximation $\mathcal{A}$ for $L = \{l^1, l^2\}$ and $U = \{u^1, u^2\}$

the approximation $\mathcal{A}$ can be defined as the largest $\varepsilon > 0$ such that there exists some $y \in \mathcal{A}$ with $y + \varepsilon e \in \mathcal{A}$. This leads exactly to the definition of $w(\mathcal{A})$ from (2.4). Moreover, this relation between $\mathcal{A}$ and $\mathcal{N}_\varepsilon$ is also shown in [12, Lemma 3.1]. In particular, for any $\varepsilon > 0$ and an approximation $\mathcal{A}$ with $w(\mathcal{A}) < \varepsilon$ it holds that $\mathcal{A} \cap f(S) \subseteq \mathcal{N}_\varepsilon$.

A similar result can be obtained for the polyhedral approach from [11] for convex multi-objective problems that generates an inner approximation $\mathcal{P}^i$ and an outer approximation $\mathcal{P}^o$ of the nondominated set. It is shown in [11, Theorem 4.3] that for the nondominated set $\mathcal{N}_{\mathcal{P}^i}$ of $\mathcal{P}^i$ it holds that $\mathcal{N}_{\mathcal{P}^i} \subseteq \mathcal{N}_\varepsilon^w$. Thereby, ε is an upper bound on the distance between any vertex v of the polyhedral approximation and the boundary of $f(S)$. More precisely, denote by V the vertex set of the polyhedron and choose a fixed interior point $p \in f(S) + \mathbb{R}^m_+$. Then, for each $v \in V$ and its corresponding (unique) boundary point $b^v := \lambda v + (1-\lambda)p \in f(S) + \mathbb{R}^m_+$, $\lambda \in (0,1)$ it holds that the distance $d(v, b^v)$ is at most ε.

In [12] the authors have also shown that there is an equivalent formulation of (2.4) that better fits the box-approximation concept. They denote the shortest edge of a box $[l, u]$ by

$$s(l,u) := \min_{i \in [m]} (u_i - l_i)$$

and show in [12, Lemma 3.2] that the width $w(\mathcal{A})$ of an enclosure $\mathcal{A}$ equals the optimal value of

$$\sup_{l,u} \; s(l,u) \quad \text{s.t.} \quad l \in L,\; u \in U,\; l \leq u.$$

We want to remark that [12] presents a branch-and-bound framework in the decision space while we focus on the criterion space. For our paper, we only make use of their enclosure concept and the corresponding quality measure w.

3 Computing lower and upper bounds

In this section, we present an approach on how to choose and how to compute the lower and upper bound sets L and U. A suitable concept for both are the so-called Local Upper Bounds (LUB). We use them as given in [25].

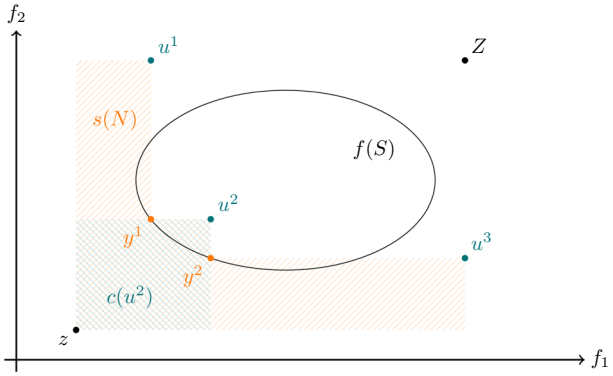


Fig. 2 Illustration of lower search region, lower search zone and local upper bounds

Definition 3.1 A set $Y \subseteq \mathbb{R}^m$ is called stable with respect to $\preceq \in \{\leq, \geq\}$ if no element of Y dominates another, i.e., $y^1 \not\preceq y^2$ for all $y^1, y^2 \in Y$ with $y^1 \neq y^2$.

Analogously to the concept of dominance, the specification of the order relation $\preceq$ is often left out if it is known by context. It is easy to see that the nondominated set $\mathcal{N}$ is a stable set w.r.t. $\leq$.

Definition 3.2 Let $N \subseteq f(S)$ be a finite and stable (w.r.t. $\leq$) set. Then the lower search region for N is $s(N) := \left\{ y \in \text{int}(B) \;\middle|\; y' \not\leq y \text{ for every } y' \in N \right\}$ and the lower search zone for some $u \in \mathbb{R}^m$ is $c(u) := \left\{ y \in \text{int}(B) \;\middle|\; y < u \right\}$. A set $U = U(N)$ is called local upper bound set given N if

1. $s(N) = \bigcup_{u \in U(N)} c(u)$,
2. $c(u^1) \not\subset c(u^2)$ for all $u^1, u^2 \in U(N)$.

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

Given the set N, the search region $s(N)$ contains all potentially nondominated points in $\text{int}(B)$ given N w.r.t. $\leq$ without N itself. The latter because (just by the definition) it always holds $N \cap s(N) = \emptyset$. In other words, $s(N)$ contains all elements $y \in \text{int}(B) \backslash N$ that are not dominated by any $y' \in N$. Hence, dominance in the context of local upper bounds is always dominance w.r.t. $\leq$ and also stable always means stable w.r.t. $\leq$. It is known that for any finite and stable set $N \subseteq f(S)$ the local upper bound set $U(N)$ is uniquely determined and finite, see [12].

For an illustration of the concept of local upper bounds, see Fig. 2. For a stable set $N = \{y^1, y^2\} \subseteq \mathbb{R}^2$ a local upper bound set $U(N) = \{u^1, u^2, u^3\}$ is shown and also the lower search zone $c(u^2)$ and the lower search region $s(N)$ are highlighted.

The following lemma presents a relation between local upper bound sets and upper bound sets as presented in Definition 2.3.

Lemma 3.3 *Let $N \subseteq f(S)$ be a finite and stable set. Then it holds*

$$\mathcal{N} \subseteq \text{cl}(s(N)) = \bigcup_{u \in U(N)} \text{cl}(c(u)) \subseteq \bigcup_{u \in U(N)} \{u\} - \mathbb{R}^m_+ = U(N) - \mathbb{R}^m_+.$$

Proof We only need to show $\mathcal{N} \subseteq \text{cl}(s(N))$. So let $\bar{y} \in \mathcal{N}$ be a nondominated point of (MOP) and assume $\bar{y} \notin s(N)$. Then there exists some $y' \in N \subseteq f(S)$ with $y' \leq \bar{y}$. This

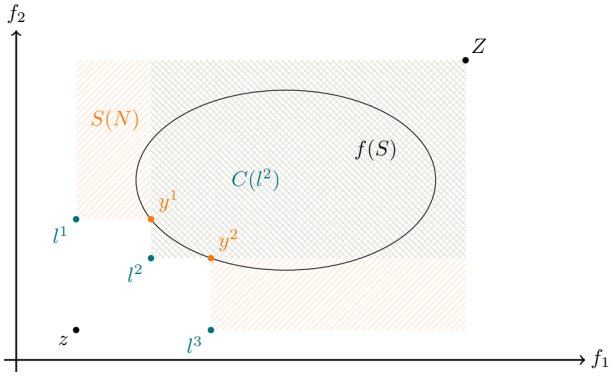


Fig. 3 Illustration of upper search region, upper search zone and local lower bounds

implies that $y' = \bar{y}$ because otherwise $\bar{y}$ would be dominated by y'. Since $y' \in N \subseteq \text{int}(B)$, there exists $\varepsilon > 0$ such that $B_\varepsilon(y') := \left\{ y \in \mathbb{R}^m \;\middle|\; \left\| y - y' \right\| \leq \varepsilon \sqrt{m} \right\} \subseteq \text{int}(B)$.

Hence, for $y^k := y' - \frac{\varepsilon}{k} e,\ k \in \mathbb{N}$ we have that $(y^k)_{k\in\mathbb{N}} \subseteq B_\varepsilon(y') \subseteq \text{int}(B)$. Moreover, it holds that $y^k < y'$ for all $k \in \mathbb{N}$. This implies that $(y^k)_{k\in\mathbb{N}} \subseteq s(N)$, because otherwise there exists $y'' \in N$ and an index $k \in \mathbb{N}$ with $y'' \leq y^k < y'$, which contradicts the assumption that N is stable. Thus, we obtain that $\bar{y} = y' = \lim_{k\to\infty} y^k \in \text{cl}(s(N))$. □

Hence, $U = U(N)$ is an upper bound set in the sense of Definition 2.3 for any finite and stable set $N \subseteq f(S)$.

This concept leads to upper bounds for the nondominated set of (MOP). Now, we show how to use it to gain lower bounds. This is one of the main differences when comparing our approach to that in [25] where only the local upper bounds are used. To distinguish between the local upper bounds and the closely related local lower bounds, which we present in the next definition, we use upper case notation instead of lower case notation for the search region and search zones.

Definition 3.4 Let $N \subseteq \text{int}(B)$ be a finite and stable (w.r.t $\geq$) set. Then the upper search region for N is $S(N) := \left\{ y \in \text{int}(B) \;\middle|\; y' \not\geq y \text{ for every } y' \in N \right\}$ and the upper search zone for some $l \in \mathbb{R}^m$ is $C(l) := \left\{ y \in \text{int}(B) \;\middle|\; y > l \right\}$. A set $L = L(N)$ is called local lower bound set given N if

1. $S(N) = \bigcup_{l\in L(N)} C(l)$,
2. $C(l^1) \not\subset C(l^2)$ for all $l^1, l^2 \in L(N)$.

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

In the context of local lower bounds, dominance is always dominance w.r.t. $\geq$ and stable sets are stable w.r.t. $\geq$ as well.

In Fig. 3 an illustration of the concept of local lower bounds is given for the same setting as in Fig. 2. We have the same stable set $N = \{y^1, y^2\} \subseteq \mathbb{R}^2$, a local lower bound set $L(N) = \{l^1, l^2, l^3\}$, the upper search zone $C(l^2)$, and the upper search region $S(N)$.

Next, we show that for some specific sets N the local lower bound set $L(N)$ is indeed a lower bound set in the sense of Definition 2.3.

Lemma 3.5 *Let $N \subseteq \text{int}(B)$ be a finite and stable (w.r.t. $\geq$) set such that for every $y \in N$ there is no $\hat{y} \in f(S)$ with $\hat{y} \leq y$, $\hat{y} \neq y$. Then $L = L(N)$ is a lower bound set in the sense of Definition 2.3.*

Proof Let $\bar{y} \in \mathcal{N} \subseteq f(S) \subseteq \text{int}(B)$ be a nondominated point of (MOP). Then for every $y' \in N$ it holds $y' = \bar{y}$ or $y' \not\geq \bar{y}$. Hence, using Definition 3.4, we have $\mathcal{N} \subseteq N \cup S(N) \subseteq \text{cl}(S(N))$, where $N \subseteq \text{cl}(S(N))$ can be shown using similar arguments as in the proof of Lemma 3.3. Finally, this leads to

$$\mathcal{N} \subseteq \text{cl}(S(N)) = \bigcup_{l \in L(N)} \text{cl}(C(l)) \subseteq \bigcup_{l \in L(N)} \{l\} + \mathbb{R}^m_+ = L(N) + \mathbb{R}^m_+$$

and $L = L(N)$ is a lower bound set in the sense of Definition 2.3. □

In particular, for any finite and stable set $N \subseteq \text{int}(B) \backslash (f(S) + (\mathbb{R}^m_+ \backslash \{0\}))$ the assumptions of Lemma 3.5 are satisfied. As we need this result later in Sect. 4 (Lemma 4.4), we briefly summarize the relation between local lower and local upper bound sets and bound sets as given in Definition 2.3.

Corollary 3.6 *Let $N^1 \subseteq f(S)$ be a finite and stable set w.r.t. $\leq$ and $N^2 \subseteq \text{int}(B) \backslash (f(S) + (\mathbb{R}^m_+ \backslash \{0\}))$ a finite and stable set w.r.t $\geq$. Then $U(N^1)$ is an upper bound set and $L(N^2)$ is a lower bound set in the sense of Definition 2.3.*

For Definitions 3.2 and 3.4 one does not necessarily need to assume N to be stable. In particular, let $N \subseteq f(S)$ be an arbitrary set and denote by $\hat{N} := \{y \in N \mid y\}$ y is nondomindated given N. Then it is known from [25, Remark 2.2] that $s(N) = s(\hat{N})$. This also implies $U(N) = U(\hat{N})$. This holds analogously for the upper search regions and the corresponding local lower bound sets.

In the following, we present a method to compute local upper and local lower bounds. To provide initial local lower bound and local upper bound sets, we set $U(\emptyset) = \{Z\}$ and $L(\emptyset) = \{z\}$. It is easy to see that these sets satisfy Definitions 3.2 and 3.4. The sets L and U are then updated using points $y \in \text{int}(B)$ to obtain smaller search regions. As updating these sets is done by using projections, we use here the following notation from [25]. For $y \in \mathbb{R}^m$, $\alpha \in \mathbb{R}$ and an index $i \in [m]$ we define

$$\begin{aligned} y_{-i} &:= (y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_m)^\top \text{ as well as} \\ (\alpha, y_{-i}) &:= (y_1, \ldots, y_{i-1}, \alpha, y_{i+1}, \ldots, y_m)^\top. \end{aligned}$$

Using this notation, Algorithm 1 shows an updating procedure for local upper bound sets as presented in [25, Algorithm 3]. We briefly explain the algorithm after the forthcoming Lemma 3.7.

Due to the close relation between local upper bounds and local lower bounds, the concept of Algorithm 1 can also be used for updating local lower bound sets. To do so, one simply has to replace every $<$ by $>$ and every $\leq$ by $\geq$. This leads to the updating procedure as given in Algorithm 2.

In Sect. 4 we present our new algorithm to generate a box-coverage of the nondominated set $\mathcal{N}$. The properties of this algorithm, e.g., finiteness, are highly depending on the properties of the updating procedures for local lower and local upper bounds. For this reason, we discuss those properties in the remaining part of this section. Due to the analogies of both procedures, we focus on local upper bounds and Algorithm 1.

Our Algorithm 1 slightly differs from [25, Algorithm 3]. Compared to the original algorithm, we do not assume the update point $y \in f(S)$ to be nondominated given N. However, the algorithm still works correctly. For any update point y that is nondominated given N the correctness of the algorithm is shown in [25]. For update points y that are dominated given N the correctness of the algorithm is shown in the following lemma.

Algorithm 1 Updating a local upper bound set

Input: Local upper bound set $U(N)$ and update point $y \in f(S)$
Output: Updated local upper bound set $U(N \cup \{y\})$

procedure UPDATELUB($U(N)$, y)
  $A = \{u \in U(N) \mid y < u\}$
  for $i \in [m]$ **do**
    $B_i = \{u \in U(N) \mid y_i = u_i \text{ and } y_{-i} < u_{-i}\}$
    $P_i = \emptyset$
  end for
  for $i \in [m]$ **do**
    for $u \in A$ **do**
      $P_i = P_i \cup \{(y_i, u_{-i})\}$
    end for
  end for
  for $i \in [m]$ **do**
    $P_i = \{u \in P_i \mid u \not\leq u' \text{ for all } u' \in P_i \cup B_i,\ u' \neq u\}$
  end for
  $U(N \cup \{y\}) = (U(N) \backslash A) \cup \bigcup_{i \in [m]} P_i$
end procedure

Algorithm 2 Updating a local lower bound set

Input: Local lower bound set $L(N)$ and update point $y \in \text{int}(B)$
Output: Updated local lower bound set $L(N \cup \{y\})$

procedure UPDATELLB($L(N)$, y)
  $A = \{l \in L(N) \mid y > l\}$
  for $i \in [m]$ **do**
    $B_i = \{l \in L(N) \mid y_i = l_i \text{ and } y_{-i} > l_{-i}\}$
    $P_i = \emptyset$
  end for
  for $i \in [m]$ **do**
    for $l \in A$ **do**
      $P_i = P_i \cup \{(y_i, l_{-i})\}$
    end for
  end for
  for $i \in [m]$ **do**
    $P_i = \{l \in P_i \mid l \not\geq l' \text{ for all } l' \in P_i \cup B_i,\ l' \neq l\}$
  end for
  $L(N \cup \{y\}) = (L(N) \backslash A) \cup \bigcup_{i \in [m]} P_i$
end procedure

Lemma 3.7 *Let N be a finite and stable set, $U = U(N)$ a local upper bound set, and $y \in f(S)$ dominated given N. Then Algorithm 1 returns the (unchanged) set $U = U(N) = U(N \cup \{y\})$.*

Proof As $y \in f(S)$ is dominated given N, there exists $y' \in N$ with $y' \leq y$, $y' \neq y$. This implies that $y \notin s(N)$ and by property (i) of $U = U(N)$ being a local upper bound set this implies that there exists no $u \in U(N)$ with $y \in c(u)$. Hence, there exists no $u \in U(N)$ with $y < u$ and for Algorithm 1 this means that $A = \emptyset$. As a result, the algorithm returns the same (unchanged) set $U = U(N)$. We already discussed that this is the same local upper bound set as $U(N \cup \{y\})$, see also [25, Remark 2.2]. □

This holds analogously for Algorithm 2. As already mentioned, new local upper bounds are generated using projections. We briefly explain how this works.

First, all local upper bounds $u \in U(N)$ that are strictly dominated by the update point y are added to the set A. These are the only local upper bounds that are possibly updated

by Algorithm 1. The sets $B_i, i \in [m]$ contain all local upper bounds that are dominated but not strictly dominated by y. The sets $P_i, i \in [m]$ contain the projections of the (old) local upper bounds $u \in A$ to the i-th component of y, i.e., (y_i, u_{-i}). In other words, those sets $P_i, i \in [m]$ contain all candidates for possible new local upper bounds. Then, in the last for loop, redundant candidates are filtered out of each of the sets $P_i, i \in [m]$. Finally, the new local upper bound set $U(N \cup \{y\})$ is computed out of the old set $U(N)$ by removing the old local upper bounds contained in A and adding the (filtered) candidates from the sets $P_i, i \in [m]$.

Thus, for a local upper bound $u \in U(N \cup \{y\})$ it is either $u \in U(N)$ or $u = (y_i, u'_{-i})$ for some $i \in [m]$ and $u' \in U(N)$. For the latter case we call u' the parent of u. Otherwise u is its own parent.

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

Proof If u is its own parent, i.e., $u \in U(N)$, then there is nothing to show. Hence, we consider the case $u \notin U(N)$ and assume that there are two different parents $u^1, u^2 \in A \subseteq U(N)$ of u with $u^1 \neq u^2$. Then there exist $i, j \in [m]$ with $u = (y_i, u^1_{-i}) = (y_j, u^2_{-j})$. If $i = j$ we have $u^1_i \neq u^2_i$ and without loss of generality we assume $u^1_i < u^2_i$. But then $c(u^1) \subset c(u^2)$ which contradicts property (ii) of Definition 3.2 for $U(N)$ to be a local upper bound set. If $i \neq j$ it is $u^2_i = y_i$ and $u^1_j = y_j$ which contradicts $u^1, u^2 \in A$. Thus, there exists only one unique parent u' of u. □

These parents do not only exist for local upper bounds $u \in U(N \cup \{y\})$ but also for the candidates for local upper bounds of $N \cup \{y\}$ contained in the sets $P_i, i \in [m]$ before the filtering step in the last for loop. Of course this holds for updates of the local lower bound set as well.

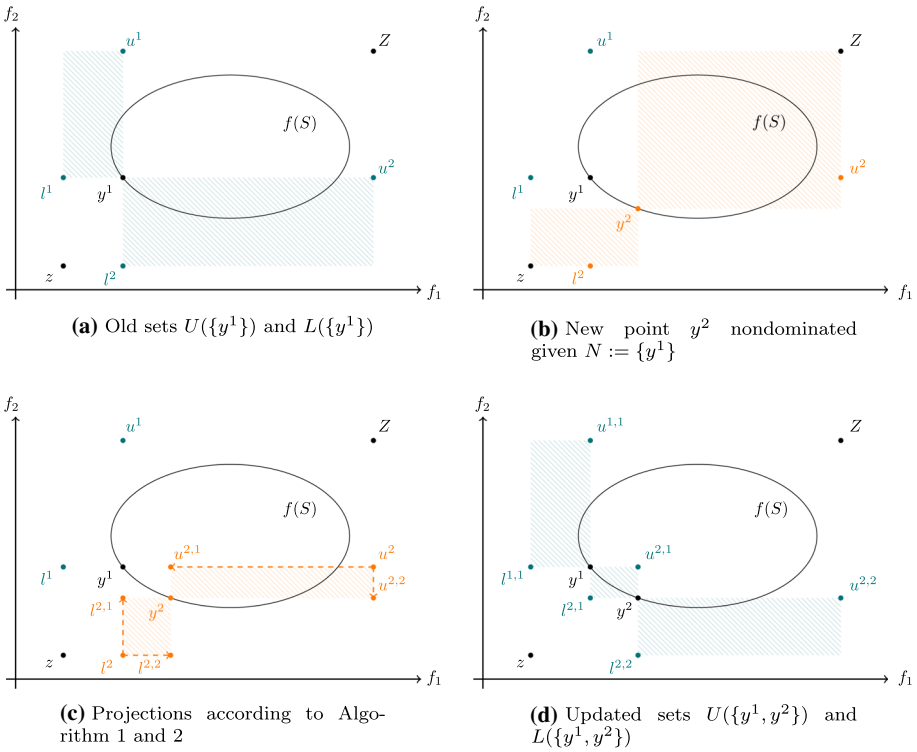
For an illustration of the update procedures, see Fig. 4. For the stable set $N = \{y^1\}$, a local upper bound set $U(N) = \{u^1, u^2\}$, and a local lower bound set $L(N) = \{l^1, l^2\}$ are already computed. The point y^2 is then added to the set N and the bounds are updated using Algorithms 1 and 2. As a result, the local upper bound set is updated to $U(N \cup \{y^2\}) = \{u^1 = u^{1,1}, u^{2,1}, u^{2,2}\}$, and the local lower bound set is updated to $L(N \cup \{y^2\}) = \{l^1 = l^{1,1}, l^{2,1}, l^{2,2}\}$. One can see that l^2 is the parent of $l^{2,1}$ and $l^{2,2}$ and that u^2 is the parent of $u^{2,1}$ and $u^{2,2}$. All remaining bounds (i.e., u^1, l^1) are their own parents. For consistency their names are changed as well. Using this way of assigning a numeration to the local lower and local upper bounds also encodes the parents of the bounds.

4 Computing the box-coverage

In this section, we present our new algorithm to compute an approximation of the nondominated set of (MOP) with a guaranteed improvement in each iteration. The approach is to use local lower bound sets and local upper bound sets as presented in Sect. 3 to compute the approximation in the form of a box-coverage. First, we discuss the initialization of these sets.

4.1 Initialization

We initialize $L = L(\emptyset) = \{z\}$ and $U = U(\emptyset) = \{Z\}$ with $z, Z \in \mathbb{R}^m$ from (2.1). These first bounds should be chosen as tight as possible. For this reason, we use the ideal and anti-ideal



(a) Old sets $U(\{y^1\})$ and $L(\{y^1\})$

(b) New point y^2 nondominated given $N := \{y^1\}$

(c) Projections according to Algorithm 1 and 2

(d) Updated sets $U(\{y^1, y^2\})$ and $L(\{y^1, y^2\})$

Fig. 4 Illustration of update procedure based on Algorithms 1 and 2

points, i.e.,

$$\bar{z} \in \mathbb{R}^m \text{ with } \bar{z}_i = \min\left\{ f_i(x) \;\middle|\; x \in S \right\} \;\forall i \in [m],$$
$$\bar{Z} \in \mathbb{R}^n \text{ with } \bar{Z}_i = \max\left\{ f_i(x) \;\middle|\; x \in S \right\} \;\forall i \in [m].$$

Then, as the ideal and anti-ideal point are not satisfying (2.1), we introduce a small offset $\sigma > 0$ and define with respect to e as the all-ones vector

$$z := \bar{z} - \sigma e \quad \text{and} \quad Z := \bar{Z} + \sigma e. \tag{4.1}$$

Those are now suited as initial local lower and local upper bounds. Both the ideal and the anti-ideal point can be hard to compute. However, any choice of $z, Z \in \mathbb{R}^m$ that satisfies (2.1) can be used for initialization.

4.2 Updating the boxes

Next, we provide a method to shrink boxes $[l, u]$ with $s(l, u) > \varepsilon$. To shrink a box, we need to improve at least one of its bounds l and u. Therefore, our aim is to find a new nondominated point in $[l, u]$. This nondominated point is then chosen as an update point for Algorithms 1 and 2. As a result, the old box $[l, u]$ is replaced by new, smaller boxes, see Fig. 4. In the following, we formalize this approach. Let $l, u \in \mathbb{R}^m$ with $l < u$. Then the search for a

nondominated (update) point is performed by solving the optimization problem

$$\min_{x,t} \; t \quad \text{s.t.} \quad f(x) - l - t(u-l) \leq 0, \quad x \in S, \; t \in \mathbb{R}. \tag{SUP(l,u)}$$

It is crucial for the performance of our algorithm that the (SUP(l,u)) can be solved fast and reliable. This is possible for instance in the case of smooth and convex subproblems (SUP(l,u)). We recommend to take this into account when choosing a solver to solve the subproblems within our overall algorithm. However, the following theoretical results do not require convexity but only continuous objective functions and a compact feasible set S. The following lemma is based on [18, Proposition 2.3.4 and Theorem 2.3.1].

Lemma 4.1 *Let $l, u \in \mathbb{R}^m$ with $l < u$. Then there exists an optimal solution $(\bar{x}, \bar{t})$ for (SUP(l,u)).*

In [33] it is shown that for every optimal solution $(\bar{x}, \bar{t})$ of (SUP(l,u)) the point $\bar{x} \in S$ is a weakly efficient solution for (MOP). Thus, $f(\bar{x})$ is weakly nondominated given $\mathcal{N}$. To perform an update step with Algorithms 1 or 2, i.e., to obtain a new local upper bound or local lower bound set, we need an update point $y \in \mathbb{R}^m$ that is nondominated and not only weakly nondominated. In case all objective functions $f_i, i \in [m]$ are strictly convex and the feasible set S is convex as well, any weakly nondominated point is also nondominated, see [3].

For our general setting, examples on how to find a nondominated point $\bar{y} \leq y$ given $y \in f(S)$ can be found in [4,10]. Another example for bi-objective problems can be found in [5], where the authors used a Pascoletti-Serafini scalarization and encountered the problem of needing to derive a nondominated point of a weakly nondominated point as well. For our implementation, we use a subproblem as formulated in [41]. Let $y \in f(S)$ be a weakly nondominated point of (MOP). Then according to [41, Theorem 2] any optimal solution $\bar{x} \in S$ of

$$\min \sum_{i=1}^{m} f_i(x) \quad \text{s.t.} \quad f(x) \leq y, \; x \in S \tag{GNP(y)}$$

is efficient for (MOP). Thus, $\bar{y} := f(\bar{x})$ is a nondominated point with $\bar{y} \leq y$.

4.3 Main algorithm

Our new algorithm BAMOP to generate a box-coverage is presented as Algorithm 3. The algorithm basically works as follows: It loops through all boxes $[l, u]$ with $l \leq u$ and $s(l,u) > \varepsilon$. Then, a new point to update the bound sets is computed using the methods described above. The whole procedure is repeated until finally all boxes and hence the approximation $\mathcal{A}$ given L and U are sufficiently small, i.e., $w(\mathcal{A}) \leq \varepsilon$.

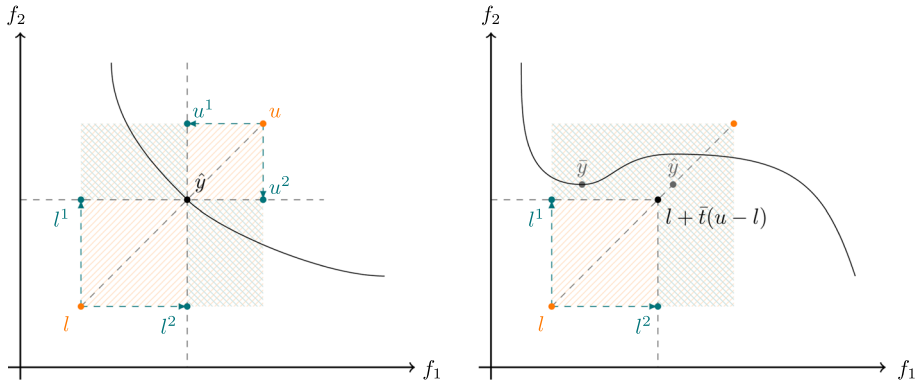
In Algorithm 3 there is a case distinction concerning the computation of the update point for Algorithms 1 and 2. The reason for that is that in order to compute an approximation with $w(\mathcal{A}) \leq \varepsilon$, we need the boxes to get thinner. To do so, we guarantee that at least in one dimension the box length is halved, see Theorem 4.2.

Before we present that theorem, we briefly illustrate a single update step of the algorithm in Fig. 5. Using the notation from Algorithm 3, one case is $\hat{y} = \bar{y}$, i.e., on the connection line between l and u there is a nondominated point $\hat{y} \in \mathcal{N}$. The illustration shows that in this

Algorithm 3 Box Approximation for (MOP)

Input: Tolerances $\varepsilon > 0$ and $\tau > 0$, initial bounds z, Z from (2.1)
Output: Lists L, U of lower and upper bounds

```
procedure BAMOP(ε, τ, z, Z)
  Initialize L = {z}, U = {Z}, done = false
  repeat
    done = true
    L_loop = L
    for l ∈ L_loop do
      if ({l + εe} + int(ℝ^m_+)) ∩ U ≠ ∅ then
        done = false
        U_loop = U
        for u ∈ ({l + εe} + int(ℝ^m_+)) ∩ U_loop do
          Solve (SUP(l, u)) with optimal solution (x̂, t̂)
            and set ŷ := l + t̂(u − l)
          Solve (GNP(y)) with optimal solution x̄ and set ȳ := f(x̄)
          if t̂ > 0.5 and ȳ ≠ ŷ then
            Define step length t̄ := max{0.5, t̂ − τ}
            L = UpdateLLB(L, l + t̄(u − l)))
          else
            L = UpdateLLB(L, ȳ)
            U = UpdateLUB(U, ȳ)
          end if
        end for
      end if
    end for
  until done == true
end procedure
```

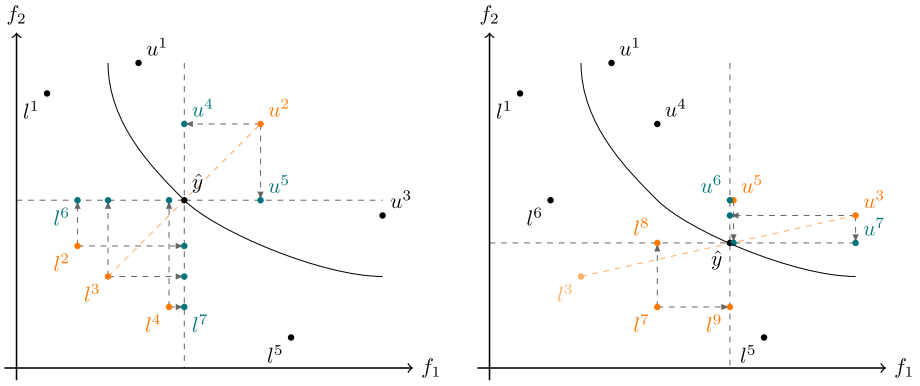


(a) Update case $\bar{y} = \hat{y}$ **(b)** Update case $\hat{t} > 0.5$

Fig. 5 Two update scenarios from Algorithm 3

case the boxes $[l^1, u^1]$ and $[l^2, u^2]$ have indeed at least one dimension that is at most half the length of $[l, u]$.

The case $\hat{t} > 0.5$ with $\hat{y} \neq \bar{y}$ is shown in Fig. 5b. This case corresponds to the if clause from Algorithm 3. Again, we consider the connection line between l and u and by the definition of $\bar{t}$ it holds that $l + \bar{t}(u - l) \geq l + 0.5(u - l) = 0.5(l + u)$. Thus, the new boxes $[l^1, u]$ and $[l^2, u]$ are at most half the length of $[l, u]$ in at least one dimension. Moreover, there is no $y \in f(S)$ with $y \leq a + \bar{t}r$. This is because we use the small offset $\tau > 0$ to ensure $\bar{t} < \hat{t}$ and $(\hat{x}, \hat{t})$



(a) First update step with $u = u^2$ **(b)** Second update step with $u = u^3$

Fig. 6 Two subsequent update steps from Algorithm 3 with $l = l^3$

is an optimal solution of (SUP(l, u)). This can be used together with Corollary 3.6 later in Theorem 4.4 to proof that valid bound sets as needed for Theorem 2.3 are generated. Please be aware that this also implies that, in general, for the local upper bound set $U = U(N^1)$ and the local lower bound set $L = L(N^2)$ in Algorithm 3 it holds that $N^1 \neq N^2$.

Next, we discuss the development of the sets L and U, see Fig. 6. Let $l = l^3 \in L_{\text{loop}}$ be the lower bound selected for the current outer for loop in Algorithm 3. We have $U = U_{\text{loop}} = \{u^1, u^2, u^3\}$ and assume that the inner for loop first selects u^2 and then u^3 (and finally u^1 which is not part of the illustration). During the first run of the inner for loop with $u = u^2$ we find the update point $\hat{y} \in \mathcal{N}$. Using Algorithms 1 and 2, this leads to the projections as shown in Fig. 6. For the lower bounds none of the children of l^3 is part of the updated lower bound set $L = \{l^1, l^6, l^7, l^5\}$. The updated upper bound set is $U = \{u^1, u^4, u^5, u^3\}$.

As the run of the inner for loop with $u = u^2$ is finished, the algorithm continues with the run of the inner for loop with $u = u^3 \in U_{\text{loop}} \neq U$. Even if $l = l^3 \notin L$, it is still the parameter l for (SUP(l, u)). However, for the same reason (i.e., $l^3 \notin L$) it is not considered in the update of the lower bound set L using Algorithm 2. This is an important mechanism to keep in mind. At the end of this run of the inner for loop it is $L = \{l^1, l^6, l^8, l^9, l^5\}$ and $U = \{u^1, u^4, u^6, u^7\}$.

4.4 Halving property and convergence

In this section we proof some important properties of Algorithm 3, such as finiteness and correctness. A key property is presented in the following theorem, which shows that with every run of the repeat loop the width of the boxes is in some sense halved.

Theorem 4.2 (Halving Theorem) *Let $\varepsilon, \tau > 0$ and $z, Z \in \mathbb{R}^m$ be the input parameters for Algorithm 3. Moreover, let L^{start} and U^{start} be the local lower bound and local upper bound sets at the beginning of some iteration of Algorithm 3, i.e., at the beginning of some run of the repeat loop. Accordingly denote by L^{end}, U^{end} the sets at the end of this iteration. Then for every $l^e \in L^{end}$ and every $u^e \in U^{end}$ with $l^e \leq u^e$ there exist $l^s \in L^{start}$ and $u^s \in U^{start}$ with*

1. $l^s \leq l^e \leq u^e \leq u^s$,
2. $(u^e - l^e)_i \leq \max\{\varepsilon,\ 0.5(u^s - l^s)_i\}$ *for at least one index* $i \in [m]$.

Proof Suppose that there exist $l^e \in L^{\text{end}}$ and $u^e \in U^{\text{end}}$ with $l^e \leq u^e$ such that for all $l^s \in L^{\text{start}}$ and $u^s \in U^{\text{start}}$ one of the statements (i) or (ii) is violated.

We denote by $P(l^e)$, $P(u^e) \subseteq \mathbb{R}^m$ the sets containing all elements of the parent history of l^e and u^e within the current iteration, i.e., their parents, their parents parents and so on. By Theorem 3.8 we have that

$$\left|P(l^e) \cap L\right| = 1 \text{ and } \left|P(u^e) \cap U\right| = 1 \tag{4.2}$$

at any point of the current iteration, i.e., for L and U at any point of time. In particular, this holds for the local lower and local upper bound sets at the beginning of the current iteration. Thus we let $l^s \in L^{\text{start}} \cap P(l^e)$ and $u^s \in U^{\text{start}} \cap P(u^e)$.

The update procedures for local upper bound and local lower bound sets, i.e., Algorithms 1 and 2, always ensure that the local lower and local upper bounds are not getting worse. In particular, for $l^c, l^p \in P(l^e)$ and $u^c, u^p \in P(u^e)$ with l^p being the parent of l^c and u^p being the parent of u^c it holds that

$$l^c \geq l^p \quad \text{and} \quad u^c \leq u^p. \tag{4.3}$$

By the definition of the parent history and using that L and U are only updated using Algorithms 1 and 2, this also implies that

$$l^s \leq l \leq l^e \quad \forall l \in P(l^e) \text{ and } u^s \geq u \geq u^e \quad \forall u \in P(u^e). \tag{4.4}$$

In particular, we have $l^s \leq l^e \leq u^e \leq u^s$ and hence (i) is satisfied. Thus, (ii) has to be violated. As a result, it holds that

$$(u^e - l^e)_i > \varepsilon \text{ for all } i \in [m]. \tag{4.5}$$

Using the notation of Algorithm 3, in particular denoting by l and u the corresponding iteration variables of the for loop, at some point of the current run of the repeat loop it is $l = l^s$ because $L_{\text{loop}} = L^{\text{start}}$. Now, fix $l = l^s$ for the outer for loop and consider the inner for loop. By (4.2), we have that there exists a unique $u \in U_{\text{loop}} \cap P(u^e)$. In the following, we consider Algorithm 3 at the point of time where this specific assignment of l and u is present. It is important to mention that u is not necessarily the first upper bound chosen by the inner for loop. For this reason, the sets L and U may have been updated several times which could lead to $l \notin L$ and/or $u \notin U$, see also Fig. 6. However, we know from (4.2) and (4.3) that at this point of the algorithm where l and u are assigned as described above, there exist $l' \in L \cap P(l^e)$ and $u' \in U \cap P(u^e)$ with

$$l' \geq l \text{ and } u' \leq u. \tag{4.6}$$

Together with (4.4), we obtain that

$$l = l^s \leq l' \leq l^e \text{ and } u \geq u' \geq u^e. \tag{4.7}$$

For the remaining part of this proof, we discuss all possible update steps for the fixed assignment of l and u. In total, there are ten cases, see also Fig. 7. In the following, we always refer to the case numbering from that figure.

First, we consider the case (A), i.e., $\hat{t} > 0.5$ and $\bar{y} \neq \tilde{y}$. This case is also shown in Fig. 5b. With $\bar{t} := \max\{0.5, \hat{t} - \tau\}$ it is $l < l + \bar{t}(u - l) =: y'$. We start with (A.1), i.e., $l' < y'$. In this setting the update procedure for L, i.e., Algorithm 2, removes l' and creates m new candidates for lower bounds

$$l^i := (l'_1, \ldots, l'_{i-1}, l_i + \bar{t}(u_i - l_i), l'_{i+1}, \ldots, l'_m)^\top \text{ for all } i \in [m].$$

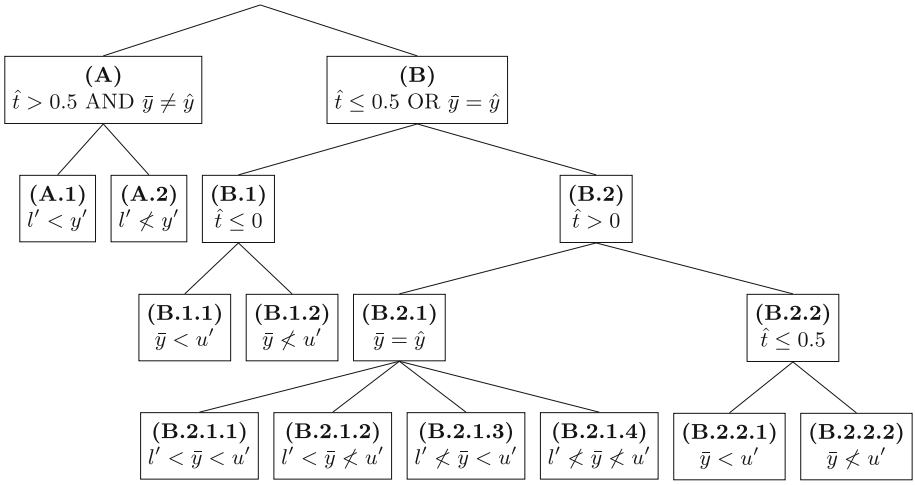


Fig. 7 Case distinction for Theorem 4.2

At least one of these candidates (the child of l' which is part of the parent history $P(l^e)$) is indeed added to the new local lower bound set L. Thus, there exists an index $i \in [m]$ such that $l^i \in L$ after executing Algorithm 2. By (4.4) we have that $l^e \geq l^i$ and together with (4.7) and again (4.4) we obtain that

$$(u^e - l^e)_i \leq (u - l^i)_i = u_i - l_i - \bar{t}(u - l)_i = (1 - \bar{t})(u - l)_i \leq 0.5(u - l)_i \leq 0.5(u^s - l^s)_i$$

As a result, (ii) would be satisfied which contradicts our assumption.

Next, we consider case (A.2), i.e., $l' \not< y'$, which implies that there exists an index $i \in [m]$ with $l'_i \geq y'_i$. Using (4.7), this implies again with (4.4) that

$$(u^e - l^e)_i \leq (u^e - l')_i \leq u_i - y'_i = u_i - l_i - \bar{t}(u - l)_i \leq 0.5(u - l)_i \leq 0.5(u^s - l^s)_i$$

which contradicts our assumption as well.

This concludes case **(A)**. For case (B) let $\hat{t} \leq 0.5$ or $\bar{y} = \hat{y}$. First we discuss case (B.1), this is, $\hat{t} \leq 0$. In this case it is $\bar{y} \leq l \leq l'$. By Theorem 3.4 (ii) there is also no $l^* \in L$ with $l^* \leq l'$, $l^* \neq l'$ and in particular no $l^* \in L$ with $l^* < \bar{y}$. Thus, the local lower bound set L is not updated in this step and only the local upper bound set has to be considered.

In case (B.1.1) with $\bar{y} < u'$, Algorithm 1 removes the upper bound u' from the local upper bound set and creates m new candidates

$$u^i := (u'_1, \ldots, u'_{i-1}, \bar{y}_i, u'_{i+1}, \ldots, u'_m)^\top \text{ for all } i \in [m].$$

At least one of these candidates (as part of the parent history $P(u^e)$) is added to the updated set of local upper bounds. Thus, there exists an index $i \in [m]$ such that $u^i \in U$ after the updating procedure. Using (4.4), we obtain

$$(u^e - l^e)_i \leq (u^i - l)_i = \bar{y}_i - l_i \leq 0 < \varepsilon.$$

Again, (ii) would be satisfied, which contradicts our assumption.

Next, we consider case (B.1.2). As $\bar{y} \not< u'$, there exists an index $i \in [m]$ such that $\bar{y}_i \geq u'_i$. Using (4.7), this leads to

$$(u^e - l^e)_i \leq (u' - l)_i \leq \bar{y}_i - l_i \leq 0 < \varepsilon$$

and contradicts our assumption that (ii) is not satisfied.

This concludes case (B.1) and we continue with case (B.2). Consequently, for the remaining part of this proof we have that $\hat{t} > 0$. We start with case (B.2.1), i.e., $\bar{y} = \hat{y}$, see Fig. 5a. First, we consider case (B.2.1.1) with $l' < \bar{y} < u'$. In this setting, the updating procedures Algorithms 1 and 2 remove l' and u' from L and U and create m new candidates

$$l^i := (\bar{y}_i, l'_{-i}),\ u^i := (\bar{y}_i, u'_{-i}) \text{ for all } i \in [m]. \tag{4.8}$$

Again, at least one of these candidates is contained in the updated local lower and local upper bound sets because it is part of the parent history $P(l^e)$ or $P(u^e)$, respectively. Hence, there exist $i, j \in [m]$ with $l^i \in L$ and $u^j \in U$ after the updating procedures. For $i = j$ we have $(u^j - l^i)_i = \bar{y}_i - \bar{y}_i = 0 < \varepsilon$ which contradicts our assumption. Thus, we only consider $i \neq j$. Using (4.4), (4.7), and $\bar{y} = \hat{y}$, we obtain

$$\begin{aligned}
(u^e - l^e)_j &\leq (u^j - l^i)_j = l_j + \hat{t}(u-l)_j - l'_j \leq l_j + \hat{t}(u-l)_j - l_j = \hat{t}(u-l)_j, \\
(u^e - l^e)_i &\leq (u^j - l^i)_i = u'_i - l_i - \hat{t}(u-l)_i \leq u_i - l_i - \hat{t}(u-l)_i = (1-\hat{t})(u-l)_i.
\end{aligned}$$

It is either $\hat{t} \leq 0.5$ or $(1 - \hat{t}) < 0.5$. Hence, there exists $\iota \in \{i, j\}$ with

$$(u^e - l^e)_\iota \leq 0.5(u-l)_\iota \leq 0.5(u^s - l^s)_\iota$$

which contradicts our assumption.

Next, we consider case (B.2.1.2) with $l' < \bar{y} \not< u'$. Consequently, there exists an index $j \in [m]$ with $\bar{y}_j \geq u'_j$. For the set L, we know that l' is removed and Algorithm 2 computes m new candidates, see (4.8). Again, there exists an index $i \in [m]$ such that $l^i \in L$ after the updating procedure. For $i = j$ we have that $(u' - l^j)_j = u'_j - \bar{y}_j \leq 0$ which contradicts our assumption. For $i \neq j$, using (4.4), (4.7), and $\bar{y} = \hat{y}$, we obtain

$$\begin{aligned}
(u^e - l^e)_j &\leq (u' - l^i)_j \leq l_j + \hat{t}(u-l)_j - l'_j \leq l_j + \hat{t}(u-l)_j - l_j = \hat{t}(u-l)_j, \\
(u^e - l^e)_i &\leq (u' - l^i)_i \leq u'_i - l_i - \hat{t}(u-l)_i \leq u_i - l_i - \hat{t}(u-l)_i = (1-\hat{t})(u-l)_i.
\end{aligned}$$

Again, it is either $\hat{t} \leq 0.5$ or $(1 - \hat{t}) < 0.5$ and there exists $\iota \in \{i, j\}$ with

$$(u^e - l^e)_\iota \leq 0.5(u-l)_\iota \leq 0.5(u^s - l^s)_\iota$$

which contradicts our assumption.

Case (B.2.1.3), i.e., $l' \not< \bar{y} < u'$, can be shown analogously. There exists an index $i \in [m]$ with $l'_i \geq \bar{y}_i$. Moreover, u' is updated by Algorithm 1. As a result, m new candidates for local upper bounds are computed, see (4.8). Again, there exists an index $j \in [m]$ such that $u^j \in U$ after the updating procedure. For $i = j$ we have that $(u^i - l')_i = \bar{y}_i - l'_i \leq 0$ which contradicts our assumption. For $i \neq j$, using (4.4), (4.7), and $\bar{y} = \hat{y}$, we obtain

$$\begin{aligned}
(u^e - l^e)_j &\leq (u^j - l')_j \leq l_j + \hat{t}(u-l)_j - l'_j \leq l_j + \hat{t}(u-l)_j - l_j = \hat{t}(u-l)_j, \\
(u^e - l^e)_i &\leq (u^j - l')_i \leq u'_i - l_i - \hat{t}(u-l)_i \leq u_i - l_i - \hat{t}(u-l)_i = (1-\hat{t})(u-l)_i.
\end{aligned}$$

Consequently, there exists $\iota \in \{i, j\}$ with

$$(u^e - l^e)_\iota \leq 0.5(u-l)_\iota \leq 0.5(u^s - l^s)_\iota$$

which contradicts our assumption.

Now, we consider case (B.2.1.4) with $l' \not< \bar{y} \not< u'$. In this case, there exist $i, j \in [m]$ with $l'_i \geq \bar{y}_i$ and $\bar{y}_j \geq u'_j$. For $i = j$ this implies $(u' - l')_i \leq 0$ and (ii) would be satisfied. This

contradicts our assumption. For $i \neq j$, using (4.7) and $\bar{y} = \hat{y}$ we have that

$$(u^e - l^e)_j \leq (u' - l')_j \leq l_j + \hat{t}(u-l)_j - l'_j \leq l_j + \hat{t}(u-l)_j - l_j = \hat{t}(u-l)_j,$$
$$(u^e - l^e)_i \leq (u' - l')_i \leq u'_i - l_i - \hat{t}(u-l)_i \leq u_i - l_i - \hat{t}(u-l)_i = (1-\hat{t})(u-l)_i.$$

Again, there exists $\iota \in \{i, j\}$ with

$$(u^e - l^e)_\iota \leq 0.5(u-l)_\iota \leq 0.5(u^s - l^s)_\iota$$

which contradicts our assumption.

This concludes case (B.2.1) and we continue with case (B.2.2), i.e., $\hat{t} \leq 0.5$. In case (B.2.2.1), i.e., $\bar{y} < u'$, the upper bound u' is still removed (and m candidates created) as in (4.8). Again, there exists $j \in [m]$ such that $u^j \in U$ after the update procedure. Thus, using (4.4), (4.7), and that $\bar{y} \leq \hat{y}$, we have that

$$\begin{aligned}(u^e - l^e)_j &\leq (u^j - l')_j \leq \bar{y}_j - l_j \leq l_j + \hat{t}(u-l)_j - l_j \\ &= \hat{t}(u-l)_j \leq 0.5(u-l)_j \leq 0.5(u^s - l^s)_j\end{aligned}$$

which contradicts our assumption.

Finally, we consider case (B.2.2.2) with $\bar{y} \not< u'$. Then there exists an index $j \in [m]$ with $\bar{y}_j \geq u'_j$. Using the same arguments as before, this implies that

$$\begin{aligned}(u^e - l^e)_j &\leq (u' - l')_j \leq \bar{y}_j - l_j \leq l_j + \hat{t}(u-l)_j - l_j \\ &= \hat{t}(u-l)_j \leq 0.5(u-l)_j \leq 0.5(u^s - l^s)_j\end{aligned}$$

which contradicts our assumption as well.

We considered all cases shown in Fig. 7. For all these cases, our assumption that (ii) is not satisfied does not hold. Hence, we have to reject our assumption and Theorem 4.2 is shown. □

With Theorem 4.2 we can show that Algorithm 3 is finite. In particular, the next lemma provides an upper bound on the number of iterations (i.e., the number of runs of the repeat loop). This bound depends on $z, Z \in \mathbb{R}^m$ from (2.1) and thus they should be chosen tight as discussed on page 12.

Lemma 4.3 *Let $\varepsilon, \tau > 0$ and $z, Z \in \mathbb{R}^m$ be the input parameters for Algorithm 3. Moreover, define*

$$\Delta := \|Z - z\|_\infty \quad \text{and} \quad \kappa := \left\lceil m \log_2 \left(\frac{\Delta}{\varepsilon} \right) \right\rceil + 1.$$

Then the number of iterations, i.e., the number of runs of the repeat loop, is bounded by $\max\{\kappa, 1\}$*, and hence, Algorithm 3 is finite.*

Proof If $\kappa \leq 1$ then $\Delta \leq \varepsilon$ and Algorithm 3 ends after the first iteration.

Hence, we consider the case $\kappa > 1$ and define $l^1 := z$, $u^1 := Z$ and for $k \geq 1$ let L^k, U^k be the local lower bound and local upper bound sets at the beginning of iteration k. By Theorem 4.2, for all $k > 1$, all $l^k \in L^k$, and all $u^k \in U^k$ there exist $l^{k-1} \in L^{k-1}$, $u^{k-1} \in U^{k-1}$, and an index $i^k \in [m]$ such that

$$(u^k - l^k)_{i^k} \leq \max\{\varepsilon,\ 0.5(u^{k-1} - l^{k-1})_{i^k}\}. \tag{4.9}$$

Suppose that Algorithm 3 has more than κ iterations. Then there exist $l^\kappa \in L^\kappa$ and $u^\kappa \in U^\kappa$ with $l^\kappa + \varepsilon e < u^\kappa$ (being equivalent to $\varepsilon e < u^\kappa - l^\kappa$). For every $k \in \{2, 3, \ldots, \kappa\}$ let $i^k \in [m]$ be the index from the iterative application of (4.9) starting with $k = \kappa$. We define

$$n(i) := \left| \left\{ k \in \{2, 3, \ldots, \kappa\} \mid i^k = i \right\} \right| \text{ for all } i \in [m].$$

As $\kappa - 1 = \left\lceil m \log_2 \left(\frac{\Delta}{\varepsilon}\right) \right\rceil$ there exists at least one index $\bar{i} \in [m]$ with $n(\bar{i}) \geq \log_2 \left(\frac{\Delta}{\varepsilon}\right)$. Iteratively using (4.9), this implies that

$$(u^\kappa - l^\kappa)_{\bar{i}} \leq 2^{-\log_2\left(\frac{\Delta}{\varepsilon}\right)} (u^1 - l^1)_{\bar{i}} \leq \frac{\varepsilon}{\Delta} \Delta = \varepsilon$$

which contradicts $\varepsilon e < u^\kappa - l^\kappa$. As a result, there cannot be more than κ iterations and Algorithm 3 is finite. □

So far, we have shown that Algorithm 3 generates a local lower bound set L and a local upper bound set U and stops after finitely many iterations. Next, we use Theorem 3.6 to show that these sets L and U are in fact lower and upper bound sets in the sense of Theorem 2.3.

Lemma 4.4 *The output sets L and U of Algorithm 3 are lower bound and upper bound sets in the sense of Theorem 2.3.*

Proof As the output set U is a local upper bound set and L is a local lower bound set, there exist $N^1, N^2 \subseteq \text{int}(B)$ with $U = U(N^1)$ and $L = L(N^2)$. First, we discuss the local upper bound set U. The algorithm starts with $U(\emptyset) = \{Z\}$. After that, the local upper bound set is only updated using points $\bar{y} = f(\bar{x}) \in \mathcal{N}$ computed from optimal solutions $\bar{x}$ of (GNP($f(\hat{x})$). As a result, it is $N^1 \subseteq \mathcal{N}$ (or $N^1 = \emptyset$) and in particular $N^1 \subseteq f(S)$.

Next, we consider the local lower bound set L. The algorithm starts with $L(\emptyset) = \{z\}$. For all further updates of the set there are two cases. The first case is that L is updated within the if part of the if clause, i.e., using $y := l + \bar{t}(u - l)$ with the notation from Algorithm 3. Thereby, it holds that $\bar{t} < \hat{t}$ with $(\hat{x}, \hat{t})$ being an optimal solution of (SUP(l, u)). Thus, there exists no $y' \in f(S)$ with $y' \leq y$ which implies $y \notin f(S) + \mathbb{R}^m_+$. The second case is that L is updated by Algorithm 2 using an update point $\bar{y} = f(\bar{x}) \in \mathcal{N}$ computed from an optimal solution $\bar{x}$ of (GNP($f(\hat{x})$). Again, it is $\bar{y} \notin f(S) + (\mathbb{R}^m_+ \backslash \{0\})$. Hence, we have $N^2 \subseteq \text{int}(B) \backslash (f(S) + (\mathbb{R}^m_+ \backslash \{0\}))$ (or $N^2 = \emptyset$).

By Theorem 4.3 we have that Algorithm 3 is finite and so N^1 and N^2 are finite sets as well. Moreover, without loss of generality, we can assume that the sets N^1 and N^2 are stable, see page 10 or [25, Remark 2.2]. Together with Theorem 3.6 this implies that L and U are lower and upper bound sets in the sense of Theorem 2.3. □

Finally, we can use all of the results from above to show that Algorithm 3 generates an approximation $\mathcal{A}$ of the nondominated set $\mathcal{N}$ of (MOP) with width $w(\mathcal{A}) \leq \varepsilon$.

Theorem 4.5 *Let $\varepsilon, \tau > 0$ and $z, Z \in \mathbb{R}^m$ be the input parameters and $L, U \subseteq \mathbb{R}^m$ be the output sets of Algorithm 3. Then $\mathcal{A} = (L + \mathbb{R}^m_+) \cap (U - \mathbb{R}^m_+)$ is an approximation of the nondominated set $\mathcal{N}$ of (MOP) with $w(\mathcal{A}) \leq \varepsilon$.*

Proof By Lemma 4.4 the sets L and U are lower and upper bound sets in the sense of Theorem 2.3. Hence, $\mathcal{A}$ is an approximation of the nondominated set $\mathcal{N}$ given L and U. Moreover, we have shown in Theorem 4.3 that Algorithm 3 is finite, i.e., it terminates after finitely many iterations. This implies that at some point the repeat loop begins and ends with done == true.

Thus, for all $l \in L$ the if clause is not satisfied, i.e., for all $u \in U$ there exists at least one index $i \in [m]$ with

$$l_i + \varepsilon \not< u_i \;\Leftrightarrow\; \varepsilon \geq u_i - l_i.$$

As a result, the approximation $\mathcal{A}$ has a width $w(\mathcal{A}) \leq \varepsilon$. □

5 Numerical results

In the following, we present numerical results for selected test instances. All numerical tests have been performed using MATLAB R2018b on a machine with Intel Core i9-10920X processor. The average of the results of `bench(5)` is: LU = 0.0470, FFT = 0.0559, ODE = 0.0137, Sparse = 0.0885, 2-D = 0.2302, 3-D = 0.2117. It provides a rough idea of the machine performance, in particular for comparing the results using other machines. Please be aware that these results are version specific according to the MATLAB documentation, see [29].

In this section we mainly focus on convex problems (MOP), i.e., such problems with convex objective functions $f_i, i \in [m]$ and a convex set $S \subseteq \mathbb{R}^n$. The big advantage of this setting is that also all single-objective subproblems like (SUP(l, u)) are convex problems and can be solved by any local solver. To solve (SUP(l, u)) and (GNP(y)) we use the local solver `fmincon`. For the initialization, we provide z, Z as in (4.1) with $\sigma = 10^{-6}$ unless stated otherwise. We also fix $\tau = 10^{-6}$ for Algorithm 3. We also discuss two nonconvex examples in this section. In case we only use a local solver for the subproblems we will no longer have a guarantee to obtain a valid approximation for those.

First, in Sect. 5.1 we present the numerical results for BAMOP as given in Algorithm 3. Then, in Sect. 5.2, we present and discuss the results for a modified version of the algorithm. Please be aware that while our algorithm works for problems (MOP) with an arbitrary number of objective functions, in this section we focus on bi- and tri-objective examples since for those examples we can also provide a visualization of the results. In Test Instance 4 we discuss the results for an optimization problem with more than three objectives. The generated data and the MATLAB files are available from the authors on reasonable request.

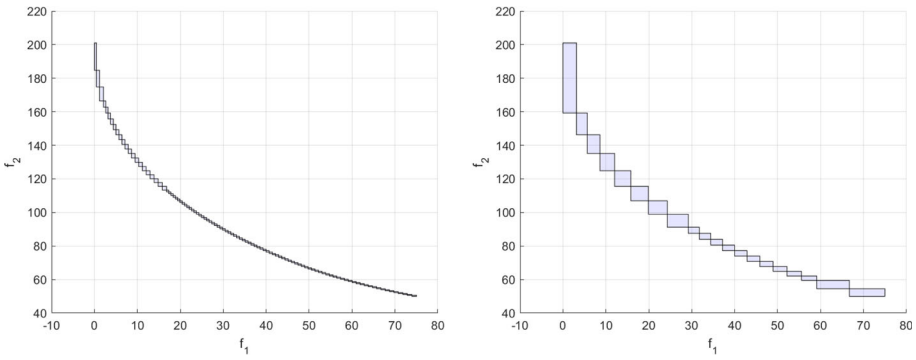
5.1 Results for BAMOP

Test Instance 1 First, we consider a convex, box-constrained problem. The second objective function is the so-called Brent function, see [2,22].

$$\min \; f(x) := \begin{pmatrix} 2x_1^2 + x_2^2 \\ (x_1+10)^2 + (x_2+10)^2 + e^{(-x_1^2 - x_2^2)} \end{pmatrix} \;\text{ s.t. }\; x \in [-5, 0]^2. \tag{Br}$$

We execute the algorithm for $\varepsilon = 1$ and $\varepsilon = 5$. The results are shown in Fig. 8. For $\varepsilon = 1$ the structure of the nondominated set is easily recognizable. However, this is also the case for $\varepsilon = 5$.

The computation time is about 0.59 seconds for $\varepsilon = 5$ and about 2.33 seconds for $\varepsilon = 1$. Hence, the quality improvement from $\varepsilon = 5$ to $\varepsilon = 1$ requires (roughly) four times the computation time. We further investigate the computation time in the next test instance.



(a) Result for $\varepsilon = 1$ **(b)** Result for $\varepsilon = 5$

Fig. 8 Numerical results for (Br)

Table 1 Numerical results for (Jin1)

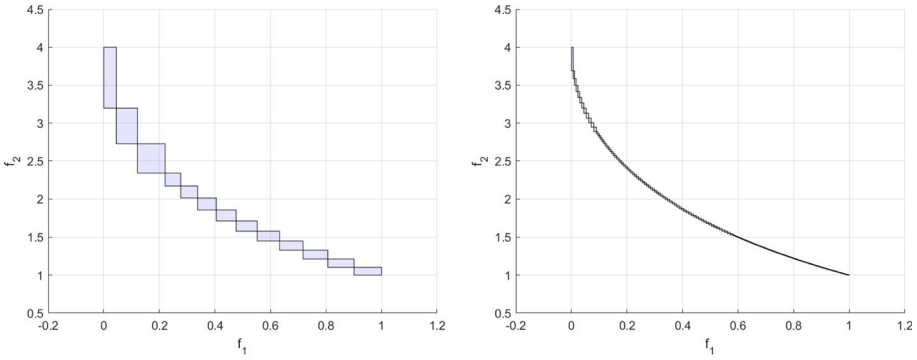
n	ε	Time (Total)	# Calls `fmincon`	Time `fmincon`
2	0.1	0.751	24	0.681
10	0.1	0.592	24	0.521
50	0.1	1.251	24	1.178
2	0.01	6.162	300	5.886
10	0.01	7.320	300	7.048
50	0.01	14.659	300	14.358
2	0.001	59.462	2928	57.084
10	0.001	116.206	2928	113.777
50	0.001	265.222	2928	262.683

Test Instance 2 For the next example, we consider a scalable test problem from [23] that was also used in [39].

$$\min \; f(x) := \begin{pmatrix} \frac{1}{n}\sum_{i=1}^{n} x_i^2 \\ \\ \frac{1}{n}\sum_{i=1}^{n} (x_i - 2)^2 \end{pmatrix} \quad \text{s.t.} \quad x \in [0,1]^n. \tag{Jin1}$$

The results, in particular the computation time, for different choices of n and ε are shown in Table 1. Those are computed using MATLAB's Run and Time feature. This causes some overhead which should be taken into account when comparing results. Table 1 shows that increasing n or ε also increases the computation time. Also, the overall computation time of Algorithm 3 is basically the computation time for solving the subproblems (SUP(l, u)) and (GNP(y)), i.e., the time used for `fmincon` in our example. Thus, it is important to carefully choose a suitable solver for these subproblems.

In Fig. 9 the results for $\varepsilon = 0.1$ and $\varepsilon = 0.01$ are shown. Together with the results shown in Table 1 this motivates that the parameter ε should be chosen carefully and with respect to the scale of the optimization problem.



(a) Result for $\varepsilon = 0.1$ **(b)** Result for $\varepsilon = 0.01$

Fig. 9 Numerical results for (Jin1)

Table 2 Numerical results for (Ex5.1) with $\varepsilon = 0.1$

a	5	7	10	20
Time	65.0054	80.6611	93.6614	123.2805

Test Instance 3 The following tri-objective example can be found in [8] and depends on a parameter $a > 0$.

$$\min \; f(x) := x \quad \text{s.t.} \quad \left(\frac{x_1 - 1}{1}\right)^2 + \left(\frac{x_2 - 1}{a}\right)^2 + \left(\frac{x_3 - 1}{5}\right)^2 \leq 1. \tag{Ex5.1}$$

The parameter a controls the scale of the ellipsoid constraint function and, hence, the steepness of the nondominated set. In [8] the authors present an approach of vertex selection for a polyhedral approximation algorithm from [28]. They demonstrate that their modification's computation time is only slightly effected by the choice of a.

In contrast, for our algorithm the choice of a has a noticeable influence on the computation time, see Table 2. However, this is not surprising. Increasing the parameter a also increases the size of the initial box $[z, Z]$. In particular, we have $z_2 = 1 - a - \sigma$ and $Z_2 = 1 + a + \sigma$. Thus, we can expect that more subboxes need to be computed to generate an approximation $\mathcal{A}$ with $w(\mathcal{A}) \leq \varepsilon = 0.1$.

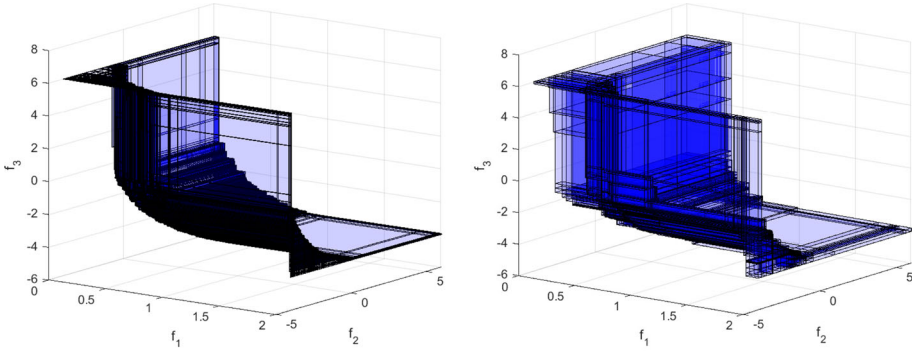
For an illustration of the result for $a = 5$, see Fig. 10. Besides the illustration for $\varepsilon = 0.1$ we also included the result for $\varepsilon = 0.5$ where the box structure can be seen more clearly.

Test Instance 4 As mentioned in the beginning of this section, our algorithm is not restricted to bi- and tri-objective problems that we use here mainly to be able to visually represent the numerical results. The following test instance is convex and has a scalable number $m \leq n$ of objectives. It can be found in various formulations, for example in [20,39]. The objective functions $f_j \colon \mathbb{R}^n \to \mathbb{R}$, $j \in [m]$ are given by

$$f_j(x) := (x_j - 1)^2 + \sum_{i \in [n], i \neq j} x_i^2.$$

The test instance is then given as

$$\min \; f(x) \quad \text{s.t.} \quad x \in [-1000, 1000]^n, \; f_j(x) \leq 1 \, \forall j \in [m]. \tag{ZLT1}$$



(a) Result for $\varepsilon = 0.1$ **(b)** Result for $\varepsilon = 0.5$

Fig. 10 Numerical results for (Ex5.1)

We added the constraints $f_j(x) \leq 1$, $j \in [m]$ to allow a tight initial box $[z, Z]$ with $Z_j := 1 + \sigma$ for all $j \in [m]$. Adding these constraints has no impact on the nondominated set of (ZLT1) since even without these constraints the nondominated set is contained in $[0, 1]^m$. For all choices of m we fixed $\varepsilon = 0.2$ and $n = 100$.

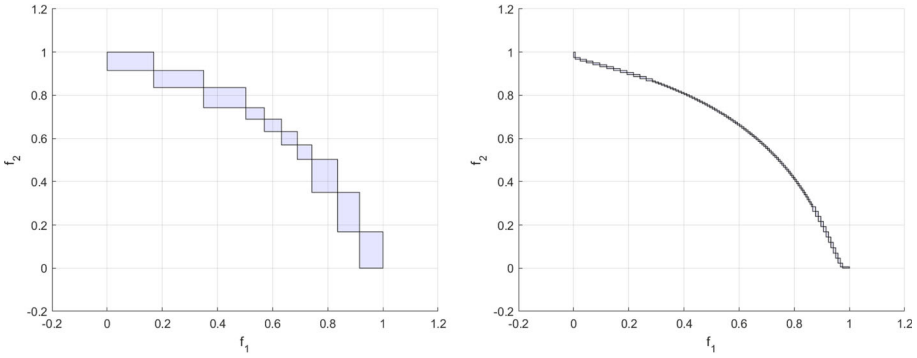
For $m = 2$ and $m = 3$ the computation times are 2.46 and 4.64 seconds. But then, with increasing m, they increase quite fast to 15.05 seconds for $m = 4$ and 122.25 seconds for $m = 5$. When choosing $\varepsilon = 0.1$ we only need 22.90 seconds for $m = 3$ but more than 3600 seconds (which we used as a time limit) for $m = 4$.

This example demonstrates that due to the fact that our algorithm is a criterion space algorithm, its computation time highly depends on the number of objectives. Moreover, the computation time increases drastically with the number of objectives in this example. This might seem surprising, since by Theorem 4.3 it is known that the maximal number of iterations of our algorithm depends linearly on the number of objective functions. The reason for the increase in computation time is that, in general, with each iteration the number of lower and upper bounds increases which implies that the iterations get more time consuming.

Test Instance 5 The next example is scalable in the decision space and nonconvex. It can be found in [20,39,40] and is a generalization of an example from [17].

$$\min \; f(x) := \begin{pmatrix} 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right) \\ 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right) \end{pmatrix} \quad \text{s.t.} \quad x \in [-4, 4]^n. \tag{MOP2}$$

As already mentioned at the beginning of this section, we keep using the local solver `fmincon` to solve all single-objective (nonconvex) subproblems. As a result, there is no longer a guarantee that our algorithm computes a valid approximation of the nondominated set $\mathcal{N}$. To ensure that all theoretical results hold, a suitable (global) solver for the nonconvex subproblems (SUP(l, u)) and (GNP(y)) would be necessary. However, for this example it turns out that our algorithm is still computing a valid result. This is also the case for other examples which are only "slightly" nonconvex. If one aims for a guaranteed valid approximation of the nondominated set, algorithms as those presented in [14,31,35,42,43] should be considered. Note that the algorithms from [14], [31] and [35] also have the advantage that



(a) Result for $\varepsilon = 0.1$ **(b)** Result for $\varepsilon = 0.01$

Fig. 11 Numerical results for (MOP2)

they compute not only an approximation of the nondominated set but also a representation or coverage of the set of (ε-) efficient solutions. On the other hand, these algorithms are often limited in the size of n.

For the dimension we choose $n = 100$. In Fig. 11 the results for $\varepsilon = 0.1$ and for $\varepsilon = 0.01$ are shown. The computation time for $\varepsilon = 0.1$ is about 1.8 s while for $\varepsilon = 0.01$ it is about 24.3 s. Considering Fig. 11, one has to decide whether the increase in computation time is worth the improvement of the result.

Test Instance 6 While in the previous example our algorithm delivered a valid approximation of the nondominated set, we now present another nonconvex test instance from [37] where our algorithm combined with the local solver `fmincon` failed to provide a valid approximation. The test instance is given as

$$\begin{aligned} \min \ f(x) := x \quad \text{s.t.} \quad & -x_1^2 - x_2^2 + 1 + 0.1\cos\left(16\arctan\left(\frac{x_1}{x_2}\right)\right) \leq 0, \\ & (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5, \\ & x \in [0, \pi]^2. \end{aligned} \tag{Tan}$$

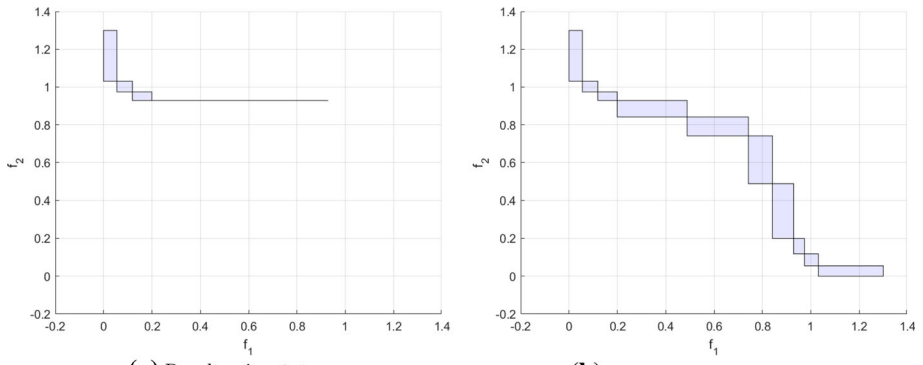
We set the tolerance $\varepsilon = 0.1$ and for the initial box $[z, Z]$ we choose $z = (-\sigma, -\sigma)^\top$ and $Z = (1.3 + \sigma, 1.3 + \sigma)^\top$.

The result for using `fmincon` is shown in Fig. 12 on the left. On the right the result for using `GlobalSearch` instead of `fmincon` to solve the subproblems is shown. This actually returns a valid approximation of the nondominated set of (Tan). Hence, in this example more effort is needed to obtain a valid approximation by BAMOP.

5.2 Results for BAMOP with selection criterion

For this section, we consider a modification of BAMOP (Algorithm 3). More precisely, we replace the inner for loop for the upper bounds by a selection criterion for a single upper bound. For the selection criterion, we use the shortest edge $s(l, u)$. More precisely, given $l \in L$ we select an upper bound $u \in (\{l + \varepsilon e\} + \text{int}(\mathbb{R}^m_+)) \cap U_{\text{loop}}$ with

$$s(l, u) \geq s(l, u') \text{ for all } u' \in (\{l + \varepsilon e\} + \text{int}(\mathbb{R}^m_+)) \cap U_{\text{loop}}.$$



(a) Result using `fmincon` **(b)** Result using `GlobalSearch`

Fig. 12 Numerical results for (Tan)

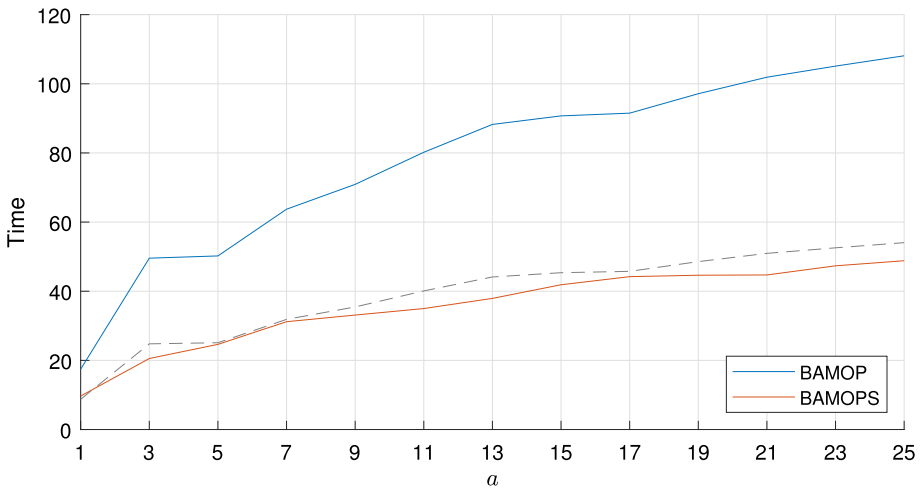


Fig. 13 Comparison of BAMOP and BAMOPS for (Ex5.1)

We refer to this modified version of Algorithm 3 as BAMOPS. Please be aware that for this modification finiteness of the algorithm is no longer guaranteed by Theorem 4.3. However, if BAMOPS terminates then for the computed approximation $\mathcal{A}$ it still holds $w(\mathcal{A}) \leq \varepsilon$.

We consider again the tri-objective optimization problem (Ex5.1) from Test Instance 3. For fixed $\varepsilon = 0.1$ we compare the computation time of BAMOP and BAMOPS for $a \in \{1, 3, 5, 7, \ldots, 25\}$. The results are presented in Fig. 13. For better comparison, the dashed line represents half the computation time of BAMOP. Thus, for this specific instance BAMOPS is roughly twice as fast as BAMOP. Please be aware that the computation time of BAMOP for Test Instance 3 on page 24 is longer because of the overhead introduced by the Run and Time feature.

Also for other test instances with $m > 2$ we obtained a significant reduction of the computation time when using BAMOPS instead of BAMOP.

6 Conclusions

We presented a new algorithm to compute an approximation of the nondominated set of (MOP) in the form of a box-coverage. This algorithm has been proven to compute an approximation $\mathcal{A}$ with width $w(\mathcal{A})$ of at most ε for any predefined $\varepsilon > 0$ after finitely many iterations. In particular, we have shown that an improvement in every iteration can be guaranteed (see Theorem 4.2) and provided an upper bound for the number of iterations needed (see Lemma 4.3).

Algorithm 3 can theoretically be used for a huge class of problems. Practically, one needs to be able to solve the single-objective subproblems fast and reliably. This is possible for instance for smooth convex optimization problems for which such solvers are available. Even for nonconvex problems our algorithm is able to obtain valid approximations when using a local solver for the subproblems as demonstrated in Sect. 5. If one wants to guarantee that BAMOP computes a valid approximation in the nonconvex case, a suitable global solver is necessary. In this case one might also consider using other approaches like those from [12], [14], [31] and [42]. In Sect. 5.1 we illustrated the algorithm on six different test instances. As the results have shown, the parameter ε should be chosen carefully and with respect to the range of the objective values. There is a significant trade-off between the approximation quality, i.e., the choice of ε, and the computation time for the algorithm.

We found out that the computation time of Algorithm 3 mainly depends on the computation time that is needed for solving the subproblems (SUP(l, u)) and (GNP(y)). Thus, an interesting question for further research would be how often these subproblems have to be solved. This could then be used to estimate the overall computation time. In particular, a relation between the amount of decreasing the value of ε and the resulting increase in the number of evaluations of the subproblems would be of great interest.

Moreover, the modification of Algorithm 3 that we presented in Sect. 5.2 seems to be promising in view of the computation time. However, we loose the Halving Theorem with this modification. Still, it should be further investigated. In particular, one should check if this modification can still be proofed to terminate after finitely many iterations. Also, working with other selection criteria, e.g., based on the Hypervolume Indicator [44], should be examined.

Acknowledgements This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—Project-ID 432218631. We thank the two anonymous referees for their remarks and comments that helped us to improve the quality of this paper.

References

1. Bokrantz, R., Forsgren, A.: An algorithm for approximating convex pareto surfaces based on dual techniques. INFORMS J. Comput. **25**(2), 377–393 (2013)

2. Branin, F.H.: Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. IBM J. Res. Dev. **16**(5), 504–522 (1972)
3. Burachik, R.S., Kaya, C.Y., Rizvi, M.M.: A new scalarization technique to approximate pareto fronts of problems with disconnected feasible sets. J. Optim. Theory Appl. **162**(2), 428–446 (2013)
4. Burachik, R.S., Kaya, C.Y., Rizvi, M.M.: A new scalarization technique and new algorithms to generate pareto fronts. SIAM J. Optim. **27**(2), 1010–1034 (2017)
5. Doğan, S., Özlem, K., Ulus, F.: An exact algorithm for biobjective integer programming problems (2019). https://arxiv.org/abs/1905.07428
6. Dächert, K., Klamroth, K.: A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems. J. Glob. Optim. **61**(4), 643–676 (2014)
7. Dächert, K., Teichert, K.: An improved hyperboxing algorithm for calculating a Pareto front representation (2020). https://arxiv.org/abs/2003.14249
8. Dörfler, D., Löhne, A., Schneider, C., Weißing, B.: A Benson-type algorithm for bounded convex vector optimization problems with vertex selection (2020). http://arxiv.org/abs/2006.15600
9. Ehrgott, M.: Multicriteria Optimization. Springer (2005)
10. Ehrgott, M., Ruzika, S.: Improved ε-constraint method for multiobjective programming. J. Optim. Theory Appl. **138**(3), 375–396 (2008)
11. Ehrgott, M., Shao, L., Schöbel, A.: An approximation algorithm for convex multi-objective programming problems. J. Glob. Optim. **50**(3), 397–416 (2010)
12. Eichfelder, G., Kirst, P., Meng, L., Stein, O.: A general branch-and-bound framework for continuous global multiobjective optimization. J. Glob. Optim. **80**(1), 195–227 (2021)
13. Eichfelder, G., Niebling, J., Rocktäschel, S.: An algorithmic approach to multiobjective optimization with decision uncertainty. J. Glob. Optim. **77**(1), 3–25 (2019)
14. Evtushenko, Y., Posypkin, M.: A deterministic algorithm for global multi-objective optimization. Optim. Methods Softw. **29**(5), 1005–1019 (2013)
15. Fernández, J., Tóth, B.: Obtaining the efficient set of nonlinear biobjective optimization problems via interval branch-and-bound methods. Comput. Optim. Appl. **42**(3), 393–419 (2007)
16. Fliege, J., Vaz, A.I.F.: A method for constrained multiobjective optimization based on SQP techniques. SIAM J. Optim. **26**(4), 2091–2119 (2016)
17. Fonseca, C.M., Fleming, P.J.: An overview of evolutionary algorithms in multiobjective optimization. Evol. Comput. **3**(1), 1–16 (1995)
18. Göpfert, A., Riahi, H., Tammer, C., Zalinescu, C.: Variational Methods in Partially Ordered Spaces. Springer (2003)
19. Hamacher, H.W., Pedersen, C.R., Ruzika, S.: Finding representative systems for discrete bicriterion optimization problems. Oper. Res. Lett. **35**(3), 336–344 (2007)
20. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans. Evol. Comput. **10**(5), 477–506 (2006)
21. Ide, J., Köbis, E., Kuroiwa, D., Schöbel, A., Tammer, C.: The relationship between multi-objective robustness concepts and set-valued optimization. Fixed Point Theory Appl. **2014**(83) (2014). https://doi.org/10.1186/1687-1812-2014-83
22. Jamil, M., Yang, X.S.: A literature survey of benchmark functions for global optimization problems. Int. J. Math. Model. Numer. Optim. **4**(2), 150–194 (2013)
23. Jin, Y., Olhofer, M., Sendhoff, B.: Dynamic weighted aggregation for evolutionary multi-objective optimization: why does it work and how? In: Proceedings of the 3rd annual conference on genetic and evolutionary computation, GECCO'01, pp. 1042–1049. Morgan Kaufmann Publishers Inc. (2001)
24. Khan, A.A., Tammer, C., Zălinescu, C.: Set-valued Optimization. Springer (2015)
25. 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)
26. Klamroth, K., Tind, J., Wiecek, M.M.: Unbiased approximation in multicriteria optimization. Math. Methods Oper. Res. (ZOR) **56**(3), 413–437 (2003)
27. Kuhn, T., Ruzika, S.: A coverage-based box-algorithm to compute a representation for optimization problems with three objective functions. J. Glob. Optim. **67**(3), 581–600 (2016)
28. Löhne, A., Rudloff, B., Ulus, F.: Primal and dual approximation algorithms for convex vector optimization problems. J. Glob. Optim. **60**(4), 713–736 (2014)
29. MATLAB: Matlab bench documentation. https://www.mathworks.com/help/matlab/ref/bench.html. Accessed 25 June 2020
30. Miettinen, K.: Nonlinear Multiobjective Optimization. Springer (1998)
31. Niebling, J., Eichfelder, G.: A branch-and-bound-based algorithm for nonconvex multiobjective optimization. SIAM J. Optim. **29**(1), 794–821 (2019)
32. Pardalos, P.M., Žilinskas, A., Žilinskas, J.: Non-Convex Multi-Objective Optimization. Springer (2017)

33. Pascoletti, A., Serafini, P.: Scalarizing vector optimization problems. J. Optim. Theory Appl. **42**(4), 499–524 (1984)
34. Ruzika, S., Wiecek, M.M.: Approximation methods in multiobjective programming. J. Optim. Theory Appl. **126**(3), 473–501 (2005)
35. Scholz, D.: The multicriteria big cube small cube method. TOP **18**(1), 286–302 (2009)
36. Shao, L., Ehrgott, M.: An objective space cut and bound algorithm for convex multiplicative programmes. J. Global Optim. **58**(4), 711–728 (2013)
37. Tanaka, M., Watanabe, H., Furukawa, Y., Tanino, T.: GA-based decision support system for multicriteria optimization. In: 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century, vol. 2, pp. 1336–1561. IEEE (1995)
38. Teichert, K.: A hyperboxing pareto approximation method applied to radiofrequency ablationtreatment planning. Ph.D. Thesis, Technical University of Kaiserslautern (2014)
39. Thomann, J., Eichfelder, G.: Numerical results for the multiobjective trust region algorithm MHT. Data in Brief **25**, 104103 (2019)
40. Van Veldhuizen, D.A.: Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Ph.D. Thesis, Air Force Institute of Technology, USA (1999)
41. Wendell, R.E., Lee, D.N.: Efficiency in multiple objective optimization problems. Math. Program. **12**(1), 406–414 (1977)
42. Žilinskas, A.: A one-step worst-case optimal algorithm for bi-objective univariate optimization. Optim. Lett. **8**(7), 1945–1960 (2013)
43. Žilinskas, A., Žilinskas, J.: Adaptation of a one-step worst-case optimal univariate algorithm of bi-objective Lipschitz optimization to multidimensional problems. Commun. Nonlinear Sci. Numer. Simul. **21**(1–3), 89–98 (2015)
44. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms—a comparative case study. In: Parallel Problem Solving from Nature—PPSN V, Lecture Notes in Computer Science, vol. 1498, pp. 292–301. Springer (1998)