Technische Universität Ilmenau
Institut für Mathematik

Preprint No. M 19/06

Solving multiobjective mixed integer convex optimization problems

Marianna de Santis, Gabriele Eichfelder, Julia Niebling, Stefan Rocktäschel

Mai 2019

URN: urn:nbn:de:gbv:ilm1-2019200296

Impressum:
Hrsg.: Leiter des Instituts für Mathematik
Weimarer Straße 25
98693 Ilmenau
Tel.: +49 3677 69-3621
Fax: +49 3677 69-3270
http://www.tu-ilmenau.de/math/

ilmedia

Solving Multiobjective Mixed Integer Convex Optimization Problems

Marianna de Santis[*], Gabriele Eichfelder[**], Julia Niebling[**], Stefan Rocktäschel[**]

May 28, 2019

Abstract

Multiobjective mixed integer convex optimization refers to mathematical programming problems where more than one convex objective function needs to be optimized simultaneously and some of the variables are constrained to take integer values. We present a branch-and-bound method based on the use of properly defined lower bounds. We do not simply rely on convex relaxations, but we built linear outer approximations of the image set in an adaptive way. We are able to guarantee correctness in terms of detecting both the efficient and the nondominated set of multiobjective mixed integer convex problems according to a prescribed precision. As far as we know, the procedure we present is the first deterministic algorithm devised to handle this class of problems. Our numerical experiments show results on biobjective and triobjective mixed integer convex instances.

Key Words: Multiobjective Optimization, Mixed Integer Convex Programming

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

1 Introduction

Multiobjective programming is concerned with mathematical problems where more than one objective function needs to be optimized simultaneously. When the problem considered involves both continuous and integer variables we are in the context of multiobjective mixed integer programming. In this paper, we focus on multiobjective mixed integer convex programming problems, namely problems of the following form:

$$\begin{array}{ll} \min & (f_1(x), \ldots, f_m(x))^T \\ \text{s.t.} & g_k(x) \leq 0 \quad k = 1, \ldots, p \\ & x \in B := [l, u] \\ & x_i \in \mathbb{Z} \quad \forall i \in I, \end{array} \tag{MOMIC}$$

[*]Department of Computer, Control and Management Engineering, Sapienza Università di Roma, Via Ariosto 25 00185 Roma, Italy, mdesantis@diag.uniroma1.it

[**]Institute for Mathematics, Technische Universität Ilmenau, Po 10 05 65, D-98684 Ilmenau, Germany, {gabriele.eichfelder,julia.niebling,stefan.rocktaeschel}@tu-ilmenau.de

where $f_j, g_k : B \to \mathbb{R};\ j = 1, \dots, m;\ k = 1, \dots, p$ are convex and continuously differentiable functions. The vectors $l, u \in \mathbb{R}^n$ are lower and upper bounds on the decision variables $x \in \mathbb{R}^n$ and define the box B. The index set $I \subseteq \{1, \dots, n\}$ specifies which variables have to take integer values. We assume w.l.o.g. $l_i, u_i \in \mathbb{Z}$ for all $i \in I$. The image of the feasible set of the problem under the vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ represents the feasible set in the criterion space, or the image set.

Multiobjective mixed integer optimization problems arise in many application fields such as location or production planning, finance, manufacturing, and emergency management (see e.g. [14, 28, 30]). As an example we can think of the uncapacitated facility location problem, studied in the single-objective case in [18]. The first objective hereby is to decide which facilities to build in order to minimize costs. As a second objective function one could consider the total negative impact on the environment with the building plan for the facilities, e.g. the carbon emissions.

Solving a multiobjective optimization problem aims at detecting the efficient set, namely the set of points in the decision space that leads to nondominated points in the criterion space. A point of the image set is nondominated if none of its components can be decreased without increasing any other component. A formal definition will be given in Section 2.

It is well known that mixed integer nonlinear optimization is NP-hard and its solution typically requires dealing with enormous search trees [1]. Handling more than one objective function adds an additional difficulty: assume there is only one binary variable, $I = 1$ with $x_1 \in \{0, 1\}$, and we have just one objective function, i. e., $m = 1$. Then for solving (MOMIC) only two convex optimization problems have to be addressed, one with x_1 fixed to 0 and one with x_1 fixed to 1. Clearly, the smallest minimal value is the optimal value of the original problem. In case of two or more objective functions already this simple setting is much more challenging. Solving the problems with fixed values for x_1 would mean to determine the whole efficient set of a multiobjective convex optimization problem, which is in general infinite. Then, after computing two sets of nondominated points one has to compare them and to determine the "smallest" values, see Figure 1 on page 5 for an illustration of this observation for four choices of the integer variables.

So far, there exist mostly algorithms for solving multiobjective mixed integer *linear* programming problems only. Those can be divided into two main classes: decision space search algorithms, i.e., approaches that work in the space of feasible points, and criterion space search algorithms, i.e., methods that work in the space of objective function values.

Among the decision space search algorithms, the method proposed by Mavrotas and Diakoulaki, [24], is the first branch-and-bound algorithm for solving multiobjective mixed binary programs. The authors improved and extended their work in [23, 25]. Other works defining branch-and-bound algorithms for multiobjective integer linear programming problems are [12, 29]. There, in the bounding procedure the aim is to define proper hypersurfaces in the objective space in order to separate the upper and lower bound sets.

Criterion space search algorithms find nondominated points by addressing a sequence of single-objective optimization problems. Once a nondominated point is computed, dominated parts of the criterion space are removed and the algorithms go on looking for new nondominated points. Several contributions in the context of criterion space search algorithms for biobjective and triobjective mixed integer linear programming have been given

by Boland and co-authors [2, 3, 4, 5].

As far as we know, the first general purpose method to tackle multiobjective mixed integer *convex* programs is the heuristic based on a branch-and-bound algorithm proposed by Cacchiani and D'Ambrosio in [10].

A classical technique to solve a multiobjective optimization problem is to convert the problem into a parameter-dependent single-objective one, known as *scalarization*. This approach was recently followed by Burachik et al. [8] (see also the comment in the conclusions of [9]). The scalarized problems are then parameter-dependent single-objective mixed integer convex optimization problems. By following this approach, many of these single-objective problems have to be solved, one for each choice of the parameter's value. No gained information of pre-solved problems are typically used thereby. Furthermore, it is not clear how to choose the parameter's values in a smart way and this is an open challenge: as the set of nondominated points is in general disconnected and can have huge gaps, many subproblems defined according to different parameter's values might lead to the same nondominated point and thus solving such subproblems is a wasted effort.

We propose in this paper for the first time a deterministic algorithm for multiobjective mixed integer convex problems which is not using a scalarization of the original problem. We directly develop a branch-and-bound algorithm based on a partitioning of the feasible region, i.e., a decision space algorithm. We present two versions of the algorithm. In one version we do not have to solve any single-objective mixed integer subproblem but only single-objective convex subproblems. In the second version we need to address also single-objective mixed integer convex optimization problems. No parameter needs to be chosen to define the subproblems we consider.

To compute lower bounds in our branch-and-bound approach we rely on linear outer approximations of the image set. We use outer approximation techniques from convex multiobjective optimization for finding lower bounds of the continuous relaxation of the problem (i. e., the problem obtained by ignoring the integrality constraints), as well as for constructing outer approximations of the convex hull of the true image set over subboxes. We keep track of upper bounds in the image space and derive by that a discarding test for the branch-and-bound procedure. This results in a deterministic solver for which we can give theoretical guarantees to find approximations of the set of efficient and of the set of nondominated points.

The paper is organized as follows. In Section 2 we report notations and definitions that will be used throughout the paper. In Section 3 we present our branch-and-bound algorithm `MOMIX`. Details on how to define a "light" version of the algorithm that does not need to address any single-objective mixed integer convex programming problem are given as well. Theoretical insights of `MOMIX` and $\texttt{MOMIX}_{light}$ are also given in Section 3. Some numerical results are reported in Section 4. Section 5 concludes.

2 Definitions and Notations

Throughout the paper, we indicate with $\|\cdot\|$ the Euclidean norm. Given a box $B = [l, u]$, we denote by $\omega(B)$ its width obtained as the Euclidean distance between l and u, namely $\omega(B) = \|u - l\|$. Given two vectors $x, y \in \mathbb{R}^n$, we write $x \leq y$ and $x < y$ if $x_i \leq y_i$ and

$x_i < y_i$ for all $i \in \{1, \dots, n\}$, respectively. We write $x \not\leq y$ when an index $i \in \{1, \dots, n\}$ exists such that $x_i > y_i$. Given a vector $x \in \mathbb{R}^n$ and an index set $I \subseteq \{1, \dots, n\}$, we denote with x_I the subvector with components x_i, $i \in I$.

Let $x \in \mathbb{R}$, we define

$$\lfloor x \rfloor := \max\{c \in \mathbb{Z} \mid c \leq x\}, \lceil x \rceil := \min\{c \in \mathbb{Z} \mid x \leq c\} \text{ and } [x] := \begin{cases} \lceil x \rceil & \text{if } x + 0.5 \geq \lceil x \rceil \\ \lfloor x \rfloor & \text{otherwise.} \end{cases}$$

For $x \in \mathbb{R}^n$, we define $\lfloor x \rfloor$, $\lceil x \rceil$ and $[x]$ componentwise.

For a nonempty set $A \subset \mathbb{R}^m$, we denote by $\text{conv}(A)$ the convex hull of A, namely the smallest convex set that contains A. By B^g, $B^{\mathbb{Z}}$ and $B^{g,\mathbb{Z}}$ we denote the following sets related to the constraints in (MOMIC):

$$\begin{aligned} &B^g := \{x \in B \mid g(x) \leq 0\}, \\ &B^{\mathbb{Z}} := \{x \in B \mid x_i \in \mathbb{Z} \text{ for all } i \in I\}, \\ &B^{g,\mathbb{Z}} := B^g \cap B^{\mathbb{Z}}. \end{aligned} \tag{1}$$

Using these sets, we can write (MOMIC) in compact form as

$$\begin{aligned} &\min\ f(x) \\ &\text{s.t. } x \in B^{g,\mathbb{Z}}. \end{aligned}$$

As mentioned, we are going to define a branch-and-bound method based on partitioning the feasible set of (MOMIC). Our branching rule is based on bisections of the box B. Let $\tilde{B}$ be a subbox of B. By $\tilde{B}^g$, $\tilde{B}^{\mathbb{Z}}$ and $\tilde{B}^{g,\mathbb{Z}}$ we denote the sets defined according to (1), where the set B is replaced by $\tilde{B}$ (i.e., $x \in \tilde{B}$ in all the set definitions).

We recall here the basic concepts of efficient and nondominated points (see [20] for further details).

Definition 2.1 *(a) A feasible point $x^* \in B^{g,\mathbb{Z}}$ is efficient for (MOMIC) if there is no $x \in B^{g,\mathbb{Z}}$ with $f(x) \leq f(x^*)$ and $f(x) \neq f(x^*)$. The set of efficient points for (MOMIC) is the efficient set of (MOMIC).*

(b) A point $z^* = f(x^*)$ is nondominated for (MOMIC) if $x^* \in B^{g,\mathbb{Z}}$ is an efficient point for (MOMIC). The set of all nondominated points of (MOMIC) is the nondominated set of (MOMIC).

(c) Let $x, x^* \in B^{g,\mathbb{Z}}$ with $f(x) \leq f(x^*)$ and $f(x) \neq f(x^*)$. Then we say that x dominates x^* and also that $f(x)$ dominates $f(x^*)$.

In Figure 1, we plot the image set of a biobjective mixed integer convex optimization problem. Here, we assume that $\{x_I \mid x \in B^{g,\mathbb{Z}}\} =: \{y^1, y^2, y^3, y^4\}$ and we show the sets $F_j := \{f(x) \mid x \in B^{g,\mathbb{Z}},\ x_I = y^j\}$, $j = 1, \dots, 4$. Then, $\bigcup_{j=1,\dots,4} F_j = \{f(x) \mid x \in B^{g,\mathbb{Z}}\}$. The point $z^* \in f(B^{g,\mathbb{Z}})$ is nondominated and the preimage of z^* is an efficient point. On the other hand, $z' \in f(B^{g,\mathbb{Z}})$ is dominated because $z^* \leq z'$ and $z^* \neq z'$. In fact, all the points $z \in F_3$ are dominated, as points $w \in f(B^{g,\mathbb{Z}})$ exist such that $w < z$. The nondominated set of the problem is visualized as the thick boundary of the image set. The

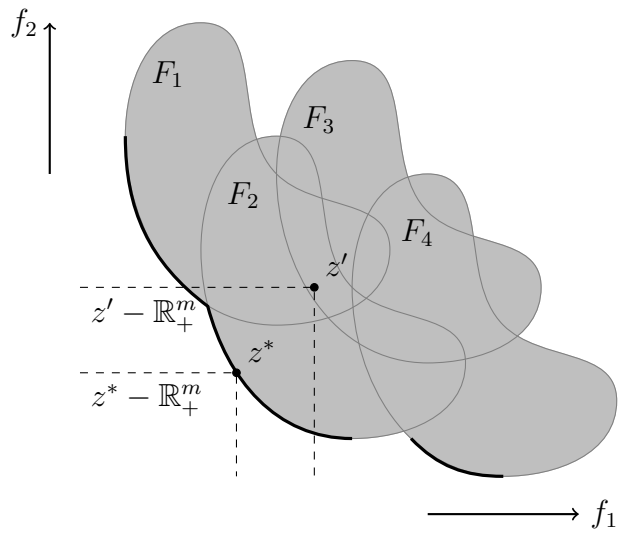


Figure 1: Image set of a biobjective instance of (MOMIC).

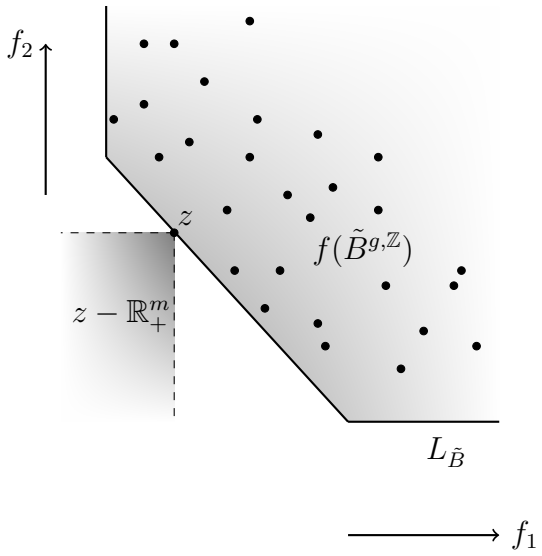


Figure 2: Image set of a biobjective purely integer instance of (MOMIC).

efficient set is made of all preimages of the nondominated set. Figure 1 shows that the nondominated set of a multiobjective mixed integer nonlinear programming problem is in general a disconnected set. From an algorithmic point of view, this makes the detection of the efficient set of (MOMIC) an extremely challenging problem. Furthermore, there is the necessity of comparing sets of points: this is a crucial difference with respect to single-objective mixed integer nonlinear optimization.

3 MOMIX: An Outer Approximation based Branch-and-Bound Algorithm for (MOMIC)

The algorithm we propose is a branch-and-bound method that looks for the efficient set of (MOMIC) by partitioning the box B. At every node of the branch-and-bound tree, a subbox $\tilde{B} \subseteq B$ is selected and lower and upper bounds on the nondominated set of (MOMIC) are derived. When considering the subbox $\tilde{B}$, a lower bound is any set $L_{\tilde{B}} \subseteq \mathbb{R}^m$ such that $L_{\tilde{B}} + \mathbb{R}^m_+$ contains the image of integer feasible points $\tilde{B}^{g,\mathbb{Z}}$ through f, namely $f(\tilde{B}^{g,\mathbb{Z}}) \subseteq L_{\tilde{B}} + \mathbb{R}^m_+$. In Figure 2, we illustrate the set $f(\tilde{B}^{g,\mathbb{Z}})$ and a lower bound $L_{\tilde{B}}$ for a biobjective purely integer programming problem: note that the image of feasible points in $\tilde{B}$ through f is a set of isolated points in $\mathbb{R}^m$.

In our algorithm we derive lower bounds by building linear outer approximations of $\text{conv}(f(\tilde{B}^{g,\mathbb{Z}}))$. As $f(\tilde{B}^{g,\mathbb{Z}}) \subseteq \text{conv}(f(\tilde{B}^{g,\mathbb{Z}}))$, we have that linear outer approximations of the convex hull of $f(\tilde{B}^{g,\mathbb{Z}})$ are valid lower bounds on $\tilde{B}$. Details on how we compute the

hyperplanes to outer approximate $\text{conv}(f(\tilde{B}^{g,\mathbb{Z}}))$ will be given in Section 3.2.

Upper bounds are computed just by evaluating the objective functions at feasible points. As soon as an upper bound z exists such that $L_{\tilde{B}} + \mathbb{R}^m_+ \subseteq \{z\} + \mathbb{R}^m_+ \setminus \{0\}$ we can discard the subbox $\tilde{B}$. Or, in other words, we can avoid to go on partitioning $\tilde{B}$, as we have an evidence that it cannot contain any efficient point for (MOMIC). Our discarding procedure is in fact using a list of upper bounds and it will be detailed in Section 3.1 and Section 3.2.

In Figure 2, the point $z \in f(\tilde{B}^{g,\mathbb{Z}})$ is an upper bound for the nondominated set of the problem, as it is the image of an integer feasible point. All the points that belong to $\mathbb{R}^m \setminus (\{z\} + \mathbb{R}^m_+ \setminus \{0\})$ are candidates to belong to the nondominated set (note that it is not enough to consider $\{z\} - \mathbb{R}^m_+$).

Let $\delta > 0$ be a positive scalar, which is the input parameter of our branch-and-bound method. As the output of our algorithm, we will have a list of subboxes $\tilde{B}$ with $\omega(\tilde{B}) < \delta$, containing the set of efficient points, and a list of upper bounds approximating the nondominated set.

Algorithm 1 is a basic scheme of our branch-and-bound procedure: $\mathcal{L}_{\mathcal{W}}$ denotes the working list and contains boxes that still have to be examined. The list $\mathcal{L}_{\mathcal{S}}$ denotes the list of boxes that fulfill the termination criteria, i.e., those subboxes $\tilde{B}$ that were not discarded and satisfy $\omega(\tilde{B}) < \delta$. In Section 3.3 we will prove that $\mathcal{L}_{\mathcal{S}}$ represents a cover of the efficient set E, namely $E \subseteq \bigcup_{\tilde{B} \in \mathcal{L}_{\mathcal{S}}} \tilde{B}$. The list $\mathcal{L}_{PNS}$ denotes a set of upper bounds and it will be defined in Section 3.1. Note that the flag $\mathcal{D}$ is used in order to decide if a box should be discarded, and it is an output of Algorithm 2. As a final step in Algorithm 1 we filter the list $\mathcal{L}_S$ by a postprocessing phase. Further details will be given in Section 3.2.

3.1 Computation of upper bounds and local upper bounds

In order to compute upper bounds of the nondominated set of (MOMIC), we evaluate the objective functions at integer feasible points $x \in B^{g,\mathbb{Z}}$. It is well known that determining feasible points of a mixed integer set is an NP-hard problem. In the literature, several heuristic methods have been proposed and we cite the Feasibility Pump [15] and some of its enhancements, among them [6, 7, 11, 16]. Within our algorithm, we either detect feasible points by addressing specific single-objective mixed integer convex programming problems (see Section 3.2) or we try to build feasible points simply by rounding the integer components of points $x \in \tilde{B}$, which are generated in our discarding test. Let $\text{round}(x)$ be the point defined as

$$\text{round}(x) = \begin{cases} [x_i] & i \in I \\ x_i & \text{otherwise.} \end{cases}$$

If $\text{round}(x) \in B^{g,\mathbb{Z}}$ holds, $f(\text{round}(x))$ is a valid upper bound.

Upper bounds are needed in order to discard boxes or, in other words, to prune nodes in the branch-and-bound tree. In order to do that we need to introduce two finite sets of points, namely the list of *potentially nondominated solutions* $\mathcal{L}_{PNS} \subseteq f(B^{g,\mathbb{Z}})$ and the list of *local upper bounds* $\mathcal{L}_{LUB} \subseteq \mathbb{R}^m$.

In our algorithm the list of *potentially nondominated solutions* $\mathcal{L}_{PNS}$ is initialized as the empty set. Then, everytime an upper bound $z \in f(B^{g,\mathbb{Z}})$ is computed, we check

Algorithm 1 `MOMIX`: a (MOMIC) Solver

INPUT: (MOMIC), $\delta > 0$

OUTPUT: $\mathcal{L}_{\mathcal{S}}$, $\mathcal{L}_{PNS}$

$\mathcal{L}_{\mathcal{S}} \leftarrow \emptyset$ $\mathcal{L}_{\mathcal{W}} \leftarrow \{B\}$ $\mathcal{L}_{PNS} \leftarrow \emptyset$
while $\mathcal{L}_{\mathcal{W}} \neq \emptyset$ **do**
    Select a box $\tilde{B}$ of $\mathcal{L}_{\mathcal{W}}$ and update $\mathcal{L}_{\mathcal{W}} := \mathcal{L}_{\mathcal{W}} \setminus \tilde{B}$
    Bisect $\tilde{B}$ into subboxes $\tilde{B}^1$ and $\tilde{B}^2$
    for $k = 1, 2$ **do**
        Apply Algorithm 2 to $\tilde{B}^k$ and obtain $\mathcal{D}$ and an updated $\mathcal{L}_{PNS}$
        if $\mathcal{D} =$ `true` **then**
            Discard $\tilde{B}^k$
        else
            if $\omega(\tilde{B}^k) < \delta$ **then**
                Add $\tilde{B}^k$ to $\mathcal{L}_{\mathcal{S}}$
            else
                Add $\tilde{B}^k$ to $\mathcal{L}_{\mathcal{W}}$.
            end if
        end if
    end for
end while
Postprocessing($\mathcal{L}_{\mathcal{S}}, \mathcal{L}_{PNS}$)

whether it is dominated by any point in $\mathcal{L}_{PNS}$. If this is the case, z is not added to $\mathcal{L}_{PNS}$. Otherwise, we update the list by adding z to $\mathcal{L}_{PNS}$ and by removing from $\mathcal{L}_{PNS}$ all the upper bounds dominated by z. By doing this, we ensure that $\mathcal{L}_{PNS}$ is a stable set of points: a set $\mathcal{N} \subseteq \mathbb{R}^m$ is said to be *stable* if there are no $x, y \in \mathcal{N}$ with $x \leq y$ and $x \neq y$.

For the list of local upper bounds $\mathcal{L}_{LUB}$ we need the following definition:

Definition 3.1 *[21] Let $\mathcal{N} \subseteq f(B)$ be a finite and stable set of points and $Z \subseteq \mathbb{R}^m$ be a box such that $f(B) \subseteq \text{int}(Z)$.*

(a) The search region related to $\mathcal{N}$ and Z is defined as

$$S := \{w \in \text{int}(Z) \mid z \not\leq w \text{ for all } z \in \mathcal{N}\}.$$

(b) The search zone for some $p \in \mathbb{R}^m$ related to Z is defined as

$$C(p) = \{w \in \text{int}(Z) \mid w < p\}.$$

(c) A list $\mathcal{L} \subseteq Z$ is called a local upper bound set with respect to $\mathcal{N}$, if

(i) $S = \bigcup_{p \in \mathcal{L}} C(p)$

(ii) $C(p)$ is not a subset of $C(\tilde{p})$ for all $p, \tilde{p} \in \mathcal{L}$.

Let $Z \subseteq \mathbb{R}^m$ be a box such that $f(B) \subseteq \text{int}(Z)$. In our algorithm we initialize the local upper bound set $\mathcal{L}_{LUB}$ with the point $p^0 \in \mathbb{R}^m$ defined as $p^0_j := \max_{w \in Z} w_j$. Then we build and keep updated $\mathcal{L}_{LUB}$ with respect to the finite and stable set $\mathcal{L}_{PNS}$ according to the procedure proposed in [21]. For an illustration of the local upper bound set $\mathcal{L}_{LUB}$ with respect to $\mathcal{L}_{PNS}$, see Figure 3 on page 9.

Remark 3.2 *Along the iterations of our algorithm, let $\mathcal{L}'_{PNS}$ and $\mathcal{L}_{PNS}$ be two consecutive lists of potentially nondominated points, and let $\mathcal{L}'_{LUB}$ and $\mathcal{L}_{LUB}$ be the related local upper bound sets. Then, based on the update procedure mentioned above, we have that to any $z' \in \mathcal{L}'_{PNS}$ there exists $z \in \mathcal{L}_{PNS}$ with either $z' = z$ or with $z \leq z'$. Hence, the search regions S' and S related to $\mathcal{L}'_{PNS}$ and $\mathcal{L}_{PNS}$ are such that $S \subseteq S'$, i.e.,*

$$S = \bigcup_{p \in \mathcal{L}_{LUB}} C(p) \subseteq \bigcup_{p \in \mathcal{L}'_{LUB}} C(p) = S'.$$

Furthermore, for every $p \in \mathcal{L}_{LUB}$ there exists a local upper bound $p' \in \mathcal{L}'_{LUB}$ such that $p \leq p'$. This can be seen by induction considering the updating procedure proposed in [21].

Note that $p \in \mathcal{L}_{LUB}$ is not necessarily the image of a feasible point. Local upper bounds are used in order to decide if a subbox $\tilde{B} \subseteq B$ should be discarded as clarified by the following results.

Lemma 3.3 *[26] Let $\mathcal{L}_{LUB}$ be a local upper bound set with respect to the finite and stable set $\mathcal{L}_{PNS} \subseteq f(B^{g,\mathbb{Z}})$. For every $z \in \mathcal{L}_{PNS}$ and for every $j \in \{1, ..., m\}$ there is a $p \in \mathcal{L}_{LUB}$ with $z_j = p_j$ and $z_r < p_r$ for all $r \in \{1, ..., m\} \setminus \{j\}$.*

Based on this lemma we can prove our main result for the pruning of nodes:

Theorem 3.4 *Consider a subbox $\tilde{B} \subseteq B$. Let $\mathcal{L}_{PNS} \subseteq f(B^{g,\mathbb{Z}})$ be a finite and stable set. Let $\mathcal{L}_{LUB}$ be the local upper bound set w.r.t. $\mathcal{L}_{PNS}$. If*

$$p \notin f(\tilde{B}^{g,\mathbb{Z}}) + \mathbb{R}^m_+ \quad \text{holds for all } p \in \mathcal{L}_{LUB}, \tag{2}$$

$\tilde{B}$ does not contain any efficient point for (MOMIC).

Proof. Assume by contradiction that an efficient point $x^* \in \tilde{B}^{g,\mathbb{Z}}$ for (MOMIC) exists. Therefore, from (2), we have

$$f(x^*) \not\leq p \text{ for all } p \in \mathcal{L}_{LUB}. \tag{3}$$

Since $\mathcal{L}_{LUB}$ is a local upper bound set w.r.t. $\mathcal{L}_{PNS}$, it follows from (3) and Definition 3.1 (b) and (c), that $f(x^*)$ does not belong to the search region S. Hence, there exists a point $z \in \mathcal{L}_{PNS}$ with $z \leq f(x^*)$. As $z \in \mathcal{L}_{PNS}$, a point $x' \in B^{g,\mathbb{Z}}$ exists such that $z = f(x')$. Since x^* is efficient for (MOMIC), it follows $z = f(x') = f(x^*)$. Lemma 3.3 implies that there is a point $p' \in \mathcal{L}_{LUB}$ with $f(x^*) \leq p'$, which is a contradiction to (3) and the theorem is proved. □

From Theorem 3.4, since $\tilde{B}^{g,\mathbb{Z}} \subseteq \tilde{B}^g$ and $f(\tilde{B}^{g,\mathbb{Z}}) \subseteq \text{conv}(f(\tilde{B}^{g,\mathbb{Z}}))$ hold, we obtain the following corollary.

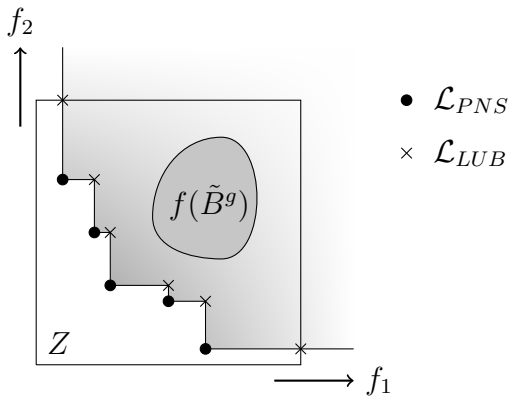


Figure 3: Illustration of Corollary 3.5 for $m = 2$. In the picture we plot the local upper bound set $\mathcal{L}_{LUB}$ with respect to the set of potentially nondominated solutions $\mathcal{L}_{PNS}$. Note that the box $\tilde{B}$ would be discarded, as the assumptions of Corollary 3.5 (a) are satisfied and $\tilde{B}$ cannot contain any efficient point for (MOMIC).

Corollary 3.5 *Let $\tilde{B}$ be a subbox of B. Let $\mathcal{L}_{PNS} \subseteq f(B^{g,\mathbb{Z}})$ be a finite and stable set and let $\mathcal{L}_{LUB}$ be the local upper bound set w.r.t. $\mathcal{L}_{PNS}$.*

(a) If

$$p \notin f(\tilde{B}^g) + \mathbb{R}^m_+ \quad \text{holds for all } p \in \mathcal{L}_{LUB},$$

$\tilde{B}$ does not contain any efficient point for (MOMIC).

(b) If

$$p \notin \operatorname{conv}(f(\tilde{B}^{g,\mathbb{Z}})) + \mathbb{R}^m_+ \quad \text{holds for all } p \in \mathcal{L}_{LUB},$$

$\tilde{B}$ does not contain any efficient point for (MOMIC).

An illustration of Corollary 3.5 (a) can be found in Figure 3.

The following remark clarifies how the assumptions of Corollary 3.5 are related. Furthermore, it gives the basis of the hierarchy of lower bounds in our bounding procedure.

Remark 3.6 *Note that due to the convexity of the objective functions f_j, $j = 1, \dots, m$ and of $\tilde{B}^g$ the following holds*

$$\operatorname{conv}(f(\tilde{B}^{g,\mathbb{Z}})) + \mathbb{R}^m_+ \subseteq f(\tilde{B}^g) + \mathbb{R}^m_+.$$

3.2 Determining lower bounds and pruning nodes

The theoretical results introduced in the previous section, namely Corollary 3.5, give the basis of the discarding procedure in our branch-and-bound algorithm: For every subbox $\tilde{B}$ we want to check whether $p \notin L_{\tilde{B}} + \mathbb{R}^m_+$ holds for all $p \in \mathcal{L}_{LUB}$, being $L_{\tilde{B}}$ a valid lower bound for $\tilde{B}^{g,\mathbb{Z}}$.

As $f(\tilde{B}^g)$ is a valid lower bound, a straightforward way to verify if a box should be discarded would be to check whether a local upper bound $p \in \mathcal{L}_{LUB}$ belongs to this lower

bound given by the convex relaxation. This would mean to check whether $p \in f(\tilde{B}^g)+\mathbb{R}^m_+$. This can be done by addressing a simple single-objective continuous convex problem.

From a computational point of view, this means that we would need to solve $|\mathcal{L}_{LUB}|$ single-objective continuous convex problems, at every node of the branching tree.

In our algorithm, in order to reduce this numerical effort, we check instead whether a local upper bound belongs to a linear outer approximation of $f(\tilde{B}^g)+\mathbb{R}^m_+$, i.e., we only need to check whether a local upper bound satisfies linear inequalities. Furthermore, our linear outer approximations are built in a smart way: the supporting hyperplanes computation is adaptively driven by some "meaningful" local upper bounds $p \in \mathcal{L}_{LUB}$.

Additionally, in case we want to improve our lower bound, we compute further hyperplanes to outer approximate $\text{conv}(f(\tilde{B}^{g,\mathbb{Z}})) + \mathbb{R}^m_+$. Again this computation is done in an adaptive way, and the supporting hyperplanes computation is steered by some specific local upper bounds $p \in \mathcal{L}_{LUB}$.

In the following, we give details on how the supporting hyperplanes are computed and how the discarding procedure works.

At an arbitrary node of our branching tree we select a subbox $\tilde{B} \subseteq B$. In order to compute valid lower bounds on $\tilde{B}$ we build linear outer approximations $L_{\tilde{B}}$ of $\text{conv}(f(\tilde{B}^{g,\mathbb{Z}}))$, so that

$$f(\tilde{B}^{g,\mathbb{Z}}) \subseteq \text{conv}(f(\tilde{B}^{g,\mathbb{Z}})) \subseteq L_{\tilde{B}} + \mathbb{R}^m_+.$$

In order to discard the subbox $\tilde{B}$ and prune the current node we check whether

$$p \notin L_{\tilde{B}} + \mathbb{R}^m \quad \text{holds for all } p \in \mathcal{L}_{LUB}.$$

Then, from Corollary 3.5 $\tilde{B}$ does not contain any efficient point for (MOMIC) and the current node can be pruned. As we will deal with linear outer approximations of sets, we recall here the definition of supporting hyperplane of a set:

Definition 3.7 *Let $P \subset \mathbb{R}^m$ be a nonempty set, let $\lambda \in \mathbb{R}^m \setminus \{0\}$ and $z \in \partial P$, where ∂P is the boundary of the set P. The hyperplane*

$$H^{\lambda,z} := \{y \in \mathbb{R}^m \mid \lambda^T y = \lambda^T z\}$$

is called supporting hyperplane (of P), if $\lambda^T y \geq \lambda^T z$ holds for all $y \in P$.

As mentioned in the introduction, we propose two versions of our branch-and-bound algorithm. The difference lies in the lower bounds computation. The first version of our algorithm, named $\texttt{MOMIX}_{light}$, computes valid lower bounds by addressing only single-objective continuous convex optimization problems. The second version, named `MOMIX`, tries to define stronger lower bounds by dealing also with single-objective mixed integer convex programming problems. In our algorithm we use a flag *light* to distinguish between the two versions of the method.

Both, $\texttt{MOMIX}_{light}$ and `MOMIX`, start by computing linear outer approximations of the convex set $f(\tilde{B}^g)+\mathbb{R}^m_+$ by solving a family of single-objective continuous convex optimization problems. As $\text{conv}(f(\tilde{B}^{g,\mathbb{Z}})) + \mathbb{R}^m_+ \subseteq f(\tilde{B}^g) + \mathbb{R}^m_+$ holds by Remark 3.6, we have that linear outer approximations of $f(\tilde{B}^g) + \mathbb{R}^m_+$ are valid lower bounds for $\text{conv}(f(\tilde{B}^{g,\mathbb{Z}}))$ as well (see Figure 4 on page 14). If the linear outer approximation of $f(\tilde{B}^g) + \mathbb{R}^m_+$ does

not allow to discard the box $\tilde{B}$, `MOMIX` tries to improve it by addressing properly defined single-objective mixed integer convex programming problems.

As a first step for the outer approximation, we compute the *ideal point* $f^{id} \in \mathbb{R}^m$ of $f(\tilde{B}^g)$, namely the point whose j-th component is the minimum of f_j on $\tilde{B}^g$:

$$f_j^{id} := \min_{x \in \tilde{B}^g} f_j(x) \quad j = 1, \dots, m. \tag{4}$$

We denote by $x^{j,id} \in \tilde{B}^g$ a minimal solution in (4). Let e^j be the j-th unit vector, then $H^{e^j, f^{id}}$ is a supporting hyperplane of $f(\tilde{B}^g)$. As a first linear outer approximation of $f(\tilde{B}^g)$ or, in other words, as a first lower bound for $f(\tilde{B}^{g,\mathbb{Z}})$ we consider

$$L_{\tilde{B}} := \partial\left(\bigcap_{j \in \{1,\dots,m\}} (H^{e^j, f^{id}} + \mathbb{R}^m_+) \right) = \{f^{id}\} + \partial(\mathbb{R}^m_+). \tag{5}$$

Note that building $L_{\tilde{B}}$ requires to solve m single-objective continuous convex optimization problems for the computation of f^{id}.

Once $L_{\tilde{B}}$ is computed we enter in a loop. For every $p \in \mathcal{L}_{LUB}$ we check whether $p \in L_{\tilde{B}} + \mathbb{R}^m_+$ holds. If this is the case, we try to improve the current linear outer approximation $L_{\tilde{B}}$ by computing a further hyperplane, based on $p \in \mathcal{L}_{LUB}$. This is done by addressing the following single-objective continuous convex programming problem (see also [13, 22])

$$\begin{aligned} \min\ & t \\ \text{s.t. } & f(x) \leq p + te \\ & x \in \tilde{B}^g \\ & t \in \mathbb{R}, \end{aligned} \tag{$\mathrm{P}_p(\tilde{B}^g)$}$$

where $e = (1, \dots, 1)^T \in \mathbb{R}^m$.

Note that Problem ($\mathrm{P}_p(\tilde{B}^g)$) needs to be addressed *only* in case of $p \in L_{\tilde{B}} + \mathbb{R}^m_+$. In other words, in our lower bound computation, we do not necessarily address Problem ($\mathrm{P}_p(\tilde{B}^g)$) for all $p \in \mathcal{L}_{LUB}$, as it would be the case if we would rely only on the convex relaxation $f(\tilde{B}^g)$.

Under regularity assumptions, we have that any minimal solution $(\hat{x}, \hat{t}) \in \tilde{B}^g \times \mathbb{R}$ of Problem ($\mathrm{P}_p(\tilde{B}^g)$) admits Lagrange multipliers. We refer to [13, 26] in case no Lagrange multiplier exists. Let $(\hat{x}, \hat{t}) \in \tilde{B}^g \times \mathbb{R}$ be a minimal solution of ($\mathrm{P}_p(\tilde{B}^g)$) and let $\hat{\lambda} \in \mathbb{R}^m_+$ be a Lagrange multiplier for the constraint $f(x) \leq p + te$. Then, the hyperplane $H^{\hat{\lambda}, \hat{y}(p)}$ with $\hat{y}(p) := p + \hat{t}e$ is a supporting hyperplane of $f(\tilde{B}^g)$, cf. [22, 27, 26].

There exist two possibilities:

(i) If $\hat{t} > 0$ holds, then $p \notin f(\tilde{B}^g) + \mathbb{R}^m_+$, we improve the outer approximation by $H^{\hat{\lambda}, \hat{y}(p)}$, and consider the next local upper bound;

(ii) if $\hat{t} \leq 0$ holds, then $p \in f(\tilde{B}^g) + \mathbb{R}^m_+$ and the assumption of Corollary 3.5 (a) is not satisfied.

If case (ii) occurs, so far we cannot discard $\tilde{B}$ based on Corollary 3.5 (a) as it may contain efficient points for (MOMIC). Then, in case we apply `MOMIX`, i.e., $light = 0$, we try to apply Corollary 3.5 (b) and thus we try to improve our linear outer approximation by addressing a single-objective mixed integer convex programming problem. Let $\hat{\lambda} \in \mathbb{R}^m$ be a Lagrange multiplier for the constraint $f(x) \leq p + te$ for the solution of ($\mathrm{P}_p(\tilde{B}^g)$). We define the following problem

$$\begin{aligned} &\min \ \hat{\lambda}^T f(x) \\ &\text{s.t. } \ x \in \tilde{B}^{g,\mathbb{Z}}. \end{aligned} \tag{$\mathrm{MICP}_p(\hat{\lambda}, \tilde{B})$}$$

Let $\hat{x} \in \tilde{B}^{g,\mathbb{Z}}$ be a minimal solution of ($\mathrm{MICP}_p(\hat{\lambda}, \tilde{B})$). Then the hyperplane $H^{\hat{\lambda}, f(\hat{x})}$ is a supporting hyperplane of $\mathrm{conv}(f(\tilde{B}^{g,\mathbb{Z}}))$ and it holds $\mathrm{conv}(f(\tilde{B}^{g,\mathbb{Z}})) + \mathbb{R}^m_+ \subseteq H^{\hat{\lambda}, f(\hat{x})} + \mathbb{R}^m_+$. Furthermore, $f(\hat{x})$ is a valid upper bound for (MOMIC). Note that in case we are at a node where all integer variables are fixed we do not need to perform this step.

Again two situations occur:

(i) If $\hat{\lambda}^T p < \hat{\lambda}^T f(\hat{x})$ holds, we improve the outer approximation by $H^{\hat{\lambda}, f(\hat{x})}$ and consider the next local upper bound

(ii) If $\hat{\lambda}^T p \geq \hat{\lambda}^T f(\hat{x})$ holds, the local upper bound p lies above the hyperplane $H^{\hat{\lambda}, f(\hat{x})}$.

If we are in case (ii), we do not go on improving our linear outer approximation and we branch the current node by bisecting $\tilde{B}$ in a later iteration.

Algorithm 2 is reporting our lower bound computation in details.

As soon as feasible points of $\tilde{B}^{g,\mathbb{Z}}$ are found, both $\mathcal{L}_{PNS}$ and $\mathcal{L}_{LUB}$ are updated. This is the reason why in Algorithm 2 we make use of the list $\mathcal{L}^*_{LUB}$ which does not change along the discarding test: We need a fixed set of local upper bounds in order to ensure the termination of the loop starting in line 10.

Note that in line 17 we update the linear outer approximation even if the subbox $\tilde{B}$ is further kept either in the working list $\mathcal{L}_{\mathcal{W}}$ or in the solution list $\mathcal{L}_{\mathcal{S}}$. This is done in order to perform the postprocessing phase in Algorithm 1: Let $\tilde{B} \in \mathcal{L}_{\mathcal{S}}$ and let $\mathcal{H}$ be the linear outer approximation of $f(\tilde{B}^{g,\mathbb{Z}})$ built by Algorithm 2. This subbox $\tilde{B}$ is removed from $\mathcal{L}_S$ if for all local upper bounds p belonging to the final list $\mathcal{L}_{LUB}$ we have that a hyperplane $H^{\lambda, z'} \in \mathcal{H}$ exists such that $\lambda^T p \geq \lambda^T z'$ holds.

Example 3.8 *In Figure 4 on page 14 we illustrate our lower bounding procedure on a biobjective purely integer convex programming instance. Note that in this case the image of integer feasible points is a set of isolated points in $\mathbb{R}^2$. The first outer approximation considered is based on the ideal point f^{id}. Then, considering the local upper bound $p \in \mathcal{L}_{LUB}$, the supporting hyperplane $H^{\hat{\lambda}, \hat{y}(p)}$ for $f(\tilde{B}^g)$ is built by solving Problem ($\mathrm{P}_p(\tilde{B}^g)$) and added to the linear outer approximation. Finally, in case `MOMIX` (and not `MOMIX`$_{light}$) is applied, the linear outer approximation is further refined by considering $H^{\hat{\lambda}, f(\hat{x})}$, being $\hat{x}$ a solution of ($\mathrm{MICP}_p(\hat{\lambda}, \tilde{B})$).*

In the following lemma, we prove the exactness of our lower bounding procedure: we show that Algorithm 2 returns $\mathcal{D} =$ `false` in case $\tilde{B}$ contains an efficient point for (MOMIC). Thus it will be further partitioned.

Algorithm 2 Lower bounding procedure

INPUT: (MOMIC), a subbox $\tilde{B} \subseteq B$, $\mathcal{L}_{PNS}$, $\mathcal{L}_{LUB}$, $light \in \{0,1\}$
OUTPUT: $\mathcal{L}_{PNS}$, $\mathcal{L}_{LUB}$, $\mathcal{D}$, where $\mathcal{D} = \texttt{true}$ means "Discard $\tilde{B}$"

Set $\mathcal{D} \leftarrow \texttt{true}$
for $j \in \{1, \ldots, m\}$ **do**
  Compute f_j^{id} and obtain $x^{j,id} \in \tilde{B}^g$
  if round$(x^{j,id}) \in B^{g,\mathbb{Z}}$ **then**
    Update $\mathcal{L}_{PNS}$ by $f(\text{round}(x^{j,id}))$ and update $\mathcal{L}_{LUB}$
  end if
end for
Set $\mathcal{L}^*_{LUB} \leftarrow \mathcal{L}_{LUB}$
Set $\mathcal{H} \leftarrow \{H^{e^j, f^{id}} \mid j \in \{1, ..., m\}\}$
for $p \in \mathcal{L}^*_{LUB}$ **do**
  if $\lambda^T p \geq \lambda^T z'$ for all $H^{\lambda, z'} \in \mathcal{H}$ **then**
    Solve ($\text{P}_p(\tilde{B}^g)$) and get $(x^*, t^*) \in \tilde{B}^g \times \mathbb{R}$, $\hat{\lambda} \in \mathbb{R}^m$ Lagrange multiplier for the constraint $f(x) \leq p + te$
    if round$(x^*) \in B^{g,\mathbb{Z}}$ **then**
      Update $\mathcal{L}_{PNS}$ by $f(\text{round}(x^*))$ and update $\mathcal{L}_{LUB}$
    end if
    if $t^* \leq 0$ and $light = 1$ **then**
      Set $\mathcal{H} \leftarrow \mathcal{H} \cup \{H^{\hat{\lambda}, p + t^* e}\}$
      Set $\mathcal{D} \leftarrow \texttt{false}$ and **break for-loop**
    else if $t^* \leq 0$ and $light = 0$ **then**
      Solve Problem ($\text{MICP}_p(\hat{\lambda}, \tilde{B})$)
      if ($\text{MICP}_p(\hat{\lambda}, \tilde{B})$) is infeasible **then**
        Set $\mathcal{D} \leftarrow \texttt{true}$ and **break for-loop**
      else
        Let $\hat{x} \in \tilde{B}^{g,\mathbb{Z}}$ be a solution of Problem ($\text{MICP}_p(\hat{\lambda}, \tilde{B})$)
        Update $\mathcal{L}_{PNS}$ by $f(\hat{x})$ and update $\mathcal{L}_{LUB}$
        Set $\mathcal{H} \leftarrow \mathcal{H} \cup \{H^{\hat{\lambda}, f(\hat{x})}\}$
      end if
      if $\hat{\lambda}^T p \geq \hat{\lambda}^T f(\hat{x})$ **then**
        Set $\mathcal{D} \leftarrow \texttt{false}$ and **break for-loop**
      end if
    else
      Set $\mathcal{H} \leftarrow \mathcal{H} \cup \{H^{\hat{\lambda}, p + t^* e}\}$.
    end if
  end if
end for

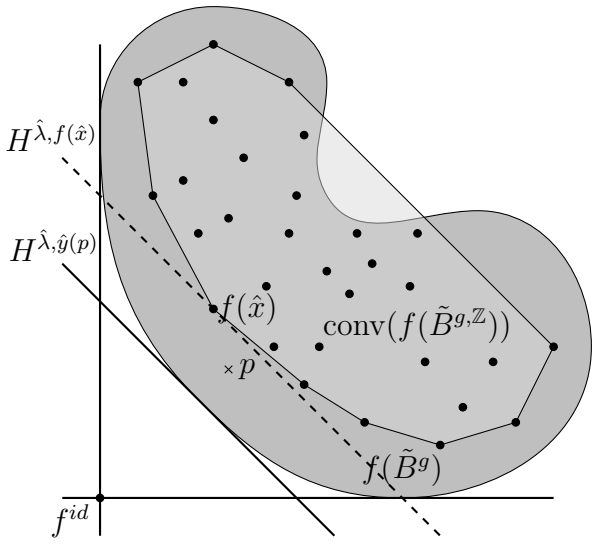


Figure 4: Illustration of our lower bounding procedure on a biobjective purely integer convex programming instance.

Lemma 3.9 *Let $\tilde{B}$ be a subbox of B that contains an efficient point $x^* \in \tilde{B}^{g,\mathbb{Z}}$ of (MOMIC). Then Algorithm 2 returns $\mathcal{D} =$ `false`.*

Proof. We distinguish two cases for which Algorithm 2 returns $\mathcal{D} =$ `true`: either $\mathcal{D} =$ `true` because (MICP$_p(\hat{\lambda}, \tilde{B})$) is infeasible for any $p \in \mathcal{L}^*_{LUB}$ or because lines 18 or 29 are never reached for any $p \in \mathcal{L}^*_{LUB}$. The first case cannot occur as $x^* \in \tilde{B}^{g,\mathbb{Z}}$. The second case may occur if for all $p \in \mathcal{L}^*_{LUB}$ either the condition in line 11 or the condition in line 16 or the condition in line 28 is not satisfied. In all three cases we get that $p \notin f(\tilde{B}^g) + \mathbb{R}^m_+$ holds for all $p \in \mathcal{L}^*_{LUB}$. Corollary 3.5 then implies that $\tilde{B}$ does not contain any efficient point for (MOMIC). □

3.3 Correctness of `MOMIX`

We already mentioned that our algorithm stops as soon as the working list $\mathcal{L}_\mathcal{W}$ is empty and we get a list of subboxes $\tilde{B}$ of width less than a prescribed value $\delta > 0$, i.e., $\omega(\tilde{B}) < \delta$. In this section we first prove the exacteness of Algorithm 1, namely we prove that it returns the set $\mathcal{L}_\mathcal{S}$ which is a cover of the efficient set E of (MOMIC). In order to do that, we need to make the following assumption related to the branching rule adopted.

Assumption 3.10 *Let $\tilde{B} \subseteq B$. Let the branching rule in Algorithm 1 be such that for the subboxes $\tilde{B}^1$ and $\tilde{B}^2$ derived from $\tilde{B}$ it holds*

$$\tilde{B}^{g,\mathbb{Z}} \subseteq \tilde{B}^1 \cup \tilde{B}^2$$

and that the algorithm performs a finite number of branching steps before stopping.

Note that Assumption 3.10 implies that the set of efficient points for (MOMIC) belonging to $\tilde{B}$ is a subset of $\tilde{B}^1 \cup \tilde{B}^2$. In Section 4 we propose and compare two branching rules which both satisfy Assumption 3.10.

From Assumption 3.10 and Lemma 3.9 we directly get the following

Theorem 3.11 *Let E be the efficient set of (MOMIC). Let $\mathcal{L}_{\mathcal{S}}$ be the output of Algorithm 1. Then $\mathcal{L}_{\mathcal{S}}$ is a cover of E, namely $E \subseteq \bigcup_{\tilde{B}\in\mathcal{L}_{\mathcal{S}}} \tilde{B}$.*

We want to underline that when `MOMIX` (and not $\texttt{MOMIX}_{light}$) is applied, the list $\mathcal{L}_{\mathcal{S}}$ is built in a way such that every subbox $\tilde{B}$ belonging to $\mathcal{L}_{\mathcal{S}}$ admits at least one feasible point, i.e., $\tilde{B}^{g,\mathbb{Z}} \neq \emptyset$. Note that a feasible point $\hat{x} \in \tilde{B}^{g,\mathbb{Z}}$ is computed in line 24 in Algorithm 2.

We further prove that the points in the final list of potentially nondominated solutions $\mathcal{L}_{PNS}$ are images of some points in the cover of the efficient set of (MOMIC).

Proposition 3.12 *Let $\mathcal{L}_{PNS}$ and $\mathcal{L}_{\mathcal{S}}$ be the output of Algorithm 1. Then, for every $z \in \mathcal{L}_{PNS}$ there exists a subbox $\tilde{B} \in \mathcal{L}_{\mathcal{S}}$ such that $z \in f(\tilde{B}^{g,\mathbb{Z}})$.*

Proof. Assume by contradiction that the preimage x of $z \in \mathcal{L}_{PNS}$ belongs to a discarded subbox $\tilde{B}^{g,\mathbb{Z}}$. Then, at a certain node of our branching tree, a lower bound $L_{\tilde{B}}$ was computed such that for all $p \in \mathcal{L}'_{LUB}$ we have

$$p \notin L_{\tilde{B}} + \mathbb{R}^m_+,$$

where $\mathcal{L}'_{LUB}$ is the list of local upper bounds at that node. Hence,

$$p \notin f(\tilde{B}^{g,\mathbb{Z}}) + \mathbb{R}^m_+ \text{ for all } p \in \mathcal{L}'_{LUB}. \tag{6}$$

Let $\mathcal{L}_{LUB}$ be the final list of local upper bounds related to $\mathcal{L}_{PNS}$. By Lemma 3.3 we have that a local upper bound $\hat{p} \in \mathcal{L}_{LUB}$ exists such that $z \leq \hat{p}$, i.e., $\hat{p} \in f(\tilde{B}^{g,\mathbb{Z}}) + \mathbb{R}^m_+$. If $\hat{p} \in \mathcal{L}'_{LUB}$ we directly get a contradiction to (6). Otherwise, from Remark 3.2, we have that a local upper bound $\bar{p} \in \mathcal{L}'_{LUB}$ exists such that $\hat{p} \leq \bar{p}$. Hence, $\bar{p} \in f(\tilde{B}^{g,\mathbb{Z}}) + \mathbb{R}^m_+$ which contradicts (6). □

We now show that, in case `MOMIX` is applied, $\mathcal{L}_{PNS}$ is a "good" approximation of the nondominated frontier, in the sense that the distance of the image of efficient points from $\mathcal{L}_{PNS}$ is bounded by a quantity that depends on $\delta > 0$, which is the input parameter of Algorithm 1. For this we exploit the Lipschitz continuity of the objective functions f_j, $j = 1, \ldots, m$, which holds as the functions are continuously differentiable and the feasible sets are compact. Let $L_j \geq 0$ be the Lipschitz constant for function f_j, $j = 1, \ldots, m$.

Theorem 3.13 *Let $\delta > 0$ be the input parameter and $\mathcal{L}_{PNS}$, $\mathcal{L}_{\mathcal{S}}$ be the output of Algorithm 1 where `MOMIX` is applied, i.e., light $= 0$. Let $\mathcal{L}_{LUB}$ be the local upper bound set with respect to $\mathcal{L}_{PNS}$ and $E \subseteq B^{g,\mathbb{Z}}$ be the efficient set of (MOMIC). Set $L = \max_{j=1,\ldots,m} L_j$. Then*

$$f(E) \subseteq \left(\bigcup_{p\in\mathcal{L}_{LUB}} (\{p\} - \mathbb{R}^m_+) \right) \bigcap \left(\bigcup_{z\in\mathcal{L}_{PNS}} (\{z - L\delta e\} + \mathbb{R}^m_+) \right)$$

holds, where $e = (1, \ldots, 1)^T \in \mathbb{R}^m$.

Proof. Let $x \in E$. In order to prove that $f(x) \in \bigcup_{p\in\mathcal{L}_{LUB}} (\{p\} - \mathbb{R}^m_+)$ we distinguish two cases. Assume first that $f(x)$ belongs to the search region S related to $\mathcal{L}_{PNS}$ (see Definition 3.1). Then, a local upper bound $p \in \mathcal{L}_{LUB}$ exists such that $f(x)$ belongs to the search zone $C(p)$ related to p. It follows that $f(x) < p$. On the other hand, if $f(x) \notin S$,

by Definition 3.1 a point $z \in \mathcal{L}_{PNS}$ exists such that $z \leq f(x)$. Since $f(x)$ is nondominated and $z \in \mathcal{L}_{PNS}$ is the image of a feasible point, we necessarily have $f(x) = z$. From Lemma 3.3, a $p \in \mathcal{L}_{LUB}$ exists such that $f(x) = z \leq p$.

We now prove that $f(x) \in \bigcup_{z \in \mathcal{L}_{PNS}} (\{z - L\delta e\} + \mathbb{R}^m_+)$. Let $\tilde{B} \in \mathcal{L}_{\mathcal{S}}$ so that $x \in \tilde{B}^{g,\mathbb{Z}}$ which exists by Theorem 3.11. From Algorithm 2, if $light = 0$, a feasible point $\hat{x} \in \tilde{B}^{g,\mathbb{Z}}$ is computed for $\tilde{B}$ (see line 24). The point $f(\hat{x})$ is an upper bound for (MOMIC) and then a candidate to belong to $\mathcal{L}_{PNS}$. Then, either $f(\hat{x})$ is an element of $\mathcal{L}_{PNS}$ or $z \in \mathcal{L}_{PNS}$ exists such that $z \leq f(\hat{x})$. Since $\omega(\tilde{B}) < \delta$ holds, we have $\|x - \hat{x}\| < \delta$ and, by Lipschitz continuity of f_j we obtain $|f_j(x) - f_j(\hat{x})| \leq L_j\delta \leq L\delta,\ j = 1, \ldots, m$. Therefore, since $L\delta \geq 0$, we have that $f_j(x) \geq f_j(\hat{x}) - L\delta \geq z_j - L\delta$ for all $j = 1, \ldots, m$ and the theorem is proved. □

An illustration of Theorem 3.13 on an instance of (MOMIC) is given in Figure 11 in Section 4.

4 Numerical Results

In this section, we present our numerical experience on different instances of (MOMIC). Next to some results on biobjective quadratic instances, we show results on an instance with $m = 3$ and results on a mixed integer convex non-quadratic instance.

In our implementation of Algorithm 1, at line 3, in order to select a subbox $\tilde{B} \in \mathcal{L}_{\mathcal{W}}$, we consider the ideal point f^{id} computed according to (4). We pick at first those subboxes with the lexicographic smallest ideal point f^{id}, with the idea that boxes with small f^{id} may lead to good upper bounds. Concerning the branching rule, we adopted two different strategies detailed in Section 4.1.

For solving the single-objective convex problems used to compute f^{id} and to define the hyperplanes $H^{\hat{\lambda},p+\hat{t}e}$ we applied `fmincon`, the solver from the optimization toolbox of MATLAB. For all runs we set $\delta = 0.1$ if it is not stated otherwise.

For the solution of the mixed integer convex programming problem used to define the hyperplane $H^{\hat{\lambda},f(\hat{x})}$ that enrich the linear outer approximation of $f(\tilde{B}^{g,\mathbb{Z}})$ (line 20 of Algorithm 2) we can adopt any solver which is able to deal with convex MINLPs as e.g. SCIP [17]. In our numerical experience we mainly used quadratic instances and within our implementation of `MOMIX` we adopted the mixed integer quadratic solver of GUROBI [19].

Both versions of Algorithm 1, `MOMIX` and $\texttt{MOMIX}_{light}$ have been implemented in MATLAB R2018a. All experiments have been performed on a computer with Intel(R) Core(TM) i5-7400T CPU and 16 Gbytes RAM on operating system Windows 10 Enterprise.

4.1 Branching rules

In our numerical experiments we make use of two different branching rules. Both rules are based on the idea of partitioning boxes considering first the largest edges, giving priority to the integer variables in two different ways.

Let $\tilde{B} = [\tilde{l}, \tilde{u}]$ be a subbox of B. We consider the following two sets of indices in order to identify the branching variable $\hat{\imath} \in \{1, \ldots, n\}$:

(br1) $J_1 = \operatorname{argmax}\{\tilde{u}_i - \tilde{l}_i \mid i \in I\}$. If $\tilde{u}_i - \tilde{l}_i = 0$ for all $i \in I$, i.e., in case all the integer variables are fixed, define $J_1 = \operatorname{argmax}\{\tilde{u}_i - \tilde{l}_i \mid i \in \{1, \dots, n\} \setminus I\}$. Choose $\hat{\imath} \in J_1$.

(br2) $J_2 = \operatorname{argmax}\{\tilde{u}_i - \tilde{l}_i \mid i \in \{1, ..., n\}\}$. If $J_2 \cap I \neq \emptyset$ holds, choose $\hat{\imath} \in J_2 \cap I$.

The first strategy is standard in mixed integer procedures: the integer variables are fixed at first. The second strategy aims to reduce the largest edge of the boxes, no matter whether it is related to an integer variable or not. Only if there is more than one largest edges and one of them belongs to an integer variable, we prefer to branch at this variable. We will show that this second non-standard branching rule performs better on some of the test instances.

Once the branching variable $\hat{\imath} \in \{1, \dots, n\}$ has been selected, we partition the box $\tilde{B}$ into two boxes $\tilde{B}_1$, $\tilde{B}_2$ as follows: We set for $c_1, c_2 \in [\tilde{l}_{\hat{\imath}}, \tilde{u}_{\hat{\imath}}]$:

$$\tilde{B}_1 := \left[\tilde{l}, (\tilde{u}_1, ..., \tilde{u}_{\hat{\imath}-1}, c_1, \tilde{u}_{\hat{\imath}+1}, ..., \tilde{u}_n)^T\right] \text{ and } \tilde{B}_2 := \left[\left(\tilde{l}_1, ..., \tilde{l}_{\hat{\imath}-1}, c_2, \tilde{l}_{\hat{\imath}+1}, ..., \tilde{l}_n\right)^T, \tilde{u}\right].$$

Thereby, we differentiate between $\hat{\imath} \in I$ and $\hat{\imath} \notin I$. If $\hat{\imath} \notin I$, we set $c_1 = c_2 = (\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}})/2$. If $\hat{\imath} \in I$, we set $c_1 = \lfloor (\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}})/2 \rfloor$ and $c_2 = \lceil (\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}})/2 \rceil$. In case $\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}}$ is an even number, in order to avoid $\tilde{B}_1 \cap \tilde{B}_2 \neq \emptyset$, we split considering $c_1 = (\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}})/2$ and $c_2 = 1 + (\tilde{l}_{\hat{\imath}} + \tilde{u}_{\hat{\imath}})/2$. Note that such a bisection excludes the infeasible part between c_1 and c_2.

As already mentioned in the introduction, we assume $B = [l, u] \subset \mathbb{R}^n$ with $l_i, u_i \in \mathbb{Z}$ for all $i \in I$. Then, for all subboxes $\tilde{B}$ obtained by any of the branching rules presented, it holds $\tilde{l}_i, \tilde{u}_i \in \mathbb{Z}$ for all $i \in I$. Furthermore, it is easy to see that both branching rules adopted in `MOMIX` and $\texttt{MOMIX}_{light}$ satisfy Assumption 3.10.

In order to clarify the differences between the two rules (br1) and (br2), we present the results obtained by `MOMIX` when applied to the following:

Test instance 4.1 *We study the biobjective mixed integer instance with two variables:*

$$\begin{aligned} \min \quad & \begin{pmatrix} x_1 + x_2 \\ x_1^2 + x_2^2 \end{pmatrix} \\ \text{s.t.} \quad & (x_1 - 2)^2 + (x_2 - 2)^2 \leq 36 \\ & x_1 \in [-2, 2] \\ & x_2 \in [-4, 4] \cap \mathbb{Z}. \end{aligned} \tag{T1}$$

In Figure 6 and Figure 8, we show in gray the image of $B^{g,\mathbb{Z}}$ under the objective functions. In black we give the set $\mathcal{L}_{PNS}$ obtained by applying `MOMIX` with (br1) and (br2), respectively. Note that `MOMIX` is able to find in both cases a good approximation of the non-connected nondominated set of the instance.

In Figure 5 and Figure 7, we report the partition of the box $B = [(-2, -4)^T, (2, 4)^T]$ obtained applying `MOMIX` with (br1) and (br2), respectively.

Both branching rules explore the whole feasible set of (T1). Even while they partition the box B in different ways, the outputs of `MOMIX` are very similar, i.e., with (br1) and (br2) the boxes in the solution list $\mathcal{L}_S$ and the list of upper bounds $\mathcal{L}_{PNS}$ are nearly the same.

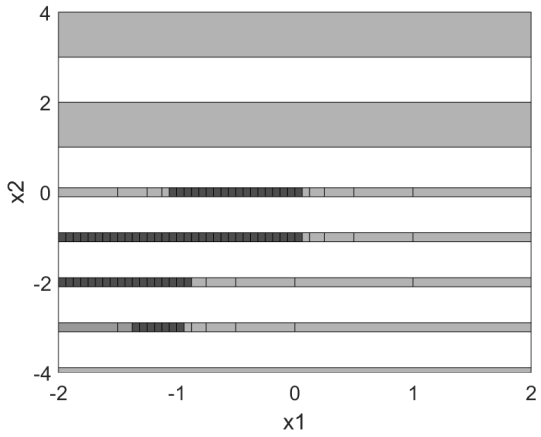


Figure 5: Partition of the box B obtained by applying `MOMIX` with (br1)

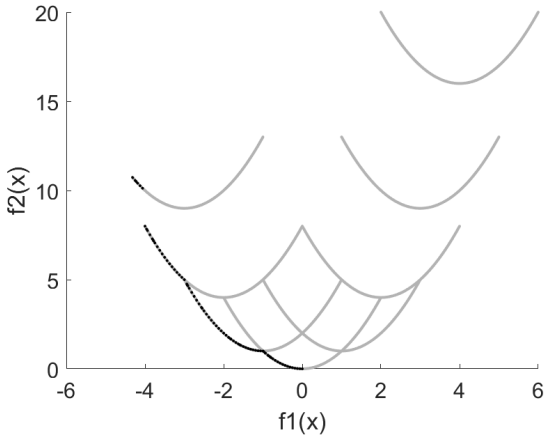


Figure 6: The set $\mathcal{L}_{PNS}$ obtained by applying `MOMIX` with (br1)

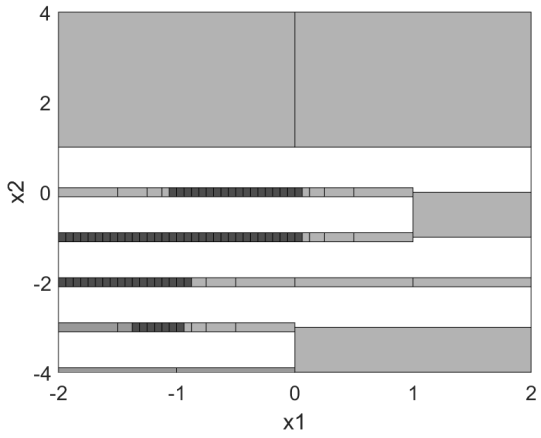


Figure 7: Partition of the box B obtained by applying `MOMIX` with (br2)

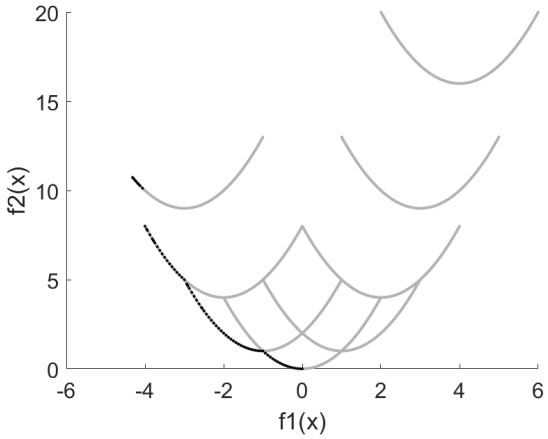


Figure 8: The set $\mathcal{L}_{PNS}$ obtained by applying `MOMIX` with (br2)

4.2 Results on scalable instances

In this section we show results on three different test instances of (MOMIC), all scalable in the number of variables. We apply `MOMIX` and $\texttt{MOMIX}_{light}$ in combination with (br1) and (br2) on all instances. We analyze the impact of the branching rules as well as the difference between `MOMIX` and $\texttt{MOMIX}_{light}$. Recall that `MOMIX` uses stronger lower bounds but these require to solve single-objective mixed integer convex programming problems.

Test instance 4.2 *This instance has quadratic objective functions and the number of integer variables can be set to different values. Let the matrices Q_1 and Q_2 be defined as follows:*

$$(Q_1)_{i,j} = \begin{cases} 3 & \text{if } i=j=1 \\ 4 & \text{if } i=j=n \\ 1 & \text{else} \end{cases} \qquad \text{and } (Q_2)_{i,j} = \begin{cases} 2 & \text{if } i=j=1 \text{ or } i=j=n \\ 4 & \text{if } i=j \text{ and } i \notin \{1,n\} \\ 1 & \text{else.} \end{cases}$$

Then the optimization problem is stated by

$$\begin{aligned} \min \quad & \begin{pmatrix} x^T Q_1^T Q_1 x + (1,2,\dots,2,1)x \\ x^T Q_2^T Q_2 x + (-1,-2,\dots,-2,5)x \end{pmatrix} \\ \text{s.t.} \quad & x_i \in [-5,5],\ i \in \{1,\dots,n\} \\ & I = \{3,\dots,n\}. \end{aligned} \tag{T2}$$

Note that $Q_1^T Q_1$ and $Q_2^T Q_2$ are positive semidefinite and hence f_1 and f_2 are convex.

Test instance 4.3 *This instance is also scalable in the number of integer variables.*

$$\begin{aligned} \min \quad & \begin{pmatrix} x_1 \\ x_2 + \sum_{i=3}^{n} 10(x_i - 0.4)^2 \end{pmatrix} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i^2 \leq 4 \\ & x_i \in [-2,2] \text{ for all } i = 1,\dots,n \\ & I = \{3,\dots,n\} \end{aligned} \tag{T3}$$

Here, we can explicitly give the set of all efficient points by

$$E = \{x \in \mathbb{R}^n \mid x_1^2 + x_2^2 = 4, x_1 \in [-2,0], x_2 \in [-2,0], x_i = 0 \text{ for all } i \geq 3\}.$$

Test instance 4.4 *In this instance both, the number of continuous and integer variables, can be set to different values, with the restriction that $k^c = |\{1,\dots,n\} \setminus I|$ has to be even.*

$$\begin{aligned} \min \quad & \begin{pmatrix} \sum_{i=1}^{k^c/2} x_i + \sum_{i=k^c+1}^{n} x_i \\ \sum_{i=k^c/2+1}^{k^c} x_i - \sum_{i=k^c+1}^{n} x_i \end{pmatrix} \\ \text{s.t.} \quad & \sum_{i=1}^{k^c} x_i^2 \leq 1 \\ & x_i \in [-2,2] \text{ for all } i = 1,\dots,n \\ & I = \{k^c+1,...,n\} \end{aligned} \tag{T4}$$

For both objective functions the Lipschitz constant is $L = \sqrt{k^c/2 + |I|}$.

For all instances but (T4) we set half an hour (1800 seconds) as time limit. For (T4) we set the time limit to one hour (3600 seconds).

In Figures 9, 10 and 11 we show our results in the image space. As the set $\mathcal{L}_{PNS}$ is similar for all versions of `MOMIX` and choices of the branching rule within one test instance, we present only the results for `MOMIX` with (br2) within the figures. In black we plot the points of $\mathcal{L}_{PNS}$. The gray points are the images of the feasible points, i.e., the upper bounds, computed along the algorithm. The parameter for the set from Theorem 3.13 applied to (T4) with $k^c = 2$ and $|I| = 1$ is $L\delta = 0.1\sqrt{2}$. Hence, the set described by $\bigcup_{z\in\mathcal{L}_{PNS}}(\{z - L\delta e\} + \mathbb{R}^m_+)$ is just a rough lower bound of the nondominated set. From a practical point of view, in all our test runs, the points from the lists $\mathcal{L}_{PNS}$ deliver a good approximation of the nondominated sets.

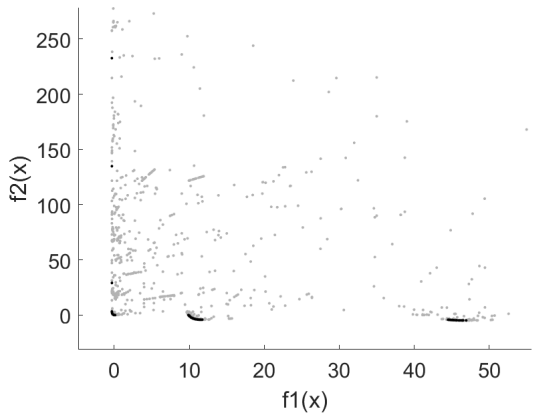


Figure 9: The set $\mathcal{L}_{PNS}$ of Instance (T2) for $|I| = 5, n = 7$.

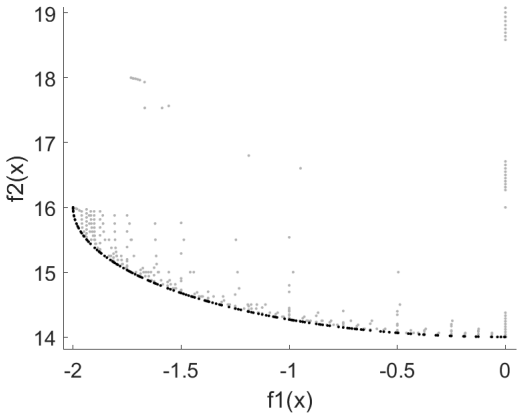
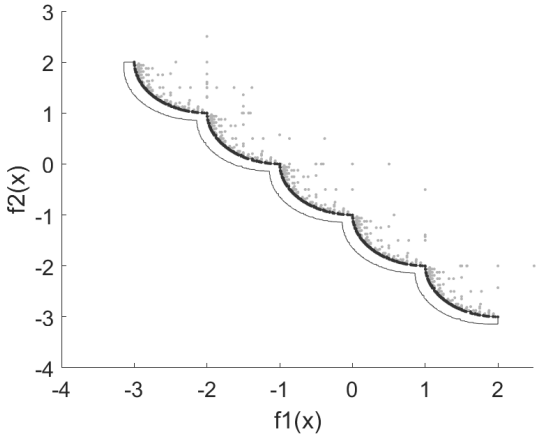


Figure 10: The set $\mathcal{L}_{PNS}$ of Instance (T3) for $|I| = 10, n = 12$.



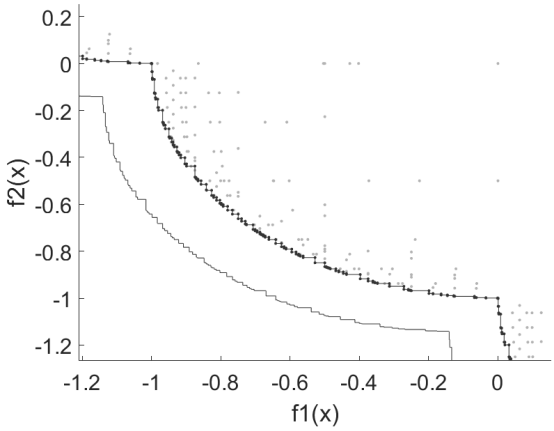


Figure 11: The set $\mathcal{L}_{PNS}$ of Instance (T4) for $|I| = 1, n - |I| = 2$ and the boundary of the set from Theorem 3.13, $L\delta = 0.1\sqrt{2}$. Right picture shows a detail of the left one.

The numerical results on all instances are shown in Table 1. In the first two columns we report the number of integer ($|I|$) and the number of continuous variables ($|C|$) for each instance. For both, `MOMIX` and $\texttt{MOMIX}_{light}$, we report the total computational time in seconds (CPU) and the number of considered boxes in the branching tree (#nod).

For MOMIX we additionally report the total time needed by Gurobi to address the single-objective mixed integer quadratic problems (MIQP). Failures, i.e., instances for which the time limit was exceeded, are marked with "-".

		MOMIX						MOMIX$_{light}$			
		(br1)			(br2)			(br1)		(br2)	
$\|I\|$	$\|C\|$	CPU	#nod	MIQP	CPU	#nod	MIQP	CPU	#nod	CPU	#nod
Test instance T2 - time limit 1800s											
1	2	40.1	757	2.3	38.7	765	2.3	849.9	609	524.5	669
2	2	30.8	537	1.6	31.6	575	1.7	667.2	555	563.0	641
3	2	31.0	535	1.5	30.8	521	1.5	1381.2	1127	814.4	917
4	2	34.7	567	1.7	65.6	1095	3.0	-	-	1134.9	1285
5	2	38.5	587	1.6	81.5	1259	3.2	-	-	-	-
10	2	350.3	2707	9.5	-	-	-	-	-	-	-
Test instance T3 - time limit 1800s											
1	2	15.5	301	0.5	14.6	299	0.4	1045.4	299	1025.6	299
10	2	36.5	413	1.2	27.1	353	0.7	-	-	-	-
20	2	-	-	-	46.9	411	0.9	-	-	-	-
30	2	-	-	-	80.4	471	1.1	-	-	-	-
50	2	-	-	-	-	-	-	-	-	-	-
Test instance T4 - time limit 3600s											
1	2	41.5	749	1.3	44.3	771	1.3	296.3	747	225.6	801
2	2	226.2	3683	6.3	240.5	3761	6.2	-	-	3090.4	3701
3	2	1354.9	19127	32.3	1321.5	18451	31.1	-	-	-	-
1	4	2199.5	23935	53.5	2246.6	24399	53.8	-	-	-	-

Table 1: Numerical results for test instances (T2), (T3) and (T4).

We observe that MOMIX outperforms MOMIX$_{light}$ on all test instances. MOMIX is able to solve a higher number of instances within the time limit. This seems to indicate that the improved lower bounding procedure of MOMIX and the effort in solving single-objective mixed integer convex problems pays off. We notice that the time Gurobi needs to address the single-objective mixed integer subproblems is a small percentage of the whole computational time. By using the MATLAB profiler on our code we got that the bottleneck in our implementation is `fmincon`: Most of the CPU time was spent to solve the single-objective continuous convex problems. In fact, for high dimensional test instances `fmincon` was not able to solve some of the single-objective continuous convex problems. This was the case for, e.g., Instance (T3) with $|I| = 50$. Note that `fmincon` can be replaced by any solver for convex problems within both MOMIX and MOMIX$_{light}$.

Regarding the two branching rules, we can notice some differences as soon as the dimension of the instances grows.

4.3 Results on a triobjective instance

Our implementation of MOMIX and MOMIX$_{light}$ can handle instances of (MOMIC) with a general number of objective functions $m \geq 2$. In the following, we present the results obtained by applying MOMIX with branching rule (br2).

Test instance 4.5 *We consider the triobjective mixed integer instance*

$$\begin{aligned} \min \quad & \begin{pmatrix} x_1 + x_4 \\ x_2 - x_4 \\ x_3 + x_4^2 \end{pmatrix} \\ \text{s.t.} \quad & \sum_{i=1}^{3} x_i^2 \leq 1 \\ & x_i \in [-2, 2] \text{ for all } i = 1, \ldots, 4 \\ & x_4 \in \mathbb{Z}. \end{aligned} \tag{T5}$$

We set $\delta = 0.5$ in `MOMIX`. In order to detect $\mathcal{L}_{\mathcal{S}}$, the cover of the efficient set of Problem (T5), `MOMIX` needed to explore 1237 nodes. This was done within 190 seconds CPU time.

In Figure 12 the points in $\mathcal{L}_{PNS}$ are plotted in black, giving an approximation of the nondominated set of Problem (T5). In gray we plot the images of the feasible points computed along the algorithm.

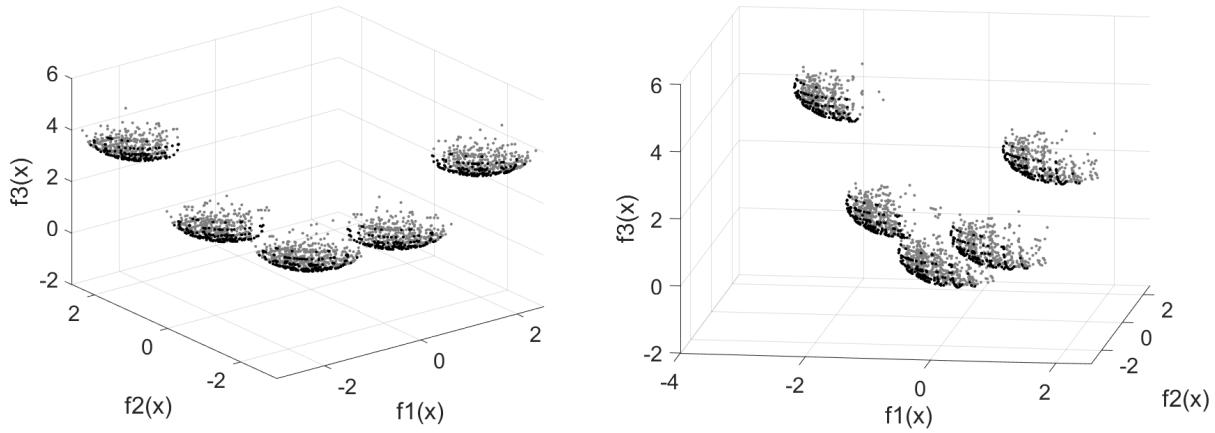


Figure 12: The set $\mathcal{L}_{PNS}$ for Problem (T5) from two different perspectives.

4.4 Results on a convex instance

As a further example, we report the results obtained applying `MOMIX`$_{light}$ with branching rule (br1) on the following non-quadratic instance:

Test instance 4.6

$$\begin{aligned} \min \quad & \begin{pmatrix} x_1 + x_3 \\ x_2 + \exp(-x_3) \end{pmatrix} \\ \text{s.t.} \quad & x_1^2 + x_2^2 \leq 1 \\ & x_i \in [-2, 2] \text{ for all } i = 1, \ldots, 3 \\ & x_3 \in \mathbb{Z} \end{aligned} \tag{T6}$$

Note that the second objective of Problem (T6) is a convex non-quadratic function.

As already mentioned at the beginning of the section, in our implementation of `MOMIX` we use GUROBI [19] as mixed integer quadratic solver and we did not include any other solver within it. Therefore, in order to solve Problem (T6) we applied $\texttt{MOMIX}_{light}$ setting $\delta = 0.1$. $\texttt{MOMIX}_{light}$ was able to detect $\mathcal{L}_{\mathcal{S}}$ by addressing 1105 nodes within 20 seconds CPU time. In Figure 13, we plot the obtained approximation of the nondominated set of Problem (T6).

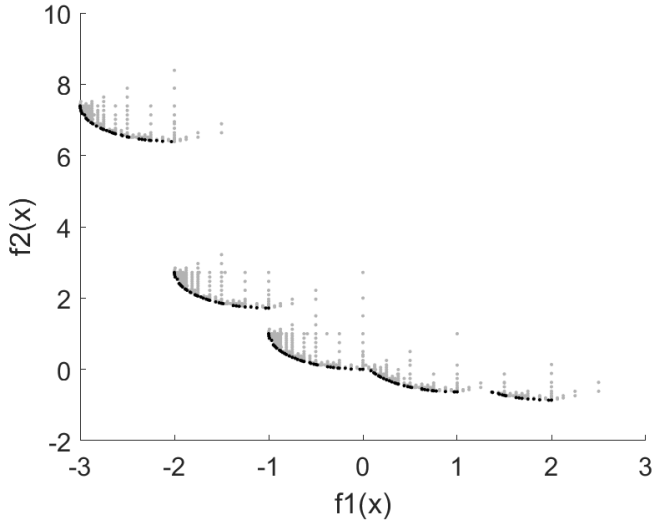


Figure 13: The set $\mathcal{L}_{PNS}$ of Problem (T6) obtained by $\texttt{MOMIX}_{light}$.

Assume that Problem (T6) is solved by using the ε-constraint method. The ε-constraint scalarization of (T6) for some $\varepsilon \in \mathbb{R}$ is then defined by

$$\begin{aligned} \min \quad & f_1(x) = x_1 + x_3 \\ \text{s.t.} \quad & f_2(x) = x_2 + \exp(-x_3) \leq \varepsilon \\ & x_1^2 + x_2^2 \leq 1 \\ & x_i \in [-2, 2] \text{ for all } i = 1, \ldots, 3 \\ & x_3 \in \mathbb{Z}. \end{aligned} \tag{7}$$

Considering the gap in the nondominated set of (T6) (see Figure 13), we have that solving Problem (7) for all values ε in the interval $[3, 6]$ would lead to the same solution. The significant values for ε are only those in the intervals $[-1, 3]$ and $[6, 7.5]$. Clearly, the significant intervals are not known in advance and this is a big issue when applying the ε-constraint method on (MOMIC), as the nondominated set of a multiobjective mixed integer convex problem may have huge gaps.

5 Conclusions

In this paper we devised the first deterministic algorithm for solving multiobjective mixed integer convex programming problems. The method is based on linear outer approximations of the image set. We first build linear outer approximations of the convex relaxation

of the problem by adaptively computing hyperplanes considering some meaningful local upper bounds. Then, in case we want to improve our lower bound, we compute additional hyperplanes that outer approximate the convex hull of the true image set. This is again done in an adaptive way, taking into account specific local upper bounds. The local upper bound sets are updated as soon as a new upper bound is found and are used both to have a pruning criterion and to approximate the dominated set. Theoretical results related to the correctness of our algorithm are provided. Numerical examples on both, biobjective and triobjective, instances show the ability of our procedure to detect nondominated points of multiobjective mixed integer convex programming problems. We also explored the possibility of using two different branching rules.

6 Acknowledgments

The first author acknowledges support within the DAAD scholarship No 57440915. She further acknowledges support within the project No RP1181641D22304F which has received funding from Sapienza, University of Rome. The third author thanks the Carl-Zeiss-Stiftung and the DFG-founded Research Training Group 1567 for financial support. The work of the fourth author is funded by the Deutsche Forschungsgemeinschaft under grant No. EI 821/4.

References

[1] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 2013.

[2] N. Boland, H. Charkhgard, and M. Savelsbergh. A criterion space search algorithm for biobjective integer programming: The balanced box method. *INFORMS J. Comput.*, 27(4):735–754, 2015.

[3] N. Boland, H. Charkhgard, and M. Savelsbergh. The l-shape search method for triobjective integer programming. *Math. Program. Comput.*, 8(2):217–251, 2016.

[4] N. Boland, H. Charkhgard, and M. Savelsbergh. A new method for optimizing a linear function over the efficient set of a multiobjective integer program. *Eur. J. Oper. Res.*, 260(3):904–919, 2017.

[5] N. Boland, H. Charkhgard, and M. Savelsbergh. The quadrant shrinking method: A simple and efficient algorithm for solving tri-objective integer programs. *Eur. J. Oper. Res.*, 260(3):873–885, 2017.

[6] N. Boland, A. C. Eberhard, F. Engineer, and A. Tsoukalas. A new approach to the feasibility pump in mixed integer programming. *SIAM J. Optim.*, 22(3):831–861, 2012.

[7] P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programs. *Math. Program.*, 119(2):331–352, 2009.

[8] R. S. Burachik, C. Y. Kaya, and M. M. Rizvi. Algorithms for generating pareto fronts of multi-objective integer and mixed-integer programming problems. arXiv:1903.07041v1.

[9] R. S. Burachik, C. Y. Kaya, and M. M. Rizvi. A new scalarization technique and new algorithms to generate pareto fronts. *SIAM J. Optim.*, 27(2):1010–1034, 2017.

[10] V. Cacchiani and C. D'Ambrosio. A branch-and-bound based heuristic algorithm for convex multi-objective minlps. *Eur. J. Oper. Res.*, 260(3):920–933, 2017.

[11] M. De Santis, S. Lucidi, and F. Rinaldi. A new class of functions for measuring solution integrality in the feasibility pump approach. *SIAM J. Optim.*, 23(3):1575–1606, 2013.

[12] M. Ehrgott and X. Gandibleux. Bound sets for biobjective combinatorial optimization problems. *Comput. Oper. Res.*, 34:2674–2694, 2007.

[13] M. Ehrgott, L. Shao, and A. Schöbel. An approximation algorithm for convex multi-objective programming problems. *J. Global Optim.*, 50(3):397–416, 2011.

[14] M. Ehrgott, C. Waters, R. Kasimbeyli, and O. Ustun. Multiobjective programming and multiattribute utility functions in portfolio optimization. *INFOR Inf. Syst. Oper. Res.*, 47(1):31–42, 2009.

[15] M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Math. Program.*, 104(1):91–104, 2005.

[16] B. Geißler, A. Morsi, L. Schewe, and M. Schmidt. Penalty alternating direction methods for mixed-integer optimization: A new view on feasibility pumps. *SIAM J. Optim.*, 27(3):1611–1636, 2017.

[17] A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, C. Schubert, F. Serrano, Y. Shinano, J. M. Viernickel, M. Walter, F. Wegscheider, J. T. Witt, and J. Witzig. The SCIP Optimization Suite 6.0. ZIB-Report 18-26, Zuse Institute Berlin, July 2018.

[18] O. Günlük, J. Lee, and R. Weismantel. Minlp strengthening for separable convex quadratic transportation-cost ufl. *IBM Res. Report*, pages 1–16, 2007.

[19] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2018.

[20] J. Jahn. *Vector Optimization*. Springer, 2009.

[21] K. Klamroth, R. Lacour, and D. Vanderpooten. On the representation of the search region in multi-objective optimization. *Eur. J. Oper. Res.*, 245(3):767–778, 2015.

[22] A. Löhne, B. Rudloff, and F. Ulus. Primal and dual approximation algorithms for convex vector optimization problems. *J. Global Optim.*, 60(4):713–736, 2014.

[23] G. Mavrotas. Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Appl. Math. Comput.*, 213(2):455–465, 2009.

[24] G. Mavrotas and D. Diakoulaki. A branch and bound algorithm for mixed zero-one multiple objective linear programming. *Eur. J. Oper. Res.*, 107(3):530–541, 1998.

[25] G. Mavrotas and D. Diakoulaki. Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming. *Appl. Math. Comput.*, 171(1):53–71, 2005.

[26] J. Niebling and G. Eichfelder. A branch-and-bound-based algorithm for nonconvex multiobjective optimization. *SIAM J. Optim.*, 29(1):794–821, 2019.

[27] Julia Niebling and Gabriele Eichfelder. A branch-and-bound algorithm for bi-objective problems. In *Proceedings of the XIII Global Optimization Workshop GOW'16*, pages 57–60, 2016.

[28] Y. Peng and L. Yu. Multiple criteria decision making in emergency management. *Comput. Oper. Res.*, 42:1–2, 2014.

[29] F. Sourd and O. Spanjaard. A multiobjective branch-and-bound framework: Application to the biobjective spanning tree problem. *INFORMS J. Comput.*, 20(3):472–484, 2008.

[30] P. Xidonas, G. Mavrotas, and J. Psarras. Equity portfolio construction and selection using multiobjective mathematical programming. *J. Global Optim.*, 47(2):185–209, 2010.