A hybrid patch decomposition approach to compute an enclosure for multi-objective mixed-integer convex optimization problems

Gabriele Eichfelder[*], Leo Warnow[*]

July 26, 2022

Abstract

In multi-objective mixed-integer convex optimization multiple convex objective functions need to be optimized simultaneously while some of the variables are only allowed to take integer values. In this paper we present a new algorithm to compute an enclosure of the nondominated set of such optimization problems. More precisely, we decompose the multi-objective mixed-integer convex optimization problem into several multi-objective continuous convex optimization problems, which we refer to as patches. We then dynamically compute and improve coverages of the nondominated sets of those patches to finally combine them to obtain an enclosure of the nondominated set of the multi-objective mixed-integer convex optimization problem. Additionally, we introduce a mechanism to reduce the number of patches that need to be considered in total. Our new algorithm is the first of its kind and guaranteed to return an enclosure of prescribed quality within a finite number of iterations. For selected numerical test instances we compare our new criterion space based approach to other algorithms from the literature and show that much larger instances can be solved with our new algorithm.

Key Words: multi-objective optimization, mixed-integer optimization, enclosure, approximation algorithm

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

1 Introduction

Multi-objective optimization deals with problems where not only one but several objective functions are minimized simultaneously. If those problems have some continuous and also some integer variables, we denote them multi-objective mixed-integer optimization problem. This kind of optimization problem arises for various practical applications, such as allocation, supply-chain, and financial problems (e.g. [26, 29, 32]).

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

We propose a new numerical solution method for such problems in this paper. More precisely, for once continuously differentiable convex objective functions $f_i \colon \mathbb{R}^{n+m} \to \mathbb{R}$, $i \in [p]$ and once continuously differentiable convex constraint functions $g_j \colon \mathbb{R}^{n+m} \to \mathbb{R}$, $j \in [q]$ we consider the optimization problem

$$\min_x \; f(x) \quad \text{s.t.} \quad g(x) \leq 0_q, \; x \in X := X_C \times X_I \tag{MOMICP}$$

where $[p] := \{1, \dots, p\}$, $f = (f_1, \dots, f_p) \colon \mathbb{R}^{n+m} \to \mathbb{R}^p$, $g = (g_1, \dots, g_q) \colon \mathbb{R}^{n+m} \to \mathbb{R}^q$ and $0_q := 0_{\mathbb{R}^q}$. We assume that $X_C := [l_C, u_C] \subseteq \mathbb{R}^n$ is a nonempty box with $l_C, u_C \in \mathbb{R}^n$ and $X_I := [l_I, u_I] \cap \mathbb{Z}^m$ is a (finite) nonempty subset of $\mathbb{Z}^m$ with $l_I, u_I \in \mathbb{Z}^m$.

Since the objectives of multi-objective optimization problems are usually conflicting, the aim is to compute so-called efficient solutions in the decision space, which correspond to nondominated points in the criterion space. Efficient solutions are defined by their property that it is not possible to find another feasible point that improves any objective without deteriorating another, see also [9]. In general, there is an infinite number of nondominated points for (MOMICP).

Many strategies to solve purely continuous multi-objective optimization problems have been studied in the last decades. In contrast to linear problems, for nonlinear (purely continuous) multi-objective optimization problems there exists no method to compute the nondominated set exactly. Instead, only approximations given by a finite representation or a coverage (e.g. using sandwiching techniques) of the nondominated set are computed, see also the survey [28].

For the mixed-integer setting several papers focus on linear optimization problems. A major difference when comparing the linear case with the general convex case is that, also in the mixed-integer setting, the nondominated set for multi-objective mixed-integer linear optimization problems can be computed exactly, while this is in general not possible for the convex case. For example, the nondominated set of bi-objective mixed-integer linear optimization problems basically consists of line segments and isolated points. Thus, by computing the isolated points and the end points defining the line segments, one obtains an exact and finite representation of the complete nondominated set (which in general still contains infinitely many points). One of the first approaches for this special setting of bi-objective mixed-integer linear optimization problems was the Triangle Splitting Method presented in [1]. Besides that, algorithms for the bi-objective linear setting have been presented for example in [30] and more recently in [23] with the Boxed Line Method.

Also for an arbitrary number of linear objectives, algorithms have been proposed in [33] as well as in [24]. However, those algorithms focus on finding only the so-called supported nondominated extreme points. An approach to find the exact nondominated set is the GoNDEF algorithm from [25]. For a survey of algorithms to solve multi-objective mixed-integer linear optimization problems, we refer to [18]. Once again, it is important to point out that all these algorithms rely heavily on the linear structure of the optimization problems. In particular, they make use of the fact that in the linear setting the nondominated set can be computed exactly since it can be completely represented by a finite number of points. This is not the case for general convex multi-objective mixed-integer optimization problems.

Within the last years, first algorithms to solve multi-objective mixed-integer nonlinear optimization problems have been published as well. One approach is to make use of scalarization-techniques, i.e., to solve the multi-objective optimization problem by solving several (parameterized) single-objective optimization problems. This is a very common approach from continuous multi-objective optimization and it is used in [4] to solve mixed-integer optimization problems as well. The downside of this approach is that a lot of single-objective mixed-integer nonlinear optimization problems have to be solved. Moreover, it is not clear how many and which of these problems one needs to solve in order to obtain a representation of the nondominated set of certain quality. Another downside of scalarization approaches is that the subproblems obtained by a scalarization of (MOMICP) are single-objective mixed-integer convex optimization problems, which still require a lot of computational effort to be solved fast and reliably.

Just recently, in [5] an approach to solve bi-objective mixed-integer convex optimization problems has been presented. That approach makes use of both the bi-objective and the mixed-integer structure of the problem. Also the quality of the computed approximation of the nondominated set can be controlled by that algorithm. What we have in common with the approach from [5] is that we also work with the same class of subproblems that are obtained from (MOMICP) by fixing the integer variables. However, the approach from [5] is only designed to solve bi-objective optimization problems and it cannot, at least not directly, be extended to solve general problems (MOMICP) with more than two objective functions. Moreover, in [5] the authors assume that the set of feasible integer assignments is known, which is not needed for our algorithm presented in this paper.

Another approach to approximate the nondominated set of bi-objective mixed-integer convex optimization problems is presented in [8]. The main idea of that approach is to make use of line segments in the criterion space and refine those in order to obtain an approximation of the nondominated set. Also this approach works only for bi-objective instances and so far it is not known whether and how it can be generalized for optimization problems (MOMICP) with three and more objective functions.

To the best of our knowledge, the only algorithms to solve arbitrary multi-objective mixed-integer convex optimization problems without using scalarization techniques and with a guaranteed quality of the computed approximation are the ones presented in [7] and [12]. Both of these algorithms are based on a branch-and-bound approach in the decision space. The downside of such branch-and-bound approaches is that they are typically not well-suited for optimization problems with a larger number of optimization variables. Another shortcoming of the method from [7] is that it can only use its full potential for multi-objective mixed-integer quadratic instances.

In this paper, we follow a completely different approach. We neither make use of a scalarization-first approach nor do we use a decision space based method like branch-and-bound. Instead, we introduce a new method that works almost entirely in the criterion space and computes an enclosure of the nondominated set of (MOMICP) of prescribed quality within a finite number of iterations. What is more, our algorithm works not only for bi-objective but for general multi-objective optimization problems (MOMICP) with an arbitrary number of objective functions.

A key ingredient of our new method is to make use of the finiteness of the set X_I of integer assignments. More precisely, we decompose the multi-objective mixed-integer convex optimization problem (MOMICP) into several multi-objective (continuous) convex optimization problems, which we refer to as patches, by fixing the integer assignments. This allows us to compute an enclosure for the nondominated set of (MOMICP) as a combination of coverages of the nondominated sets of these patches. Such a decomposition technique is also used in [5].

Since the nondominated sets of the patches can either contribute completely, partially, or not at all to the nondominated set of (MOMICP), we introduce a strategy to iteratively improve the coverages of the nondominated sets of those patches that contribute to the overall nondominated set and discard the other ones. We also present a technique to reduce the number of patches that need to be considered in total. In particular, this allows us to avoid full enumeration of all possible integer assignments. Our new algorithm keeps switching between two tasks: Computing integer assignments $x_I \in X_I$ to obtain new patches and improving the coverages of the nondominated sets of those patches that have already been found. Since we have that interplay of searching for new patches and dynamically improving patches from previous iterations, we call our algorithm a hybrid approach. To the best of our knowledge, this new strategy to dynamically improve the coverages of the patches and then combining them to obtain an enclosure of the nondominated set of (MOMICP) without having an explicit representation of the set X_I of integer assignments is the first of its kind.

The remaining paper is organized as follows. In Section 2 we introduce notations and definitions used throughout this paper. We briefly recall the concept of an enclosure for the nondominated set of (MOMICP) as well as a corresponding strategy to compute lower and upper bounds in Section 3. In Section 4 we give a formal definition of the patches and discuss their handling in our algorithm. Our strategy to compute integer assignments (to obtain new patches) and to reduce the number of patches that our algorithm needs to consider is presented in Section 5. In Section 6 we combine all these techniques and present our new algorithm to compute an enclosure of the nondominated set of (MOMICP) of prescribed quality. Finally, we present numerical results on selected test instances in Section 7 where we also compare our results to those from [7] and [12].

2 Notations and Definitions

In this paper, all relations, e.g., $x \leq x'$ for $x, x' \in \mathbb{R}^n$, are meant to be read componentwise. We denote by $e \in \mathbb{R}^p$ the all-ones vector in $\mathbb{R}^p$. Further, for $l, u \in \mathbb{R}^p$ we define the (closed) box $[l, u] := (\{l\} + \mathbb{R}^p_+) \cap (\{u\} - \mathbb{R}^p_+)$ and the open box $(l, u) := (\{l\} + \operatorname{int}(\mathbb{R}^p_+)) \cap (\{u\} - \operatorname{int}(\mathbb{R}^p_+))$.

We write $x = (x_C, x_I)$ for all $x \in X$ to distinguish between the continuous and integer variables of our optimization problem (MOMICP). The feasible set of (MOMICP) is denoted by S and is assumed to be nonempty. Its projection on $\mathbb{R}^m$ is defined by

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

We call $x_I \in S_I$ a feasible integer assignment. Since we will often fix the integer variables, we define for $\hat{x}_I \in \mathbb{Z}^m$

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

By our assumptions, all objective functions $f_i, i \in [p]$ are continuous and the feasible set S is compact. As a result, we have that

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

We assume that such a box B, which we also refer to as initial box B, is known.

The objective functions of (MOMICP) are usually competing with each other. For this reason, in general, it is not possible to find a feasible point that minimizes all objectives at the same time. Hence, we use the concept of efficiency.

Definition 2.1 *A point $\bar{x} \in S$ is called an efficient solution of* (MOMICP) *if there exists no $x \in S$ with*

$$\begin{aligned} &f_i(x) \leq f_i(\bar{x}) \text{ for all } i \in [p], \\ &f_j(x) < f_j(\bar{x}) \text{ for at least one } j \in [p]. \end{aligned}$$

It is called a weakly efficient solution of (MOMICP) *if there exists no $x \in S$ with*

$$f_i(x) < f_i(\bar{x}) \text{ for all } i \in [p].$$

We also need two concepts of dominance. Since we make use of lower and upper bound concepts, see Section 3, we need a dominance concept with respect to the relation $\leq$ and one with respect to $\geq$.

Definition 2.2 *Let $y^1, y^2 \in \mathbb{R}^p$ 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 $N \subseteq \mathbb{R}^p$ a vector $y \in \mathbb{R}^p$ is dominated given N with respect to $\preceq$ if

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

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 $\prec$.

Usually the specification of the relation $\preceq / \prec$ is known from the context. Thus, we do not explicitly mention it in most cases. As the images $f(\bar{x})$ of efficient solutions $\bar{x} \in S$ of (MOMICP) are nondominated given $f(S)$ w.r.t. $\leq$, they are called nondominated points of (MOMICP). 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 (MOMICP), i.e., $\mathcal{N} := \{f(x) \in \mathbb{R}^p \mid x \in \mathcal{E}\}$. We want to point out that even if the objective and constraint functions are all assumed to be continuous and convex, the nondominated set of (MOMICP) can be a disconnected set due to the integrality constraints. This is also illustrated in Figure 1 for a bi-objective example with $|S_I| = 2$, where the nondominated set is highlighted in orange.

3 Enclosure

In general, there is an infinite number of nondominated points for (MOMICP). In contrast to the linear case, there exists no finite and at the same time exact representation of the complete nondominated set in the general convex setting. Hence, the aim of this paper is to compute an approximation of the nondominated set. More precisely, we will make use of a concept that allows to compute an approximation of the nondominated set of the overall problem (MOMICP) as a combination of approximations of the nondominated sets of patches. Thereby, a patch corresponds to (MOMICP) with a fixed integer assignment $\hat{x}_I \in S_I$. We provide a formal definition of patches in the next section.

In general, there are two classes of approximation concepts in multi-objective optimization. The first is what we refer to as representation approaches. There, the goal is to compute a finite number of nondominated points to represent the overall nondominated set. The distance of these points is then often used as a quality criterion for the representation. However, due to gaps and potentially even isolated points in the nondominated set of (MOMICP), this can be hard to apply in the setting of multi-objective mixed-integer optimization.

We focus on the second class, where the goal is to compute a superset of the nondominated set, referred to as coverage approaches. A suitable concept for such a coverage is the enclosure as presented in [11]. In particular, that concept respects the natural ordering since it is a box-based approach. This enables us to compute the overall coverage of the nondominated set $\mathcal{N}$ of (MOMICP) as a combination of the coverages of the nondominated sets of the patches.

Definition 3.1 *Let $L, U \subseteq \mathbb{R}^p$ be two finite sets with*

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

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

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

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

Besides the enclosure itself, we also need a corresponding quality criterion which serves us as a termination criterion later in the algorithm. Therefore, we use a quality concept from [11] called the width of $\mathcal{A}$. It is denoted by $w(\mathcal{A})$ and equals the optimal value of

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

where $s(l, u) := \min \{u_i - l_i \mid i \in [p]\}$ denotes the shortest edge length of a box $[l, u]$. While in single-objective global optimization one is typically interested in the largest edge length of boxes in the decision space, it might seem surprising that here the shortest edge length (in the criterion space) is used. However, it turns out that working with exactly this quality measure is a natural extension from the width of the interval containing the optimal value in the single-objective case and yields to some desirable properties. It is shown in [11, Lemma 3.1] that for an enclosure $\mathcal{A}$ with $w(\mathcal{A}) < \varepsilon$ for some $\varepsilon > 0$ we have that all $y \in \mathcal{A} \cap f(S)$ are ε-nondominated points. For more details on the motivation and a discussion of this quality measure, we refer to [11, 13].

Before we present a sketch of our algorithm to compute an enclosure, we first present a method to compute the lower and upper bound sets from Definition 3.1. More precisely, we use the concept of so-called Local Upper Bounds (LUBs) from [21], which is related to the bound concept that appeared earlier in [10]. The authors from [21] call a set $Y \subseteq \mathbb{R}^p$ stable if the elements of Y are not pairwise comparable, i.e., for all $y^1, y^2 \in Y$ with $y^1 \neq y^2$ there exists $i, j \in [p]$ such that $y^1_i < y^2_i$ and $y^1_j > y^2_j$. In [21] only the concept to compute an upper bound set is presented. We use the idea from [21] to compute also a lower bound set. For this reason, we slightly extended the notation to be able to distinguish between these two cases.

Definition 3.2 *Let $N \subseteq f(S)$ be a finite and stable set. Then the lower search region for N is $s(N) := \{y \in \operatorname{int}(B) \mid y' \nleq y \text{ for every } y' \in N\}$ and the lower search zone for some $u \in \mathbb{R}^p$ is $c(u) := \{y \in \operatorname{int}(B) \mid y < u\}$. 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) $c(u^1) \nsubseteq c(u^2)$ *for all* $u^1, u^2 \in U(N)$, $u^1 \neq u^2$.

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

As already mentioned, the same concept can be used to obtain so-called local lower bounds as follows.

Definition 3.3 *Let $N \subseteq \operatorname{int}(B)$ be a finite and stable set. Then the upper search region for N is $S(N) := \{y \in \operatorname{int}(B) \mid y' \ngeq y \text{ for every } y' \in N\}$ and the upper search zone for some $l \in \mathbb{R}^p$ is $C(l) := \{y \in \operatorname{int}(B) \mid y > l\}$. A set $L = L(N)$ is called local lower bound set given N if*

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

(ii) $C(l^1) \nsubseteq C(l^2)$ *for all* $l^1, l^2 \in L(N)$, $l^1 \neq l^2$.

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

It is known that the local upper bound set and the local lower bound set from Definition 3.2 and Definition 3.3 are uniquely determined and finite, see [11]. We provide an illustration of both concepts in Figure 1.

The following result provides a relation between local lower/upper bounds and lower/upper bounds as used in Definition 3.1.

Lemma 3.4 *Let $N^1 \subseteq f(S)$ and $N^2 \subseteq \operatorname{int}(B) \setminus (f(S) + \operatorname{int}(\mathbb{R}^p_+))$ be 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.*

Proof. Let $N^1 \subseteq f(S)$ be a finite and stable set. Then it was already shown in [13, Lemma 3.3] that $U(N^1)$ is an upper bound set in the sense of Definition 3.1.

Let $N^2 \subseteq \operatorname{int}(B) \setminus (f(S) + \operatorname{int}(\mathbb{R}^p_+))$ be a finite and stable set. First, we show that it holds $\mathcal{N} \subseteq \operatorname{cl}(S(N^2))$. Let $\bar{y} \in \mathcal{N} \subseteq f(S) \subseteq \operatorname{int}(B)$ be a nondominated point. Assume that $\bar{y} \notin S(N^2)$. Then, by Definition 3.3, there exists $y' \in N^2$ with $y' \geq \bar{y}$. Since $y' \in N^2$ and $\bar{y} \in f(S)$ it also holds that $y' \not> \bar{y}$. As a result, we have that there exists an index $i \in [p]$ such that $y'_i = \bar{y}_i$. Since $y' \in N^2 \subseteq \operatorname{int}(B)$ there exists $\varepsilon > 0$ such that $y' + \varepsilon e \in \operatorname{int}(B)$. Also, B is a box and hence $\operatorname{int}(B)$ is a convex

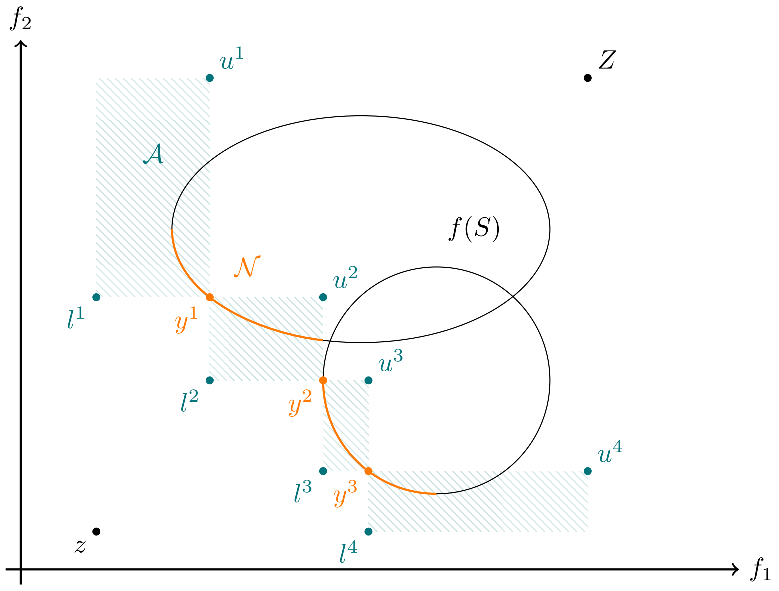


Figure 1: Approximation $\mathcal{A}(L,U)$ of $\mathcal{N}$ with $N = \{y^1, y^2, y^3\}$, local lower bound set $L = L(N) = \{l^1, l^2, l^3, l^4\}$, and local upper bound set $U = U(N) = \{u^1, u^2, u^3, u^4\}$

set. Thus, for all $\lambda \in [0,1]$ it holds that $\bar{y} + \lambda((y' + \varepsilon e) - \bar{y}) \in \text{int}(B)$. This implies that for all $k \in \mathbb{N}$ we have that $y^k := \bar{y} + \frac{1}{k}((y' + \varepsilon e) - \bar{y}) \in \text{int}(B)$. Moreover, since $y' \geq \bar{y}$ and $\varepsilon e > 0_p$ we obtain that $y^k > \bar{y}$. In particular, for all $k \in \mathbb{N}$ it holds that $y^k_i = \bar{y}_i + \frac{1}{k}\varepsilon = y'_i + \frac{1}{k}\varepsilon$ and hence $y' \not\geq y^k$. Further, there exists no $y'' \in N^2$ with $y'' \geq y^k > \bar{y} \in f(S)$ since this contradicts the choice of N^2. As a result, $y^k \in S(N^2)$ for all $k \in \mathbb{N}$ and $\bar{y} = \lim_{k\to\infty} y^k \in \text{cl}(S(N^2))$. Finally, by Definition 3.3 and since $L(N^2)$ is finite, we obtain that $\mathcal{N} \subseteq \text{cl}(S(N^2)) = \text{cl}(\bigcup_{l \in L(N^2)} C(l)) = \bigcup_{l \in L(N^2)} \text{cl}(C(l)) \subseteq \bigcup_{l\in L(N^2)}\{l\} + \mathbb{R}^p_+ = L(N^2) + \mathbb{R}^p_+$. □

It is also important to mention that for Definition 3.2 and Definition 3.3 one does not necessarily need to assume N to be stable, see [21, Remark 2.2].

Remark 3.5 *Let $N^1 \subseteq f(S)$ be an arbitrary set and denote by $\hat{N}^1 := \{y \in N^1 \mid$ y is nondominated given N^1 w.r.t. $\leq\}$. Then it holds that $s(N^1) = s(\hat{N}^1)$, which also implies $U(N^1) = U(\hat{N}^1)$. Analogously, let $N^2 \subseteq \text{int}(B) \setminus (f(S) + \text{int}(\mathbb{R}^p_+))$ be an arbitrary set and denote by $\hat{N}^2 := \{y \in N^2 \mid y$ is nondominated given N^2 w.r.t. $\geq\}$. Then it holds that $S(N^2) = S(\hat{N}^2)$, which also implies $L(N^2) = L(\hat{N}^2)$.*

In our new algorithm, the sets N^1 and N^2 from Remark 3.5 and hence the local lower and local upper bound sets are built up iteratively. It is known from [21] that in such a setting it is not necessary to recompute the whole local upper or local lower bound set each time those sets are updated. Instead, whenever a new point $y \in \mathbb{R}^p$ is added to the set N from Definition 3.2 or Definition 3.3 only certain local upper or local lower bounds need to be updated. We refer to these points $y \in \mathbb{R}^p$ as update points. To update a local upper bound set, we use [21, Algorithm 3], see Algorithm 1. While in [21] only the concept of local upper bounds is considered, it is an easy task to apply the same technique to update the set of local lower bounds, see Algorithm 2. Within both algorithms we use the following notation from [21].

For $y \in \mathbb{R}^p, \alpha \in \mathbb{R}$ and an index $i \in [p]$ we define

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

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 $U(N \cup \{y\})$
 1: procedure UpdateLUB($U(N), y$)
 2:     $A = \{u \in U(N) \mid y < u\}$
 3:     for $i \in [p]$ do
 4:         $B_i = \{u \in U(N) \mid y_i = u_i \text{ and } y_{-i} < u_{-i}\}$
 5:         $P_i = \emptyset$
 6:     end for
 7:     for $i \in [p]$ do
 8:         for $u \in A$ do
 9:             $P_i = P_i \cup \{(y_i, u_{-i})\}$
10:         end for
11:     end for
12:     for $i \in [p]$ do
13:         $P_i = \{u \in P_i \mid u \not\leq u' \text{ for all } u' \in P_i \cup B_i,\ u' \neq u\}$
14:     end for
15:     $U(N \cup \{y\}) = (U(N) \setminus A) \cup \bigcup_{i \in [p]} P_i$
16: end procedure
```

Next, we briefly present a sketch of our algorithm to compute an enclosure of the nondominated set $\mathcal{N}$ of (MOMICP).

An upper bound set will be computed based on all images of feasible points that are generated within our algorithm. These will be weakly nondominated points of the patches. Since we expect the upper bounds to improve and get smaller, this also allows us to quickly discard certain patches (and hence certain integer assignments) that do not contribute to the nondominated set of (MOMICP). In particular, we will be able to detect if a patch is in some sense dominated by the upper bound set.

For the lower bound set, two strategies are applied simultaneously. The first strategy is to compute an individual lower bound set for every patch. We start with a single point that dominates all image points of the patch and then improve this lower bound set iteratively. The second strategy is to simultaneously compute a global lower bound set for the nondominated set of the original problem (MOMICP). We add here the word global to emphasize that this lower bound set corresponds to the overall problem (MOMICP) and to clearly distinguish between this second strategy to compute a lower bound set and the first strategy on the patch-level. This global lower bound set, in general, converges towards the upper bound set before all integer assignments have been computed. This allows us to avoid that we need to do some computations for all feasible integer assignments $x_I \in S_I$ which can be very time consuming. In particular, without this second strategy we would need to compute at least one lower bound for all patches in order to compute the overall enclosure. This is also one of the key differences between our approach and the approach from [5] for the bi-objective case. There the

Algorithm 2 Updating a local lower bound set

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

```
1: procedure UpdateLLB($L(N), y$)
2:     $A = \{l \in L(N) \mid y > l\}$
3:     for $i \in [p]$ do
4:         $B_i = \{l \in L(N) \mid y_i = l_i$ and $y_{-i} > l_{-i}\}$
5:         $P_i = \emptyset$
6:     end for
7:     for $i \in [p]$ do
8:         for $l \in A$ do
9:             $P_i = P_i \cup \{(y_i, l_{-i})\}$
10:        end for
11:    end for
12:    for $i \in [p]$ do
13:        $P_i = \{l \in P_i \mid l \not\geq l'$ for all $l' \in P_i \cup B_i,\ l' \neq l\}$
14:    end for
15:    $L(N \cup \{y\}) = (L(N) \setminus A) \cup \bigcup_{i \in [p]} P_i$
16: end procedure
```

authors assume that the set S_I of feasible integer assignments is known and they have to perform at least some computations for each $x_I \in S_I$, see [5, Algorithm 1]. The interplay of both strategies, i.e., computing a lower bound set for the nondominated set of the overall problem and improving the coverages of the patches, is also the reason why we call our algorithm a hybrid approach.

In the next section, we give a formal definition of the patches and describe their handling in more detail. This also includes the computation of the upper bound set and the individual lower bound sets.

Since a patch always belongs to a feasible integer assignment, we need to compute such an assignment as a prerequisite to handle the patches. We present a technique to do that in Section 5. We will also show that this step can be combined with the computation of a global lower bound set.

Finally, in section Section 6 we combine all these mechanisms to obtain our new algorithm to compute an enclosure of the nondominated set $\mathcal{N}$ of (MOMICP).

4 Handling of the patches

As already mentioned, we obtain a patch of (MOMICP) by fixing the integer assignment $x_i \in X_I$. Thus, for $\hat{x}_I \in S_I$ we define the corresponding patch (problem) as

$$\min_{x_C} f(x_C, \hat{x}_I) \quad \text{s.t.} \quad g(x_C, \hat{x}_I) \leq 0_q,\ x_C \in X_C. \tag{P($\hat{x}_I$)}$$

For a patch, we always need a feasible integer assignment $\hat{x}_I \in S_I$. In general the set of feasible integer assignments S_I is not known and is rather hard to compute. In this section, we focus entirely on the handling of the patches, but we present a technique to compute feasible integer assignments in the next section.

Since it holds for all $\hat{x}_I \in S_I$ and all feasible points x_C of (P($\hat{x}_I$)) that $f(x_C, \hat{x}_I) \in f(S)$, image points of patches can be used to generate an upper bound set for (MOMICP), see also Lemma 3.4, Remark 3.5, and in particular the forthcoming Lemma 6.6 in which this statement is finally proved for the overall algorithm.

We denote by $\mathcal{N}_{\hat{x}_I}$ the nondominated set of (P($\hat{x}_I$)). Then for the lower bound set $L_{\hat{x}_I}$ of (P($\hat{x}_I$)) we need that $\mathcal{N}_{\hat{x}_I} \subseteq L_{\hat{x}_I} + \mathbb{R}^p_+$. Such a lower bound set can be computed using the concept of local lower bounds.

Thus, we need a mechanism to compute update points for the upper bound set U and the lower bound set $L_{\hat{x}_I}$. For this purpose, we use the following approach. The overall enclosure $\mathcal{A}$ of the nondominated set $\mathcal{N}$ of (MOMICP) can be interpreted as a combination of box coverages of the nondominated sets of certain patches (P($\hat{x}_I$)), i.e., coverages of the form $(L_{\hat{x}_I} + \mathbb{R}^p_+) \cap (U - \mathbb{R}^p_+)$. For this reason, we aim for such update points that quickly improve those coverages. We consider a patch (P($\hat{x}_I$)) for fixed $\hat{x}_I \in S_I$. When searching for an update point, we loop through the lower bound set $L_{\hat{x}_I}$. Since it holds $\mathcal{N}_{\hat{x}_I} \subseteq L_{\hat{x}_I} + \mathbb{R}^p_+$, this also implies that the search covers the whole nondominated set of the patch. For each of the lower bounds $l \in L_{\hat{x}_I}$ we identify the upper bound $u \in U$ with maximal shortest edge length $s(l, u)$. We choose this selection criterion since the width of the overall enclosure $\mathcal{A}$ of $\mathcal{N}$ is the supremum of such shortest edge lengths and hence it is a reasonable approach to update those boxes with the largest shortest edge length first. More precisely, given $\hat{x}_I \in S_I$ and $l, u \in \mathbb{R}^p$ with $l < u$, the search for update points is performed by solving

$$\begin{aligned} \min_{x_C, t} \ t \quad \text{s.t.} \quad & f(x_C, \hat{x}_I) - l - t(u - l) \leq 0_p, \\ & g(x_C, \hat{x}_I) \leq 0_q, \\ & x_C \in X_C, \ t \in \mathbb{R}. \end{aligned} \tag{SUP($\hat{x}_I, l, u$)}$$

By [17, Proposition 2.3.4 and Theorem 2.3.1] we know that for all $\hat{x}_I \in S_I$ and all $l, u \in \mathbb{R}^p$ with $l < u$ there exists an optimal solution $(\bar{x}_C, \bar{t})$ of (SUP($\hat{x}_I, l, u$)). In [22, Theorem 3.2] it is shown that for every optimal solution $(\bar{x}_C, \bar{t})$ of (SUP($\hat{x}_I, l, u$)) the point $f(\bar{x}_C, \hat{x}_I) \in f(S)$ is a weakly nondominated point of (P($\hat{x}_I$)). In particular, for every optimal solution $(\bar{x}_C, \bar{t})$ of (SUP($\hat{x}_I, l, u$)) it holds that $f(\bar{x}_C, \hat{x}_I) \in f(S)$ and $l + \bar{t}(u - l) \notin f(S_{\hat{x}_I}) + \text{int}(\mathbb{R}^p_+)$. Hence, those points can be used to update U and $L_{\hat{x}_I}$.

To keep track of the lower bounds for the different patches, we introduce a new data structure. We denote this structure by $\mathcal{D}$ and refer to it as the integer data structure since it consists of data related to certain integer assignments. For each integer assignment $\hat{x}_I \in S_I$ this data structure has an entry $\mathcal{D}(\hat{x}_I)$ that consists of four substructures.

The first substructure is the set of local lower bounds of (P($\hat{x}_I$)), denoted by $\mathcal{D}(\hat{x}_I).L$. The second one is a boolean value $\mathcal{D}(\hat{x}_I).S$ that indicates whether the coverage of the nondominated set $\mathcal{N}_{\hat{x}_I}$ needs further improvement. For example, this value is set to false in case it is recognized that $\hat{\mathcal{N}}$ does not contribute to the nondominated set $\mathcal{N}$ of (MOMICP). We call the integer assignment $\hat{x}_I$ active if $\mathcal{D}(\hat{x}_I).S$ is set to true and inactive otherwise. All weakly efficient points $x \in S_{\hat{x}_I}$ of the subproblem (P($\hat{x}_I$)) computed when solving (SUP($\hat{x}_I, l, u$)) are saved in the set $\mathcal{D}(\hat{x}_I).E$. Analogously, the fourth and last substructure $\mathcal{D}(\hat{x}_I).N$ contains the weakly nondominated points $y \in \mathbb{R}^p$ of the subproblem (P($\hat{x}_I$)) that are generated within the algorithm. We denote by $\mathcal{D}.L, \mathcal{D}.E$ and $\mathcal{D}.N$ the union of the sets for all integer assignments, e.g., $\mathcal{D}.N := \{y \in \mathbb{R}^p \mid y \in \mathcal{D}(\hat{x}_I).N \text{ for some } \hat{x}_I \in S_I\}$.

In the following, we present the algorithms that initialize and update the integer data structure $\mathcal{D}$. Any entry $\mathcal{D}(\hat{x}_I)$ of the integer data structure is initialized by Algorithm 3. The lower bound $z \in \mathbb{R}^p$ in line 2 of Algorithm 3 can be computed using the ideal point $z' \in \mathbb{R}^p$ of (P($\hat{x}_I$)) and a small offset $\sigma > 0$, i.e., $z_i = z'_i - \sigma$ for all $i \in [p]$. Depending on how exactly the lower bound $z \in \mathbb{R}^p$ is computed, it is also possible that some first weakly efficient points or weakly nondominated points of (P($\hat{x}_I$)) are computed. In that case $\mathcal{D}(\hat{x}_I).E$ and $\mathcal{D}(\hat{x}_I).N$ are not necessarily initialized as empty sets in Algorithm 3.

Algorithm 3 Initialization of $\mathcal{D}(\hat{x}_I)$ for a new integer assignment $\hat{x}_I \in S_I$

Input: New integer assignment $\hat{x}_I \in S_I$
Output: Initialized entry $\mathcal{D}(\hat{x}_I)$ of the integer data structure

```
1: procedure InitIDS($\hat{x}_I$)
2:     Compute lower bound $z \in \mathbb{R}^p$ with $f(S_{\hat{x}_I}) \subseteq \{z\} + \mathrm{int}(\mathbb{R}^p_+)$
3:     Initialize $\mathcal{D}(\hat{x}_I).L = \{z\}$, $\mathcal{D}(\hat{x}_I).S = \text{true}$, $\mathcal{D}(\hat{x}_I).E = \emptyset$, $\mathcal{D}(\hat{x}_I).N = \emptyset$
4: end procedure
```

If the same integer assignment $\hat{x}_I$ is visited again, then $\mathcal{D}(\hat{x}_I)$ is updated. This procedure computes update points for U and $L_{\hat{x}_I}$ as described above and is presented in Algorithm 4.

As explained in the previous section, our new algorithm applies two strategies to compute lower bounds simultaneously. The first strategy is to compute the lower bound set L for the enclosure $\mathcal{A}$ of $\mathcal{N}$ as a combination of the lower bound sets $L_{\hat{x}_I}$ for different integer assignments $\hat{x}_I \in S_I$. The proof of finiteness of our algorithm is largely based on this strategy. More precisely, by our assumptions there are only finitely many integer assignments $\hat{x}_I \in S_I$. If for each of these feasible integer assignments $\hat{x}_I \in S_I$ the handling of the patch (P($\hat{x}_I$)) is finite, i.e., $\mathcal{D}(\hat{x}_I).S = \text{false}$ after a finite number of calls of Algorithm 4, then the overall algorithm is finite as well, see Theorem 6.5.

The next theorem guarantees an improvement of the coverage of a patch (P($\hat{x}_I$)) with each call of Algorithm 4. Within the theorem we make use of the following notation. For $\hat{x}_I \in S_I$, the corresponding lower bound set $L_{\hat{x}_I} \neq \emptyset$ and the upper bound set $U \neq \emptyset$ we denote the corresponding coverage of $\mathcal{N}_{\hat{x}_I}$ by

$$C = \mathcal{C}(L_{\hat{x}_I}, U) := (L_{\hat{x}_I} + \mathbb{R}^p_+) \cap (U - \mathbb{R}^p_+).$$

Since $\mathcal{C}$ is just a combination of boxes, its volume $\mathrm{vol}(\mathcal{C})$ can be easily evaluated.

Theorem 4.1 *Let $\hat{x}_I \in S_I, \varepsilon > 0, U$, and $\mathcal{D}$ be the input parameters for Algorithm 4. Assume that $\mathcal{D}(\hat{x}_I)$ is already initialized. Denote by $L^{start}_{\hat{x}_I} := \mathcal{D}(\hat{x}_I).L$ and U^{start} the lower and upper bound sets at the beginning of the current call of Algorithm 4 and by $L^{end}_{\hat{x}_I}, U^{end}$ the sets afterwards. Additionally, assume that there exist $l \in L^{start}_{\hat{x}_I}$ and $u \in U^{start}$ with $u \in (\{l+\varepsilon e\}+\mathrm{int}(\mathbb{R}^p_+))$. Then, the volume of the corresponding coverage $\mathcal{C}$ is reduced by at least $(\frac{\varepsilon}{2})^p$, i.e., $\mathrm{vol}(\mathcal{C}(L^{end}_{\hat{x}_I}, U^{end})) \leq \mathrm{vol}(\mathcal{C}(L^{start}_{\hat{x}_I}, U^{start})) - (\frac{\varepsilon}{2})^p$.*

Proof. In the following, we use the notation from Algorithm 4. By our assumptions there exist $l \in L^{\text{start}}_{\hat{x}_I}, u \in U^{\text{start}}$ with $u \in (\{l + \varepsilon e\} + \mathrm{int}(\mathbb{R}^p_+))$, i.e., $u_i - l_i > \varepsilon$ for all $i \in [p]$. We can assume w.l.o.g. that these bounds correspond exactly to the assignment of l and u when line 7 of Algorithm 4 is reached for the first time.

Algorithm 4 Updating $\mathcal{D}(\hat{x}_I)$ for an integer assignment $\hat{x}_I \in S_I$

Input: Integer assignment $\hat{x}_I \in S_I$, quality $\varepsilon > 0$, upper bound set U, and data structure $\mathcal{D}$

Output: Updated set U and updated integer data structure $\mathcal{D}$

```
 1: procedure UpdateIDS($\hat{x}_I, \varepsilon, U, \mathcal{D}$)
 2:     Initialize $L_{\hat{x}_I} = \mathcal{D}(\hat{x}_I).L$, done = true
 3:     Choose a small offset $\sigma \in (0, \varepsilon/2)$
 4:     for $l \in L_{\hat{x}_I}$ do
 5:         if $(\{l + \varepsilon e\} + \mathrm{int}(\mathbb{R}^p_+)) \cap U \neq \emptyset$ then
 6:             done = false
 7:             Select $u \in (\{l + \varepsilon e\} + \mathrm{int}(\mathbb{R}^p_+)) \cap U$ with maximal $s(l, u)$
 8:             Solve (SUP($\hat{x}_I, l, u$)) with optimal solution $(\bar{x}_C, \bar{t})$ and set
                $\bar{x} := (\bar{x}_C, \hat{x}_I)$, $\bar{y} := f(\bar{x})$ and $\tilde{y} := l + \bar{t}(u - l)$
 9:             if $\tilde{y} \not< Z$ then
10:                 Set $\bar{t} := \min\{(Z_i - l_i)/(u_i - l_i) \mid i \in [p]\}$ and
                      $\tilde{y} := l + \bar{t}(u - l) - \sigma e$
11:             end if
12:             $\mathcal{D}(\hat{x}_I).E = \mathcal{D}(\hat{x}_I).E \cup \{\bar{x}\}$
13:             $\mathcal{D}(\hat{x}_I).N = \mathcal{D}(\hat{x}_I).N \cup \{\bar{y}\}$
14:             $\mathcal{D}(\hat{x}_I).L = $ UpdateLLB($\mathcal{D}(\hat{x}_I).L, \tilde{y}$)
15:             $U = $ UpdateLUB($U, \bar{y}$)
16:         end if
17:     end for
18:     if done == true then
19:         Set integer assignment inactive: $\mathcal{D}(\hat{x}_I).S = $ false
20:     end if
21: end procedure
```

First, we consider the case that $\bar{t} \geq 1$ which implies that the lower bound set $L_{\hat{x}_I}$ is updated by Algorithm 2 using $\tilde{y} \in \mathrm{int}(B)$ as update point. As a result, the open box $(l, \tilde{y})$ is removed from $\mathcal{C}$. We have that $\tilde{y} \geq l + \bar{t}(u - l) - \sigma e \geq u - \sigma e, 0 < \sigma < 0.5\,\varepsilon$, and $u_i - l_i > \varepsilon$ for all $i \in [p]$. This implies that

$$\mathrm{vol}((l, \tilde{y})) = \prod_{i=1}^{p} (\tilde{y}_i - l_i) \geq \prod_{i=1}^{p} ((u_i - \sigma) - l_i) \geq \prod_{i=1}^{p} 0.5\,\varepsilon = \left(\frac{\varepsilon}{2}\right)^p.$$

Consequently, we obtain that $\mathrm{vol}(\mathcal{C}(L^{\mathrm{end}}_{\hat{x}_I}, U^{\mathrm{end}})) \leq \mathrm{vol}(\mathcal{C}(L^{\mathrm{start}}_{\hat{x}_I}, U^{\mathrm{start}})) - (\frac{\varepsilon}{2})^p$.

Next, we consider the case $\bar{t} \in (0.5, 1)$. In this case we update the lower bound set $\mathcal{D}(\hat{x}_I).L$ using Algorithm 2 with update point $\tilde{y} := l + \bar{t}(u - l) \geq 0.5(u + l)$. By Definition 3.3 this implies that the open box $(l, \tilde{y})$ is removed from the upper search region and in particular from $\mathcal{C}$. Since we have that

$$\mathrm{vol}((l, \tilde{y})) = \prod_{i=1}^{p} (\tilde{y}_i - l_i) \geq \prod_{i=1}^{p} (0.5(u_i + l_i) - l_i) \geq \prod_{i=1}^{p} 0.5\,\varepsilon = \left(\frac{\varepsilon}{2}\right)^p,$$

we obtain that $\mathrm{vol}(\mathcal{C}(L^{\mathrm{end}}_{\hat{x}_I}, U^{\mathrm{end}})) \leq \mathrm{vol}(\mathcal{C}(L^{\mathrm{start}}_{\hat{x}_I}, U^{\mathrm{start}})) - (\frac{\varepsilon}{2})^p$.

The final case to consider is $\bar{t} \in [0, 0.5]$. In that case, the upper bound set $U = U^{\text{start}}$ is updated using Algorithm 1 with update point $f(\bar{x}) \leq l + \bar{t}(u - l) \leq 0.5(u + l)$. By Definition 3.2 this implies that the open box $(f(\bar{x}), u)$ is removed from $\mathcal{C}$ and with

$$\operatorname{vol}((f(\bar{x}), u)) = \prod_{i=1}^{p}(u_i - f_i(\bar{x})) \geq \prod_{i=1}^{p}(u_i - 0.5(u_i + l_i) \geq \prod_{i=1}^{p} 0.5\,\varepsilon = \left(\frac{\varepsilon}{2}\right)^p,$$

this implies that $\operatorname{vol}(\mathcal{C}(L^{\text{end}}_{\hat{x}_I}, U^{\text{end}})) \leq \operatorname{vol}(\mathcal{C}(L^{\text{start}}_{\hat{x}_I}, U^{\text{start}})) - (\frac{\varepsilon}{2})^p$. □

5 Computing integer assignments and global lower bounds

In this section, we present a method to compute the feasible integer assignments $\hat{x}_I \in S_I$ which are needed for the handling of the patches ($\text{P}(\hat{x}_I)$).

But first, we introduce a method to compute a global lower bound set for the nondominated set $\mathcal{N}$ of (MOMICP). We start by motivating why such a method to compute lower bounds is needed. For this reason, assume that there was an efficient way to compute all feasible integer assignments, i.e., the set S_I. Then we would be able to handle all the corresponding patches and in particular the integer data structure $\mathcal{D}$. However, there can be a large number of feasible integer assignments of which only a few contribute to the nondominated set. Still, to compute a global lower bound set L out of the lower bounds $L_{\hat{x}_I}, \hat{x}_I \in S_I$, we would need to at least initialize $\mathcal{D}(\hat{x}_I)$ for each $\hat{x}_I \in S_I$ to ensure that we actually obtain a valid lower bound set. For example, let there be three feasible integer assignments $x^1, x^2, x^3 \in S_I$. If $\mathcal{D}(x^3)$ is not initialized then we do not know whether $L = L_{x^1} \cup L_{x^2}$ is a valid lower bound set and hence, if $\mathcal{A}(L, U)$ is a valid enclosure for the nondominated set $\mathcal{N}$ of (MOMICP), see also Figure 2.

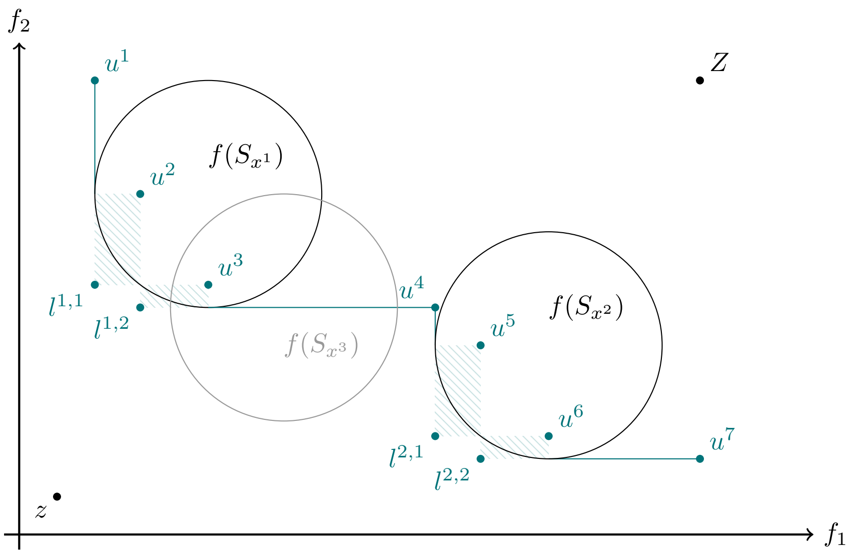


Figure 2: Preliminary enclosure based on the results for the patches $\text{P}(x^1)$ and $\text{P}(x^2)$ and $\mathcal{D}(x^3)$ not initialized

To overcome the problem of needing to initialize $\mathcal{D}(\hat{x}_I)$ for all $\hat{x}_I \in S_I$, we introduce a strategy to compute a global lower bound set. More precisely, we compute a lower bound set for a relaxation of (MOMICP) and then iteratively improve both the relaxation and the corresponding lower bound set. For the relaxation, we linearize the objective and constraint functions of (MOMICP) to obtain a multi-objective mixed-integer linear optimization problem. Thereby, we use that for any convex continuously differentiable function $h\colon \mathbb{R}^n \to \mathbb{R}$ and every $\hat{x} \in \mathbb{R}^n$ it holds

$$h(x) \geq h(\hat{x}) + \nabla h(\hat{x})^\top (x - \hat{x}) \text{ for all } x \in \mathbb{R}^n. \tag{5.1}$$

The right hand side of this inequality is an affine-linear function that approximates h in the point $\hat{x}$ and is called linearization of h with linearization point $\hat{x}$. Also, for $x = \hat{x}$ we have equality in (5.1), i.e., an exact approximation. Thus, let $\emptyset \neq \mathcal{X} \subseteq \mathbb{R}^{n+m}$ be a (not necessarily finite) set of linearization points. Then we obtain a relaxed version of (MOMICP) by

$$\begin{aligned} \min_{x,\eta}\ \eta \quad \text{s.t.} \quad & f_i(\hat{x}) + \nabla f_i(\hat{x})^\top (x - \hat{x}) \leq \eta_i \quad \forall i \in [p],\ \forall \hat{x} \in \mathcal{X}, \\ & g_j(\hat{x}) + \nabla g_j(\hat{x})^\top (x - \hat{x}) \leq 0 \quad \forall j \in [q],\ \forall \hat{x} \in \mathcal{X}, \\ & x \in X,\ \eta \in \mathbb{R}^p. \end{aligned} \tag{R($\mathcal{X}$)}$$

Since we linearized both the objective and the constraint functions, the objective functions have been lifted to the constraints by introducing a new variable $\eta \in \mathbb{R}^p$. This kind of relaxation is well known from single-objective mixed-integer convex optimization and can be found for example in the context of outer approximation, see [3, 15]. To compute a global lower bound set, we use weakly efficient points of (R($\mathcal{X}$)) as update points and make use of the following lemma.

Lemma 5.1 *Let $\mathcal{X}$ be a set of linearization points with $\emptyset \neq \mathcal{X} \subseteq \mathbb{R}^{n+m}$ and denote by $(\bar{x}, \bar{\eta})$ a weakly efficient solution of (R($\mathcal{X}$)). Then $\bar{\eta} \notin f(S) + \operatorname{int}(\mathbb{R}^p_+)$.*

Proof. Assume that $\bar{\eta} \in f(S) + \operatorname{int}(\mathbb{R}^p_+)$. Then there exist $x' \in S$ and $k \in \operatorname{int}(\mathbb{R}^p_+)$ with $\bar{\eta} = f(x') + k$. Hence, for all $i \in [p], j \in [q]$ and for all $\hat{x} \in \mathcal{X}$ it holds

$$\begin{aligned} \bar{\eta}_i - k_i = f_i(x') &\geq f_i(\hat{x}) + \nabla f_i(\hat{x})^\top (x' - \hat{x}), \\ 0 \geq g_j(x') &\geq g_j(\hat{x}) + \nabla g_j(\hat{x})^\top (x' - \hat{x}). \end{aligned}$$

Since $k > 0_p$ this contradicts $(\bar{x}, \bar{\eta})$ being a weakly efficient solution of (R($\mathcal{X}$)). □

Thus, if we denote by $N^2 \subseteq \mathbb{R}^p$ a set of weakly nondominated points $\bar{\eta}$ of (R($\mathcal{X}$)) with $\bar{\eta} \in \operatorname{int}(B)$ (also for different choices of $\mathcal{X}$), then $L(N^2)$ is a lower bound set for the nondominated set $\mathcal{N}$ of (MOMICP) by Lemma 3.4 and Remark 3.5. We will state this formerly for the final outcome set L of our algorithm in Lemma 6.1.

Besides the computation of lower bounds, this particular relaxation approach has some more benefits in the context of our overall algorithm. First of all, the handling of the patches (see Algorithm 4) automatically generates linearization points, namely all the points contained in $\mathcal{D}.E$. Further, we can use (R($\mathcal{X}$)) to compute integer assignments, which we will now discuss in more detail.

Let $\mathcal{X}$ be a set of linearization points and (x, η) be a feasible point for (R($\mathcal{X}$)). In particular, $x_I \in X_I$ is an integer assignment of (MOMICP). To identify whether this integer assignment is feasible, i.e., $x_I \in S_I$, or infeasible, i.e., $x_I \notin S_I$, we solve the convex feasibility problem

$$\min_{x_C, \alpha} \alpha \quad \text{s.t.} \quad g_j(x_C, \hat{x}_I) \leq \alpha \quad \forall j \in [q], \qquad x_C \in X_C, \; \alpha \in \mathbb{R} \tag{F($\hat{x}_I$)}$$

with $\hat{x}_I = x_I$. If the optimal value $\bar{\alpha}$ of (F($\hat{x}_I$)) is positive, then $\hat{x}_I \notin S_I$. If $\bar{\alpha} \leq 0$, then $\hat{x}_I$ is a feasible integer assignment, i.e., $\hat{x}_I \in S_I$.

First, we consider the case that $\hat{x}_I \notin S_I$. Denote by $(\bar{x}_C, \bar{\alpha})$ an optimal solution of (F($\hat{x}_I$)). Then we can ensure that the same integer assignment is not generated by (R($\mathcal{X}$)) again by including $(\bar{x}_C, \hat{x}_I)$ in the set $\mathcal{X}$ of linearization points. This was already shown in [15, Lemma 1] for a more general feasibility problem that includes (F($\hat{x}_I$)) as a special case. Hence, we include that lemma here adapted to our notation.

Lemma 5.2 *Let $\hat{x}_I \notin S_I$ and $(\bar{x}_C, \bar{\alpha})$ be an optimal solution of (F($\hat{x}_I$)). Further, define $\bar{x} := (\bar{x}_C, \hat{x}_I)$. Then for every $x \in X_{\hat{x}_I}$ there exists at least one index $j \in [q]$ such that x violates the constraint*

$$g_j(\bar{x}) + \nabla g_j(\bar{x})^\top (x - \bar{x}) \leq 0.$$

Next, we discuss the feasible integer assignments $\hat{x}_I \in S_I$. Such a feasible integer assignment can be generated by (R($\mathcal{X}$)) infinitely often, i.e., for an arbitrary choice of $\mathcal{X}$ there always exists a feasible point (x, η) for (R($\mathcal{X}$)) with $x_I = \hat{x}_I$. This is mainly because (R($\mathcal{X}$)) is a relaxation of (MOMICP) and there always exists a feasible point $x \in S$ for (MOMICP) with $x_I = \hat{x}_I$.

When $\hat{x}_I$ is generated by (R($\mathcal{X}$)), i.e., found as part of a weakly efficient solution of (R($\mathcal{X}$)), for the first time, then Algorithm 3 will be called to initialize the corresponding $\mathcal{D}(\hat{x}_I)$. As long as $\mathcal{D}(\hat{x}_I).S = \text{true}$, the corresponding patch (in particular the lower bounds) will be improved by Algorithm 4 each time this integer assignment is generated by (R($\mathcal{X}$)).

The challenging part is when $\hat{x}_I$ is generated by (R($\mathcal{X}$)) but the corresponding patch is inactive, i.e., $\mathcal{D}(\hat{x}_I).S = \text{false}$. In that situation we need another mechanism to keep improving the enclosure or compute a new integer assignment. We handle this as follows: If there exists another active patch, i.e., some $\hat{x}'_I \in S_I$ with $\mathcal{D}(\hat{x}'_I).S = \text{true}$, then we improve that patch using Algorithm 4. If no more active integer assignments are available, then we need to compute a new integer assignment. Since this can be done in various ways, we consider this step as a black box, see Algorithm 5. We briefly describe a realization of that algorithm when presenting our numerical results in Section 7.

For the computation of lower bounds, see Lemma 5.1, and for the computation of integer assignments, we need to compute weakly efficient solutions of (R($\mathcal{X}$)). By now, the possibilities to efficiently solve multi-objective mixed-integer linear problems are limited. While there exist some algorithms to solve bi- and tri-objective instances (e.g., [2, 23]), this is not the case for higher dimensional image spaces. Moreover, solvers for single-objective mixed-integer linear optimization problems like CPLEX [19] or Gurobi [16] have become extremely fast. Since we only need to compute one integer assignment, i.e., one feasible point of (R($\mathcal{X}$)), at a time, we decided to compute weakly efficient solutions of (R($\mathcal{X}$)) by using a scalarization approach. More precisely, we follow the same approach as on the patch level. This means, we loop through the lower bound

Algorithm 5 Search new integer assignment

```
Input: Linearization points $\mathcal{X}$, integer data structure $\mathcal{D}$
Output: Updated set $\mathcal{X}$, integer data structure $\mathcal{D}$ (, bound sets $L, U$)
 1: procedure SNIA($\mathcal{X}, \mathcal{D}$)
 2:     Search new $\tilde{x} \in X$ such that there exists no $x \in \mathcal{X}$ with $x_I = \tilde{x}_I$
            and $\mathcal{D}(\tilde{x}_I)$ is not initialized
 3:     if no such $\tilde{x}$ exists then
 4:         Let $L := \{y \in \mathcal{D}.L \mid y$ is nondominated given $\mathcal{D}.L$ w.r.t $\leq\}$
 5:         Terminate Algorithm 6 with output sets $L, U$
 6:     else if $\tilde{x}_I \in S_I$ then
 7:         InitIDS($\tilde{x}_I$)                                  ▷ see Algorithm 3
 8:     else
 9:         Solve (F($\tilde{x}_I$)) with optimal solution $(\bar{x}_C, \bar{\alpha})$
10:         Update linearization points: $\mathcal{X} = \mathcal{X} \cup \{(\bar{x}_C, \tilde{x}_I)\}$
11:     end if
12: end procedure
```

set L and for each $l \in L$ select the upper bound $u \in U$ with maximal $s(l,u)$, see Algorithm 6. Given $l, u \in \mathbb{R}^p$ with $l < u$ we then compute a weakly efficient solution of (R($\mathcal{X}$)) by solving the single-objective mixed-integer linear optimization problem

$$\begin{aligned} \min_{x,\eta,t}\ t \quad \text{s.t.} \quad & \eta - l - t(u-l) \leq 0_p, \\ & f_i(\hat{x}) + \nabla f_i(\hat{x})^\top (x - \hat{x}) \leq \eta_i \quad \forall i \in [p],\ \forall \hat{x} \in \mathcal{X}, \\ & g_j(\hat{x}) + \nabla g_j(\hat{x})^\top (x - \hat{x}) \leq 0 \quad \forall j \in [q],\ \forall \hat{x} \in \mathcal{X}, \\ & x \in X,\ \eta \in \mathbb{R}^p,\ t \in \mathbb{R}. \end{aligned} \tag{RSUP($\mathcal{X}, l, u$)}$$

Again, we have by [17, Proposition 2.3.4 and Theorem 2.3.1] that for all nonempty sets $\mathcal{X}$ and all $l, u \in \mathbb{R}^p$ with $l < u$ there exists an optimal solution $(\bar{x}, \bar{\eta}, \bar{t})$ of (RSUP($\mathcal{X}, l, u$)). By [22, Theorem 3.2] we also know that for every optimal solution $(\bar{x}, \bar{\eta}, \bar{t})$ of (RSUP($\mathcal{X}, l, u$)) the point $(\bar{x}, \hat{\eta})$ is a weakly efficient solution of (R($\mathcal{X}$)).

6 Main Algorithm HyPaD

In the previous sections we presented our method to compute upper bounds as well as two methods to compute lower bounds for the computation of an enclosure $\mathcal{A}(L, U)$ of the nondominated set $\mathcal{N}$ of (MOMICP). In this section, we finally merge all these mechanisms and present our new hybrid algorithm to compute that enclosure. We call it a hybrid algorithm since it is an ongoing interplay between improving the global lower bound set using the methods from Section 5 and iteratively improving specific patches and the corresponding lower bound sets, see Section 4.

For a better overview, we have split the pseudocode of our algorithm into the part that handles the computation of integer assignments and global lower bounds, see Algorithm 6, and the part that handles the patches, see Algorithm 7. We start by briefly explaining Algorithm 6. As described in the previous section, that part of our algorithm solves (RSUP($\mathcal{X}, l, u$)) with $l \in L$ and $u \in U$ to obtain a weakly efficient solution $(\bar{x}, \bar{\eta})$ and in particular a weakly nondominated point $\bar{\eta} \in \mathbb{R}^p$ of (R($\mathcal{X}$)). The weakly nondominated point $\bar{\eta}$ is used to update the lower bound set L which is indeed a lower bound set in the sense of Definition 3.1.

Algorithm 6 Hybrid patch decomposition algorithm for (MOMICP)

```
Input: Initial point $\hat{x} \in X$, quality $\varepsilon > 0$, and initial bounds $z, Z \in \mathbb{R}^p$
Output: Lower and upper bound sets $L, U \subseteq \mathbb{R}^p$
 1: procedure HyPaD($\hat{x}, \varepsilon, z, Z$)
 2:     Initalize $L = \{z\},\ U = \{Z\},\ \mathcal{X} = \{\hat{x}\},\ \mathcal{D} = ()$
 3:     Solve (F($\hat{x}_I$)) with optimal solution $(\bar{x}_C, \bar{\alpha})$ and set $\bar{x} := (\bar{x}_C, \hat{x}_I)$
 4:     if $\bar{\alpha} \leq 0$ then
 5:         InitIDS($\hat{x}_I$)                                        ▷ see Algorithm 3
 6:         Update linearization points: $\mathcal{X} = \mathcal{X} \cup \mathcal{D}.E$
 7:     end if
 8:     while $w(\mathcal{A}(L, U)) > \varepsilon$ do
 9:         for $l \in L$ do
10:             if $(\{l + \varepsilon e\} + \mathrm{int}(\mathbb{R}^p_+)) \cap U \neq \emptyset$ then
11:                 Select $u \in (\{l + \varepsilon e\} + \mathrm{int}(\mathbb{R}^p_+)) \cap U$ with maximal $s(l, u)$
12:                 Solve (RSUP($\mathcal{X}, l, u$)) with optimal solution $(\bar{x}, \bar{\eta}, \bar{t})$
13:                 Update lower bound set: $L = $ UpdateLLB($L, \bar{\eta}$)
14:                 Solve (F($\bar{x}_I$)) with optimal solution $(\hat{x}_C, \hat{\alpha})$ and set $\hat{x} := (\hat{x}_C, \bar{x}_I)$
15:                 if $\hat{\alpha} \leq 0$ then
16:                     Improve($\hat{x}, \varepsilon, U, \mathcal{X}, \mathcal{D}$)          ▷ see Algorithm 7
17:                     Update linearization points: $\mathcal{X} = \mathcal{X} \cup \mathcal{D}.E$
18:                 else
19:                     Update linearization points: $\mathcal{X} = \mathcal{X} \cup \{\hat{x}\}$
20:                 end if
21:             end if
22:         end for
23:     end while
24: end procedure
```

Lemma 6.1 *Assume that Algorithm 6 is not terminated by Algorithm 5. Let L be the lower bound set at some arbitrary point in the algorithm. Then L is a lower bound set in the sense of Definition 3.1.*

Proof. Since Algorithm 6 is not terminated by Algorithm 5, the set L is only updated in Algorithm 6, line 13. We know by [13, Lemma 3.7] that since L is only updated using Algorithm 2, it is a local lower bound set. We start with $L = \{z\}$ and update this set using weakly nondominated points $\bar{\eta}$ of (R($\mathcal{X}$)) as update points for Algorithm 2. Without loss of generality, we only consider those points $\bar{\eta} \in \mathbb{R}^p$ with $\bar{\eta} > z$ since for an update point $\bar{\eta} \not> z$ the lower bound set L would not be updated by Algorithm 2. Further, by the construction of the upper bound set U (see Algorithm 1) we know that for all $u \in U$ there exists $x \in S$ with $f(x) \leq u, f(x) < Z$. This implies that for every optimal solution $(\bar{x}, \bar{\eta}, \bar{t})$ of (RSUP($\mathcal{X}, l, u$)) we have $\bar{\eta} < Z$. Thus, denote by $\bar{N} \subseteq \mathrm{int}(B)$ the set of all those update points $\bar{\eta}$, i.e., we start with $\bar{N} = \emptyset$ and then add the update points $\bar{\eta} \in \mathrm{int}(B)$ obtained by solving (RSUP($\mathcal{X}, l, u$)) for different choices of $\mathcal{X}, l, u$. By Lemma 5.1 it holds that $\bar{N} \subseteq \mathrm{int}(B) \setminus (f(S) + \mathrm{int}(\mathbb{R}^p_+))$. Together with Lemma 3.4 and Remark 3.5 this implies that L is a lower bound set in the sense of Definition 3.1. □

The case that Algorithm 6 is terminated by Algorithm 5 will be considered in Lemma 6.3. There, we will show that also in that case the output set L of Algorithm 6 is a lower bound set in the sense of Definition 3.1.

For the integer assignment $\bar{x}_I$ obtained from an optimal solution $(\bar{x}, \bar{\eta}, \bar{t})$ of the subproblem (RSUP$(\mathcal{X}, l, u)$), see line 12 of Algorithm 6, we solve the corresponding feasibility problem (F$(\bar{x}_I)$) with optimal solution $(\hat{x}_C, \hat{\alpha})$ to decide whether that integer assignment is feasible, i.e., $\bar{x}_I \in S_I$, or not. The following result is then obtained using Lemma 5.2.

Corollary 6.2 *Let $\hat{x}_I \notin S_I$ be an infeasible integer assignment. Then Algorithm 6 (line 12) computes at most once an optimal solution $(\bar{x}, \bar{\eta}, \bar{t})$ of (RSUP$(\mathcal{X}, l, u)$) with $\bar{x}_I = \hat{x}_I$.*

Next, we focus on feasible integer assignments $\bar{x}_I \in S_I$. Denote by $(\hat{x}_C, \hat{\alpha})$ an optimal solution of the corresponding feasibility problem (F$(\bar{x}_I)$). Then $\hat{\alpha} \leq 0$ and Algorithm 6 (line 16) calls Algorithm 7 to perform an improvement on the patch-level.

Algorithm 7 Improvement Step on Patch-Level

```
Input: Feasible point $\hat{x} \in S$, quality $\varepsilon > 0$, upper bound set $U$, set of linearization
    points $\mathcal{X}$, and integer data structure $\mathcal{D}$
Output: Updated sets $U, \mathcal{X}$, and updated data structure $\mathcal{D}$
 1: procedure IMPROVE($\hat{x}, \varepsilon, U, \mathcal{X}, \mathcal{D}$)
 2:     if $\mathcal{D}(\hat{x}_I)$ is not initialized then
 3:         INITIDS($\hat{x}_I$)                                      ▷ see Algorithm 3
 4:     else if $\mathcal{D}(\hat{x}_I).S$ == true then
 5:         UPDATEIDS($\hat{x}_I, \varepsilon, U, \mathcal{D}$)       ▷ see Algorithm 4
 6:     else if $\exists x'_I \in S_I$ with $\mathcal{D}(x'_I)$ initialized and $\mathcal{D}(x'_I).S$ == true then
 7:         UPDATEIDS($x'_I, \varepsilon, U, \mathcal{D}$)             ▷ see Algorithm 4
 8:     else
 9:         SNIA($\mathcal{X}, \mathcal{D}$)                           ▷ see Algorithm 5
10:     end if
11: end procedure
```

For a feasible integer assignment $\hat{x}_I \in S_I$ there are exactly three possibilities concerning the handling of the corresponding patch (P$(\hat{x}_I)$), see also Section 4. The first scenario is that this particular integer assignment was computed by Algorithm 6 for the first time. This means that the corresponding entry $\mathcal{D}(\hat{x}_I)$ of the integer data structure is not yet initialized. Hence, Algorithm 7 calls Algorithm 3 for the initialization. The second case is that $\hat{x}_I$ already appeared in the algorithm before and that the integer assignment is active, i.e., $\mathcal{D}(\hat{x}_I)$ has already been initialized and it is $\mathcal{D}(\hat{x}_I).S$ = true. As a result, Algorithm 7 calls Algorithm 4 for further improvement, in particular with regard to the corresponding lower bound set $L_{\hat{x}_I}$. The final case to consider is that $\mathcal{D}(\hat{x}_I)$ is initialized but $\mathcal{D}(\hat{x}_I).S$ = false, i.e., the integer assignment is inactive. In that setting Algorithm 7 implements exactly the approach that we already discussed in Section 5 to ensure further progress of the overall algorithm. First, Algorithm 7 checks if there exists another active integer assignment, i.e., some $x'_I \in S_I$ with $\mathcal{D}(x'_I)$ initialized and $\mathcal{D}(x'_I).S$ = true. If this is the case, then Algorithm 4 is called to improve the corresponding patch. Otherwise, Algorithm 5 is called.

Algorithm 5 now searches for a new integer assignment $\tilde{x}_I \in X_I$ such that there exists no $x \in \mathcal{X}$ with $x_I = \tilde{x}_I$ and $\mathcal{D}(\tilde{x}_I)$ is not yet initialized. This leads to one of the following three situations. The first situation is that all integer assignments $x_I \in X_I$ have already been computed. Then Algorithm 5 terminates Algorithm 6 and the global lower bound set L is computed using the lower bounds from the patches. This leads indeed to a lower bound set in the sense of Definition 3.1 as the following lemma shows.

Lemma 6.3 *Assume that Algorithm 6 is terminated by Algorithm 5. Let L be the lower bound set computed in Algorithm 5, line 4. Then L is a lower bound set in the sense of Definition 3.1.*

Proof. Denote by $\mathcal{N}$ the nondominated set of (MOMICP) and let $\bar{y} \in \mathcal{N} \subseteq f(S)$ be some nondominated point. Then there exists $\bar{x} = (\bar{x}_C, \bar{x}_I) \in S$ such that $\bar{y} = f(\bar{x})$ and $\mathcal{D}(\bar{x}_I)$ was initialized and updated until $\mathcal{D}(\bar{x}_I).S$ was set to false in Algorithm 4, line 19. Thus, there exists $l \in \mathcal{D}(\bar{x}_I).L$ with $l \leq \bar{y}$. The computation of L ensures that there exists an $l' \in L$ with $l' \leq l \leq \bar{y}$ and as a result, L is a lower bound set in the sense of Definition 3.1. □

The second situation which can occur in Algorithm 5 is that a new integer assignment $\tilde{x}_I \in X_I$ is computed and that this integer assignment is feasible, i.e., $\tilde{x}_I \in S_I$. In this case, Algorithm 3 is called to initialize the corresponding patch with the corresponding entry $\mathcal{D}(\tilde{x}_I)$ of the integer data structure. The final case is that Algorithm 5 computes an infeasible integer assignment $\tilde{x}_I \notin S_I$. In that case we solve (F($\tilde{x}_I$)) with optimal solution $(\bar{x}_C, \bar{\alpha})$ and $(\bar{x}_C, \tilde{x}_I) \in X$ is included in the set $\mathcal{X}$ of linearization points to ensure that the same integer assignment is not considered again within Algorithm 6.

Before we discuss the overall correctness of Algorithm 6, i.e., that it really computes an enclosure $\mathcal{A}$ of the nondominated set $\mathcal{N}$ of (MOMICP), we need to ensure that it is finite. Thereby we make use of our assumption that there are only finitely many integer assignments $x_I \in X_I$ and the fact that each patch is only considered finitely many times by Algorithm 7.

Lemma 6.4 *Let $\hat{x}_I \in S_I$ be a feasible integer assignment with $\mathcal{D}(\hat{x}_I)$ initialized. Then after finitely many calls of Algorithm 4 (with input $\hat{x}_I$) $\mathcal{D}(\hat{x}_I).S$ is set to false.*

Proof. Let $\hat{x}_I \in S_I$ be a feasible integer assignment with $\mathcal{D}(\hat{x}_I)$ initialized and $\mathcal{D}(\hat{x}_I).S =$ true. We know by Theorem 4.1 that after a single call of Algorithm 4 either the volume of the coverage $\mathcal{C}$ of the nondominated set $\mathcal{N}_{\hat{x}_I}$ of (P($\hat{x}_I$)) is reduced by at least $\left(\frac{\varepsilon}{2}\right)^p$ or $\mathcal{D}(\hat{x}_I).S$ is set to false.

Hence, we only need to show that between two subsequent calls of Algorithm 4 the volume of $\mathcal{C} = \mathcal{C}(\mathcal{D}(\hat{x}_I).L, U)$ does not increase. First of all, the set $\mathcal{D}(\hat{x}_I).L$ of lower bounds does not change outside of Algorithm 4. Thus, we consider the development of the upper bound set U between two subsequent calls of Algorithm 4. Let U^1 be the upper bound set at the end of the first call and U^2 be the upper bound set at the beginning of the second one. The upper bound set (from Algorithm 6) is only updated in Algorithm 4, line 15, for different input parameters, especially for different integer assignments. In particular, it is always updated by Algorithm 1 that computes a new generation of upper bounds as a projection of the old generation. For our setting this means that for every $u \in U^2$ there exists $u' \in U^1$ with $u \leq u'$. As a result, we have that $\mathrm{vol}(\mathcal{C}(\mathcal{D}(\hat{x}_I).L, U^2)) \leq \mathrm{vol}(\mathcal{C}(\mathcal{D}(\hat{x}_I).L, U^1))$.

This implies that after at most $\left\lceil \mathrm{vol}(B)/\left(\frac{\varepsilon}{2}\right)^p \right\rceil$ calls of Algorithm 4 for $\hat{x}_I$ there exist no more $l \in \mathcal{D}(\hat{x}_I).L$ and $u \in U$ with $(\{l + \varepsilon e\} + \mathrm{int}(\mathbb{R}^p_+)) \cap U \neq \emptyset$ and $\mathcal{D}(\hat{x}_I).S$ is set to false. □

Finally, we can combine the results of Corollary 6.2 and Lemma 6.4 to show finiteness of Algorithm 6.

Theorem 6.5 *Algorithm 6 is finite.*

Proof. By Corollary 6.2 we know that each infeasible integer assignment is computed at most once in Algorithm 6, line 12. Since X_I and the number of infeasible integer assignments is finite, this means that for all optimal solutions $(\bar{x}, \bar{\eta}, \bar{t})$ of (RSUP($\mathcal{X}, l, u$)) (in different iterations) we have $\bar{x}_I \notin S_I$ only finitely many times.

Next, we consider the feasible integer assignments. With each iteration of the while loop where $\hat{\alpha} \leq 0$, Algorithm 7 is called. This algorithm initializes or updates $\mathcal{D}(\hat{x}_I)$ for some $\hat{x}_I \in S_I$. By Lemma 6.4 we know that for each feasible integer assignment this happens only finitely often until $\mathcal{D}(\hat{x}_I).S$ is set to false. Moreover, in case no initialization or improvement is performed by Algorithm 7, Algorithm 5 is called to compute a new integer assignment $\tilde{x}_I \in X_I$. Once again, since X_I is finite this is only possible finitely often as well.

As a result, after finitely many iterations all infeasible integer assignments have been computed and for all feasible integer assignments $\hat{x}_I \in S_I$ the entry $\mathcal{D}(\hat{x}_I).S$ is set to false. We obtain that at some point there exists no more $\tilde{x}_I \in X_I$ such that there exists $x \in \mathcal{X}$ with $x_I = \tilde{x}_I$ and $\mathcal{D}(\hat{x}_I)$ not initialized. So after a finite number of iterations either the condition $w(\mathcal{A}(L, U)) > \varepsilon$ for the while loop in Algorithm 6 is no longer fulfilled or Algorithm 5 terminates the overall algorithm. Both scenarios imply that Algorithm 6 is finite. □

Although the proof of Theorem 6.5 is based on the fact that the number of integer assignments is finite, i.e. $|X_I| < \infty$, a central goal of our algorithm is to avoid to visit all these integer assignments. This is the reason why we included the second strategy to compute a lower bound set based on the optimal solutions of (RSUP($\mathcal{X}, l, u$)). In general, the lower bound set computed by this second strategy converges towards the upper bounds with $w(\mathcal{A}(L, U)) \leq \varepsilon$ before all patches have been initialized and set inactive such that Algorithm 6 is terminated before all integer assignments have been computed.

So far we have shown in Theorem 6.5 that Algorithm 6 is finite and in Lemma 6.1 and Lemma 6.3 that the output set L is indeed a lower bound set in the sense of Definition 3.1. Thus, to show that Algorithm 6 computes indeed an enclosure in the sense of Definition 3.1 it only remains to show that the computed set U is an upper bound set.

Lemma 6.6 *At any point of Algorithm 6, the set U is an upper bound set for the nondominated set $\mathcal{N}$ of (MOMICP) in the sense of Definition 3.1.*

Proof. After the initialization of Algorithm 6 it is $U = \{Z\}$. Furthermore, U is only updated in Algorithm 4 using Algorithm 1 with update points $\bar{y} := f(\bar{x}) \in f(S)$. If we denote by $N \subseteq f(S)$ the set of all these points, then $U = U(N)$. In general, this set N is not stable, but we know by Remark 3.5 that for $N' := \{y \in N \mid y \text{ is nondominated given } N \text{ w.r.t. } \leq\}$ it holds $U(N') = U(N)$. Finally, by Lemma 3.4 this implies that U is an upper bound set in the sense of Definition 3.1. □

With all the results from this section we are now able to prove that Algorithm 6 works correct and that the enclosure $\mathcal{A}(L,U)$ corresponding to the sets L and U computed by the algorithm is of the predefined quality ε.

Theorem 6.7 *Let L and U be the output sets of Algorithm 6. Then L and U are lower and upper bound sets in the sense of Definition 3.1 and for the width of the enclosure $\mathcal{A}(L,U) = (L + \mathbb{R}^p_+) \cap (U - \mathbb{R}^p_+)$ of the nondominated set $\mathcal{N}$ of (MOMICP) it holds that $w(\mathcal{A}(L,U)) \leq \varepsilon$.*

Proof. By Lemma 6.1, Lemma 6.3, and Lemma 6.6 we have that the sets L and U computed by Algorithm 6 are lower and upper bound sets in the sense of Definition 3.1 and hence define a corresponding enclosure $\mathcal{A}(L,U)$.

To prove that $w(\mathcal{A}(L,U)) \leq \varepsilon$, we consider two cases based on the termination of Algorithm 6. The first case is that Algorithm 6 terminates because the condition $w(\mathcal{A}(L,U)) > \varepsilon$ in the while loop is no longer satisfied. Then it obviously holds $w(\mathcal{A}(L,U)) \leq \varepsilon$.

The second case to consider is that L was computed using the lower bounds of the patches in Algorithm 5, line 4. Then, for every $l \in L$ there exists an integer assignment $\hat{x}_I \in S_I$ with $l \in \mathcal{D}(\hat{x}_I).L$. At some point in the algorithm this integer assignment was set inactive and we denote by U' the upper bound set at that point. Since an integer assignment can only be set inactive by Algorithm 4, we know that for all $u' \in U'$ there existed an index $i \in [p]$ with $u'_i - l_i \leq \varepsilon$. While the upper bound set U' might have been updated after the integer assignment was set inactive, updating it using Algorithm 1 ensures that upper bounds never get worse. This means that if U^1 is the upper bound set before calling Algorithm 1 and U^2 the set afterwards, then for all $u^2 \in U^2$ there exists $u^1 \in U^1$ with $u^2 \leq u^1$. Thus, for all $u \in U$ with $l \leq u$ there exists $u' \in U'$ and an index $i \in [p]$ such that $u_i - l_i \leq u'_i - l_i \leq \varepsilon$, which implies $w(\mathcal{A}(L,U)) \leq \varepsilon$. □

7 Numerical Results

In this final section, we present our numerical results for selected test instances. In addition to that, we provide a detailed discussion on the implementation details of the HyPaD algorithm together with more than 30 test instances in [14]. We performed all numerical tests in MATLAB R2021a on a machine with Intel Core i9-10920X processor and 32GB of RAM. The average of the results of `bench(5)` is: LU = 0.2045, FFT = 0.2127, ODE = 0.3666, Sparse = 0.3919, 2-D = 0.1968, 3-D = 0.2290.

The initial bounds $z, Z \in \mathbb{R}^p$ are computed using interval arithmetic provided by INTLAB [27]. This also allows for a fair comparison of our results with the ones obtained by the MOMIBB algorithm from [12] since that algorithm uses the same initial bounds for its computation of an enclosure. The initial point $\hat{x} \in X$ in Algorithm 6 is always computed as an optimal solution of a scalarization of the integer-relaxed version of (MOMICP), i.e., (MOMICP) but with $x \in X_C \times [l_I, u_I] \subseteq \mathbb{R}^n \times \mathbb{R}^m$ instead of $x \in X_C \times ([l_I, u_I] \cap \mathbb{Z}^m) \subseteq \mathbb{R}^n \times \mathbb{Z}^m$. In [14] we present more details on the initialization phase of the algorithm. The offset parameter for Algorithm 4 is chosen as $\sigma := 10^{-3}\varepsilon$. If not stated otherwise, the quality parameter was set to $\varepsilon = 0.1$. For all test instances we used a time limit of 3600 seconds.

Algorithm 5 is realized by dividing the set of all integer assignments $\mathcal{X}_I$ into a fixed number of 16 boxes and then searching for a new integer assignment within the box with the least number of already computed integer assignments. This procedure is presented more in-depth in [14, Section 2.3], where it is also compared with some other possible approaches.

All of the single-objective mixed-integer linear problems within our algorithm are solved using GUROBI [16]. All single-objective (purely continuous) convex problems are solved using `fmincon`. We also tested alternative solvers, for example IPOPT [31] via OPTI [20]. However, this did not significantly improve the overall performance of our algorithm and hence, we decided to use `fmincon`. Another benefit of choosing `fmincon` is that this allows for a fair comparison of our results with those from [7] and [12].

All results in this paper for the MOMIBB algorithm from [12] have been computed using the same parameters as HyPaD, i.e., the same quality parameter $\varepsilon > 0$, the same initial box $B = [z, Z] \subseteq \mathbb{R}^p$ for the enclosure, the same solvers for the subproblems, and the same time limit of 3600 seconds. This allows a fair comparison of HyPaD and MOMIBB both qualitatively and quantitatively.

There are four different configurations of MOMIBB that have been presented in [12, Section 6] as MOMIBB-c0, MOMIBB-c1, MOMIBB-c2, and MOMIBB$_{\text{direct}}$. Thereby, the last two configurations can only be used in special cases including instances of (MOMICP) with convex quadratic objective and constraint functions. Whenever comparing HyPaD with MOMIBB in the remaining part of this section, the result for MOMIBB represents the best result (in terms of computation time) that was obtained by all configurations of MOMIBB that were applicable for the corresponding test instance.

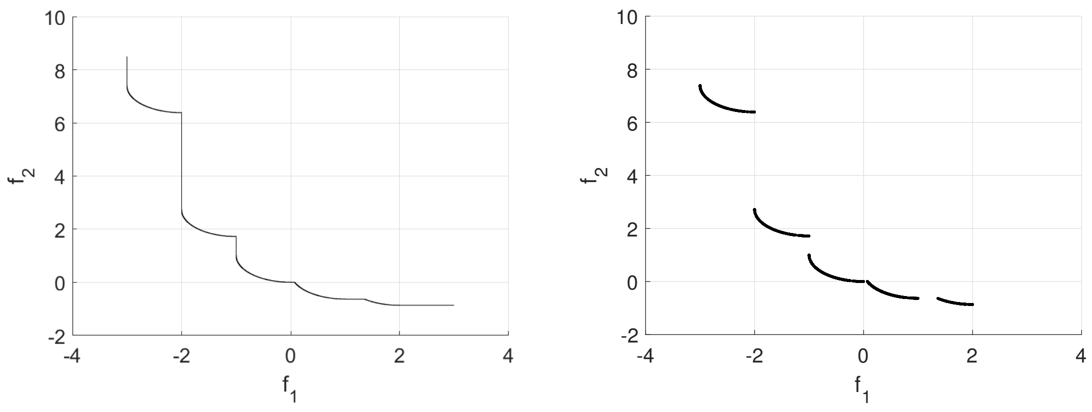
All results in this paper for MOMIX and MOMIX light from [7] have been computed for $\delta = 0.1$ as the corresponding quality parameter using the code from [6]. We have also discussed in [14] that a quantitative comparison of the results is only possible to some extent. This due to the fact that the HyPaD algorithm works with a quality criterion in the criterion space (based on the parameter $\varepsilon > 0$) and the authors of [7] work with a quality criterion in the decision space (based on their parameter $\delta > 0$). Still, qualitative comparisons between HyPaD and MOMIX are possible.

Test instance 1 First, we consider a bi-objective test instance with quadratic constraint functions and a non-quadratic objective function from [7]. Both, the number $n = 2$ of continuous and the number $m = 1$ of integer variables are fixed.

$$\min\ (x_1 + x_3, x_2 + \exp(-x_3))^\top \quad \text{s.t.} \quad \begin{aligned} & x_1^2 + x_2^2 \leq 1, \\ & x_C \in [-2, 2]^2, \\ & x_I \in [-2, 2] \cap \mathbb{Z}. \end{aligned} \tag{T6}$$

The MOMIX algorithm from [7] involves solving single-objective subproblems that are basically of the same type as the original problem (MOMICP). For (T6) these would be single-objective mixed-integer non-quadratic convex quadratically constrained optimization problems. Since those cannot be solved by Gurobi, which is used as the solver for the single-objective mixed-integer subproblems in [7], the MOMIX algorithm cannot be applied to this problem and one needs to switch to MOMIX light. The latter is a modification which uses purely continuous subproblems and those can be solved by `fmincon` (even for the non-quadratic case).

Only with (br2), which is one of two available branching rules, MOMIX light computed a representation of the nondominated set of (T6), which took 1385.72 seconds. For (br1) it exceeded the time limit of 3600 seconds. HyPaD on the other hand computes an enclosure within only a few seconds. For $\varepsilon = 0.1$ it computes an enclosure within 1.34 seconds and even for $\varepsilon = 0.01$ it only needs 6.09 seconds. For a visual comparison of the results see Figure 3.



(a) Enclosure by HyPaD (b) Representation by MOMIX

Figure 3: Results for test instance 30 ($\delta = 0.1, \varepsilon = 0.01$)

For both choices of $\varepsilon \in \{0.1, 0.01\}$, Algorithm 6 is terminated by Algorithm 5, i.e., the enclosure of the overall nondominated set is computed using the lower bounds $\mathcal{D}.L$ from the different patches and the global upper bound set U. In particular, this means that all integer assignments $x_I \in X_I$ have been computed by our algorithm. This is a very typical behavior of HyPaD if the number of integer assignments, i.e., $|X_I|$, is quite small. For (T6) this is the case since we have $|X_I| = 5$. Hence, HyPaD is a highly effective algorithm for such problems with a small number of possible integer assignments.

Another advantage of HyPaD is that the single-objective mixed-integer subproblems (RSUP($\mathcal{X}, l, u$)) are always linear. Hence, these subproblems can always be solved using Gurobi. This makes HyPaD the better choice (compared to the methods from [7]) if one of the objective or constraint functions of (MOMICP) is non-quadratic.

The MOMIBB algorithm from [12] needs 15.33 seconds to solve (T6) for $\varepsilon = 0.1$. This is roughly ten times slower than HyPaD, but also roughly ten times faster than MOMIX. Since MOMIBB also computes an enclosure and hence has a criterion space based termination criterion (the width of the enclosure), but besides that is a decision space based branch-and-bound algorithm, it is not surprising that its performance is in between HyPaD and MOMIX. A more detailed comparison of MOMIBB and HyPaD is provided for the next test instance.

Test instance 2 Next, we consider a scalable test instance with quadratic objective functions and a quadratic constraint function. Both the number $n \in \mathbb{N}$ of continuous variables and the number $m \in \mathbb{N}$ of integer variables have to be even.

		HyPaD						MOMIBB
n	m	t	$\lvert L\rvert$	$\lvert U\rvert$	#(RSUP)	#(SUP)	flag	t
2	2	2.44	42	63	50	51	1	12.41
2	4	4.71	80	123	90	97	1	86.90
2	8	25.70	222	280	293	184	1	2108.21
2	10	79.54	458	433	573	290	1	-
2	12	446.60	1198	685	1441	459	1	-
2	14	-	-	-	-	-	-1	-
4	2	3.63	52	75	59	70	1	283.32
4	4	7.91	97	149	115	141	1	1114.57
4	8	39.70	252	266	310	245	1	-
4	10	134.67	515	393	634	363	1	-
4	12	960.08	1214	546	1434	510	1	-
4	14	-	-	-	-	-	-1	-

Table 1: Computational results of HyPaD compared to MOMIBB for (H1)

$$\min \begin{pmatrix} \sum\limits_{i=1}^{n/2} x_i + \sum\limits_{i=n+1}^{n+m/2} x_i^2 - \sum\limits_{i=n+m/2+1}^{n+m} x_i \\ \sum\limits_{i=n/2+1}^{n} x_i - \sum\limits_{i=n+1}^{n+m/2} x_i + \sum\limits_{i=n+m/2+1}^{n+m} x_i^2 \end{pmatrix} \quad \text{s.t.} \quad \begin{aligned} &\sum_{i=1}^{n} x_i^2 \leq 1, \\ &x_C \in [-2,2]^n, \\ &x_I \in [-2,2]^m \cap \mathbb{Z}^m. \end{aligned} \tag{H1}$$

The computational results for various choices of n and m are shown in Tables 1 and 2. For HyPaD we included the overall computation time t, the number of computed lower and upper bounds, the number of subproblems (RSUP($\mathcal{X}, l, u$)) and (SUP($\hat{x}_I, l, u$)) as well as the exit flag which indicates whether Algorithm 6 was terminated by Algorithm 5 (flag = 0), by the condition in the main while loop (flag = 1) or by reaching the time limit (flag = -1). For MOMIBB we only included the best result (in terms of computation time) out of the four configurations that have been presented in [12, Section 6]. In Table 1 we present the results for realizations of (H1) with $n = 2$ and $n = 4$. These were the only realizations where also MOMIBB was able to compute an enclosure within the time limit of 3600 seconds. In Table 2 we then present the results for all other realizations of (H1). However, there MOMIBB was not able to compute an enclosure within the given time limit.

For all choices of n and m (where the time limit was not reached) HyPaD performs significantly better than MOMIBB. In particular, HyPaD is able to solve even large instances of (H1) with up to $n = 256$ continuous variables. One possible explanation for this could be that while MOMIBB also computes an enclosure in the criterion space, it mostly is a decision space branch-and-bound approach. As such it is usually more influenced by the number of variables than our (purely) criterion space based approach.

		HyPaD					
n	m	t	$\|L\|$	$\|U\|$	#(RSUP)	#(SUP)	flag
8	2	4.66	61	77	71	74	1
8	4	8.60	106	162	124	161	1
8	8	45.16	278	280	356	267	1
8	10	179.76	582	428	721	406	1
8	12	-	-	-	-	-	-1
8	14	-	-	-	-	-	-1
16	2	6.86	81	88	78	94	0
16	4	14.34	121	220	142	246	1
16	8	75.05	339	370	416	395	1
16	10	310.88	721	545	868	575	1
16	12	1730.75	1432	708	1696	709	1
16	14	-	-	-	-	-	-1
32	2	12.03	103	108	90	130	0
32	4	21.35	129	213	150	261	1
32	8	130.31	374	406	467	495	1
32	10	648.08	813	560	947	672	1
32	12	-	-	-	-	-	-1
32	14	-	-	-	-	-	-1
64	8	262.16	434	419	514	628	1
64	10	1115.92	787	543	940	912	1
64	12	-	-	-	-	-	-1
64	14	-	-	-	-	-	-1
128	8	547.94	461	492	527	851	1
256	8	1602.95	476	544	542	1070	1
512	8	-	-	-	-	-	-1

Table 2: Computational results of HyPaD for (H1)

Test instance 3 With the following tri-objective test instance from [7] we illustrate the capability of our algorithm to handle optimization problems (MOMICP) with three and more objective functions.

$$\min \begin{pmatrix} x_1 + x_4 \\ x_2 - x_4 \\ x_3 + x_4^2 \end{pmatrix} \quad \text{s.t.} \quad \begin{aligned} & \sum_{i=1}^{3} x_i^2 \leq 1, \\ & x_C \in [-2,2]^3, \\ & x_I \in \{-2,-1,0,1,2\}. \end{aligned} \tag{T5}$$

We present here the results for different choices of ε such that the reader gets a better idea of the impact of that parameter. For a visualization of the improvement comparing $\varepsilon = 0.5$ and $\varepsilon = 0.1$ see Figure 4.

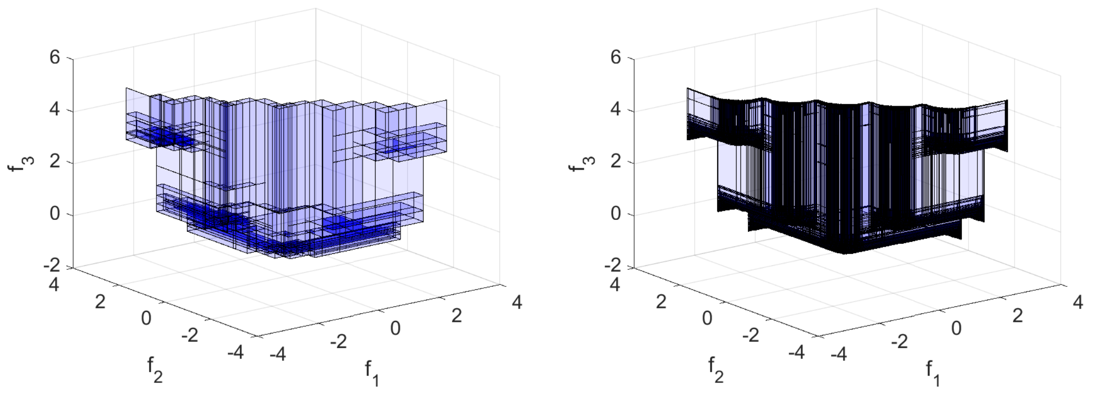


Figure 4: The enclosure computed for (T5) with $\varepsilon = 0.5$ on the left and $\varepsilon = 0.1$ on the right

While MOMIX (in the best of all four configurations) needs about 90 seconds to compute the result for $\delta = 0.5$, the HyPaD algorithm needs only about 9 seconds to compute the result for $\varepsilon = 0.1$. Again, the performance of MOMIBB, with a computation time of roughly 75 seconds needed for $\varepsilon = 0.1$, is in between HyPaD and MOMIX.

Acknowledgments

This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project-ID 432218631.

References

[1] N. Boland, H. Charkhgard, and M. Savelsbergh, *A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method*, INFORMS Journal on Computing, 27 (2015), pp. 597–618.

[2] N. Boland, H. Charkhgard, and M. Savelsbergh, *The l-shape search method for triobjective integer programming*, Mathematical Programming Computation, 8 (2015), pp. 217–251.

[3] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter, *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optimization, 5 (2008), pp. 186–204.

[4] R. S. Burachik, C. Y. Kaya, and M. M. Rizvi, *Algorithms for generating pareto fronts of multi-objective integer and mixed-integer programming problems*, Engineering Optimization, (2021), https://doi.org/10.1080/0305215x.2021.1939695.

[5] G. Cabrera-Guerrero, M. Ehrgott, A. J. Mason, and A. Raith, *Bi-objective optimisation over a set of convex sub-problems*, Annals of Operations Research, (2021), https://doi.org/10.1007/s10479-020-03910-3.

[6] M. De Santis, G. Eichfelder, J. Niebling, and S. Rocktäschel, *MOMIX*. https://github.com/mariannadesantis/MOMIX. Accessed 2022-06-23.

[7] M. De Santis, G. Eichfelder, J. Niebling, and S. Rocktäschel, *Solving multiobjective mixed integer convex optimization problems*, SIAM Journal on Optimization, 30 (2020), pp. 3122–3145.

[8] E. Diessel, *An adaptive patch approximation algorithm for bicriteria convex mixed-integer problems*, Optimization, (2021), https://doi.org/10.1080/02331934.2021.1939699.

[9] M. Ehrgott, *Multicriteria optimization*, Springer, 2005.

[10] M. Ehrgott and X. Gandibleux, *Bound sets for biobjective combinatorial optimization problems*, Computers & Operations Research, 34 (2007), pp. 2674–2694.

[11] G. Eichfelder, P. Kirst, L. Meng, and O. Stein, *A general branch-and-bound framework for continuous global multiobjective optimization*, Journal of Global Optimization, 80 (2021), pp. 195–227.

[12] G. Eichfelder, O. Stein, and L. Warnow, *A deterministic solver for multiobjective mixed-integer convex and nonconvex optimization*. http://www.optimization-online.org/DB_HTML/2022/02/8796.html, 2022.

[13] G. Eichfelder and L. Warnow, *An approximation algorithm for multi-objective optimization problems using a box-coverage*, Journal of Global Optimization, 83 (2021), pp. 329–357.

[14] G. Eichfelder and L. Warnow, *On implementation details and numerical experiments for the HyPaD algorithm to solve multi-objective mixed-integer convex optimization problems*. http://www.optimization-online.org/DB_HTML/2021/08/8538.html, 2021.

[15] R. Fletcher and S. Leyffer, *Solving mixed integer nonlinear programs by outer approximation*, Mathematical Programming, 66 (1994), pp. 327–349.

[16] Gurobi Optimization LLC, *Gurobi*. https://www.gurobi.com/. Accessed 2022-06-23.

[17] A. Göpfert, H. Riahi, C. Tammer, and C. Zalinescu, *Variational Methods in Partially Ordered Spaces*, Springer, 2003.

[18] P. Halffmann, L. E. Schäfer, K. Dächert, K. Klamroth, and S. Ruzika, *Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey*, Journal of Multi-Criteria Decision Analysis, (2022), https://doi.org/10.1002/mcda.1780.

[19] IBM, *CPLEX optimizer*. https://www.ibm.com/analytics/cplex-optimizer. Accessed 2022-06-23.

[20] Inverse Problem Ltd, *OPTI toolbox*. https://www.inverseproblem.co.nz/OPTI/index.php. Accessed 2022-06-23.

[21] K. Klamroth, R. Lacour, and D. Vanderpooten, *On the representation of the search region in multi-objective optimization*, European Journal of Operational Research, 245 (2015), pp. 767–778.

[22] A. Pascoletti and P. Serafini, *Scalarizing vector optimization problems*, Journal of Optimization Theory and Applications, 42 (1984), pp. 499–524.

[23] T. Perini, N. Boland, D. Pecin, and M. Savelsbergh, *A criterion space method for biobjective mixed integer programming: The boxed line method*, INFORMS Journal on Computing, 32 (2020), pp. 16–39.

[24] A. Przybylski, K. Klamroth, and R. Lacour, *A simple and efficient dichotomic search algorithm for multi-objective mixed integer linear programs*. https://arxiv.org/abs/1911.08937, 2019.

[25] S. A. B. Rasmi and M. Türkay, *GoNDEF: an exact method to generate all non-dominated points of multi-objective mixed-integer linear programs*, Optimization and Engineering, 20 (2019), pp. 89–117.

[26] R. Roozbahani, B. Abbasi, and S. Schreider, *Optimal allocation of water to competing stakeholders in a shared watershed*, Annals of Operations Research, 229 (2015), pp. 657–676.

[27] S. Rump, *INTLAB - INTerval LABoratory*, in Developments in Reliable Computing, T. Csendes, ed., Kluwer Academic Publishers, 1999, pp. 77–104.

[28] S. Ruzika and M. M. Wiecek, *Approximation methods in multiobjective programming*, Journal of Optimization Theory and Applications, 126 (2005), pp. 473–501.

[29] S. K. Singh and M. Goh, *Multi-objective mixed integer programming and an application in a pharmaceutical supply chain*, International Journal of Production Research, 57 (2018), pp. 1214–1237.

[30] B. Soylu and G. B. Yildiz, *An exact algorithm for biobjective mixed integer linear programming problems*, Computers & Operations Research, 72 (2016), pp. 204–213.

[31] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Mathematical Programming, 106 (2005), pp. 25–57.

[32] P. Xidonas, G. Mavrotas, and J. Psarras, *Equity portfolio construction and selection using multiobjective mathematical programming*, Journal of Global Optimization, 47 (2009), pp. 185–209.

[33] Ö. Özpeynirci and M. Köksalan, *An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs*, Management Science, 56 (2010), pp. 2302–2315.