Multiobjective Mixed Integer Nonlinear Programming (MOMINLP): decision and criterion space search algorithms

Marianna De Santis

ESR days - 4 March 2021

SAPIENZA
UNIVERSITÀ DI ROMA

Outline of the lecture

- Introduction to MOMINLP
  - formulation of the problem
  - basic definitions
  - solution approaches

Outline of the lecture

- Introduction to MOMINLP
  - formulation of the problem
  - basic definitions
  - solution approaches
- FPA: a criterion space search algorithm for bi-objective integer nonlinear programming problems

Outline of the lecture

- Introduction to MOMINLP
  - formulation of the problem
  - basic definitions
  - solution approaches

- `FPA`: a criterion space search algorithm for bi-objective integer nonlinear programming problems

- `MOMIX`: a decision space search algorithm for multi-objective mixed integer convex programming problems

Problem Formulation

Multiobjective Mixed Integer Nonlinear programming problems (MOMINLPs) can be formulated as follows:

$$\begin{aligned} \min \quad & (f_1(x), \ldots, f_m(x))^T \\ \text{s.t.} \quad & g_k(x) \leq 0 \quad k = 1, \ldots, p \\ & x_i \in \mathbb{Z} \quad \forall i \in I, \end{aligned} \tag{MOMINLP}$$

Problem Formulation

Multiobjective Mixed Integer Nonlinear programming problems (MOMINLPs) can be formulated as follows:

$$\begin{aligned} \min \quad & (f_1(x), \ldots, f_m(x))^T \\ \text{s.t.} \quad & g_k(x) \leq 0 \quad k = 1, \ldots, p \\ & x_i \in \mathbb{Z} \quad \forall i \in I, \end{aligned} \tag{MOMINLP}$$

where

- $f_j, g_k : \mathbb{R}^n \to \mathbb{R}; \; j = 1, \ldots, m; \; k = 1, \ldots, p$
- the index set $I \subseteq \{1, \ldots, n\}$ specifies which variables have to take integer values

Motivation

Multiobjective mixed integer optimization problems arise in **many application fields** such as

- engineering
- finance
- design of water distribution networks
- location or production planning
- emergency management

see e.g. [Pecci et al. OPTE (2018)], [Yenisey et al. Omega (2014)], [Liu et al. C&OR (2014)], [Xinodas et al. JOGO (2010)], [Ehrgott et al. INFOR (2009)]

Basic definitions

- point $x^* \in \mathcal{F}$ is **efficient** for (MOMIC) if there is no $x \in \mathcal{F}$ 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)

- point $z^* = f(x^*) \in \mathbb{R}^m$ is **nondominated** for (MOMIC) if $x^* \in \mathcal{F}$ is an efficient point for (MOMIC)

  The set of all nondominated points of (MOMIC) is the **nondominated set** of (MOMIC)

Basic definitions

- point $x^* \in \mathcal{F}$ is **efficient** for (MOMIC) if there is no $x \in \mathcal{F}$ 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)

- point $z^* = f(x^*) \in \mathbb{R}^m$ is **nondominated** for (MOMIC) if $x^* \in \mathcal{F}$ is an efficient point for (MOMIC)

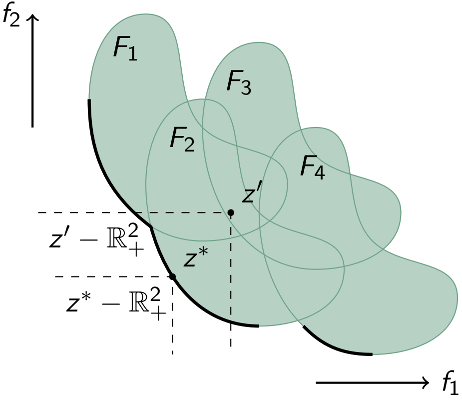
  The set of all nondominated points of (MOMIC) is the **nondominated set** of (MOMIC)

- Let $x^*, x \in \mathcal{F}$ 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** $f(x)$

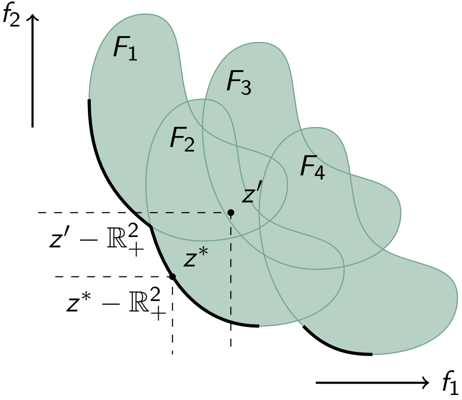
Challenges of multiobjective mixed integer programming

Example: image set of a bi-objective instance



Challenges of multiobjective mixed integer programming

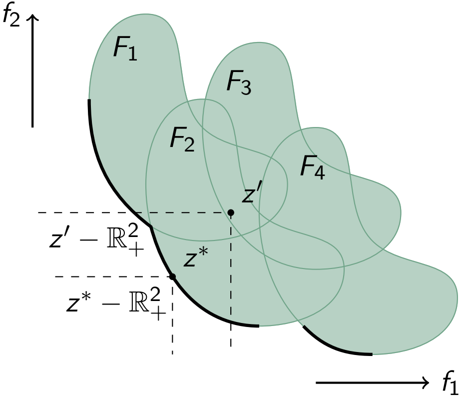
Example: image set of a bi-objective instance



- the union of all F_j describes the whole image set

Challenges of multiobjective mixed integer programming

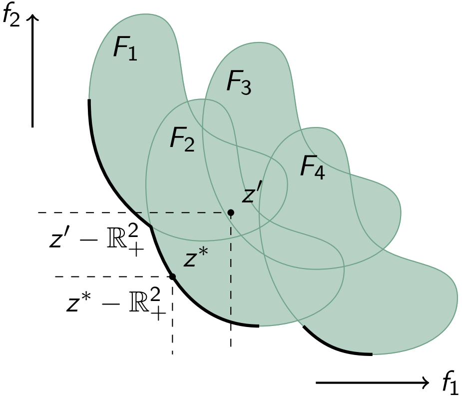
Example: image set of a bi-objective instance



- the union of all F_j describes the whole image set
- z^* is a nondominated point and the preimage of z^* is an efficient point

Challenges of multiobjective mixed integer programming

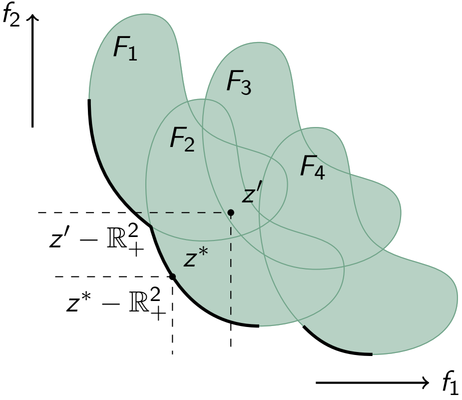
Example: image set of a bi-objective instance



- the union of all F_j describes the whole image set
- z^* is a nondominated point and the preimage of z^* is an efficient point
- z' is dominated because $z^* \leq z'$ and $z^* \neq z'$.

Challenges of multiobjective mixed integer programming

Example: image set of a bi-objective instance



- the union of all F_j describes the whole image set
- z^* is a nondominated point and the preimage of z^* is an efficient point
- z' is dominated because $z^* \leq z'$ and $z^* \neq z'$.
- all the points $z \in F_3$ are dominated

Solution approaches

Solution approaches

- **Criterion space search algorithms**:
  methods that work in the space of the objective functions

Solution approaches

- **Criterion space search algorithms**:
  methods that work in the space of the objective functions

  find non-dominated points by addressing a **sequence of single-objective optimization problems**

Solution approaches

- **Criterion space search algorithms**:
  methods that work in the space of the objective functions

  find non-dominated points by addressing a **sequence of single-objective optimization problems**

- **Decision space search algorithms**:
  approaches that work in the space of decision variables

Solution approaches

- **Criterion space search algorithms**:
  methods that work in the space of the objective functions

  find non-dominated points by addressing a **sequence of single-objective optimization problems**

- **Decision space search algorithms**:
  approaches that work in the space of decision variables

  extend approaches developed for single-objective MINLPs to the case of multiple objectives

FPA: a criterion space search algorithm for bi-objective integer nonlinear programming problems

M. De Santis, G. Grani, L. Palagi
Branching with hyperplanes in the criterion space: The frontier partitioner algorithm for bi-objective integer programming.
European Journal of Operational Research, 283(1), 57-69 (2020)

Problem Formulation

We address **bi-objective integer programming problems**

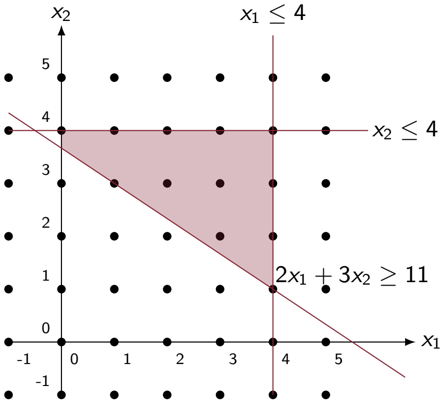
$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} (f_1(x), f_2(x)) \qquad \text{(BOIP)}$$

where

- $\mathcal{X} \subseteq \mathbb{R}^n$
- $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ are continuous

Example:

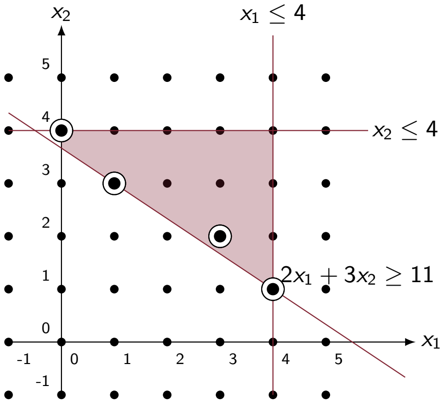
[Ehrgott "Multicriteria Optimization" - 2005]



$$
\begin{aligned}
\text{min} \quad & (x_1, x_2) \\
\text{s.t.} \quad & 2x_1 + 3x_2 \geq 11 \\
& x \in [0, 4] \cap \mathbb{Z}^2
\end{aligned}
$$

Example: $\mathcal{Y}_N = \{(0,4);\ (1,3);\ (3,2);\ (4,1)\}$

[Ehrgott "Multicriteria Optimization" - 2005]



$$
\begin{array}{ll}
\min & (x_1, x_2) \\
\text{s.t.} & 2x_1 + 3x_2 \geq 11 \\
 & x \in [0,4] \cap \mathbb{Z}^2
\end{array}
$$

Criterion space algorithms

Criterion space search algorithms find non-dominated points by addressing a **sequence of single-objective optimization problems**

Criterion space algorithms

Criterion space search algorithms find non-dominated points by addressing a **sequence of single-objective optimization problems**

Once a non-dominated point is computed, the **dominated parts of the criterion space are removed** and the algorithms go on looking for **new** non-dominated points

Criterion space algorithms

solving single-objective optimization problems to get non-dominated points

Criterion space algorithms

solving single-objective optimization problems to get non-dominated points

We refer to the **weighted-sum scalarization problem (INLP)** defined as

$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \lambda_1 f_1(x) + \lambda_2 f_2(x) \qquad \text{(INLP)}$$

where $\lambda_1 + \lambda_2 = 1$, with $\lambda_i \geq 0, \;$ for $i = 1, 2$

Criterion space algorithms

solving single-objective optimization problems to get non-dominated points

We refer to the **weighted-sum scalarization problem (INLP)** defined as

$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \lambda_1 f_1(x) + \lambda_2 f_2(x) \qquad \text{(INLP)}$$

where $\lambda_1 + \lambda_2 = 1$, with $\lambda_i \geq 0$, for $i = 1, 2$

Proposition

Let $\lambda_1, \lambda_2 > 0$, then each solution of Problem (INLP) *is an efficient solution for Problem* (BOIP)

Criterion space algorithms

solving single-objective optimization problems to get non-dominated points

We refer to the **weighted-sum scalarization problem (INLP)** defined as

$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \lambda_1 f_1(x) + \lambda_2 f_2(x) \qquad \text{(INLP)}$$

where $\lambda_1 + \lambda_2 = 1$, with $\lambda_i \geq 0$, for $i = 1, 2$

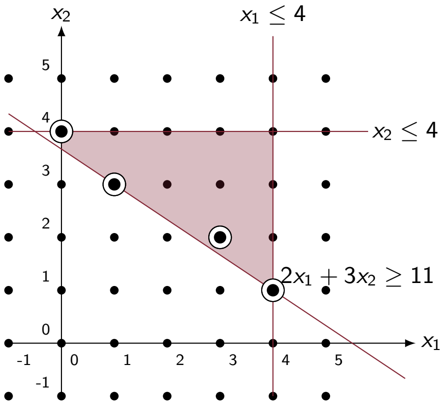
Proposition

Let $\lambda_1, \lambda_2 > 0$, then each solution of Problem (INLP) *is an efficient solution for Problem* (BOIP)

The converse is true **only under proper convexity assumptions!!**

Example: $\mathcal{Y}_N = \{(0,4);\ (1,3);\ (3,2);\ (4,1)\}$

[Ehrgott "Multicriteria Optimization" - 2005]



Point $(3,2)^\top$ **cannot be found** by weighted-sum scalarization!!

The Frontier Partitioner Algorithm FPA

Key ingredients

FPA is a **Criterion Space search Algorithm**

The Frontier Partitioner Algorithm FPA

Key ingredients

FPA is a **Criterion Space search Algorithm**

At each iteration FPA:

- computes one non-dominated solution (when it exists) **addressing a weighted-sum scalarization problem**

The Frontier Partitioner Algorithm FPA

Key ingredients

FPA is a **Criterion Space search Algorithm**

At each iteration FPA:

- computes one non-dominated solution (when it exists) **addressing a weighted-sum scalarization problem**
- in case a non-dominated solution is found, **two subproblems are constructed** using properly defined inequalities

The Frontier Partitioner Algorithm FPA

Positive gap assumption

Definition

Let $f : \mathbb{R}^n \to \mathbb{R}$. We say that f is a positive γ-function if $\gamma \in \mathbb{R}_+$ exists such that $|f(x) - f(z)| \geq \gamma$, for all $x, z \in \mathcal{X} \cap \mathbb{Z}^n$ with $f(x) \neq f(z)$.

The Frontier Partitioner Algorithm FPA

Positive gap assumption

Definition

Let $f : \mathbb{R}^n \to \mathbb{R}$. We say that f is a positive γ-function if $\gamma \in \mathbb{R}_+$ exists such that $|f(x) - f(z)| \geq \gamma$, for all $x, z \in \mathcal{X} \cap \mathbb{Z}^n$ with $f(x) \neq f(z)$.

Assumption (Positive gap value)

The functions $f_i : \mathbb{R}^n \to \mathbb{R}$ in (BOIP) are positive γ-functions

The Frontier Partitioner Algorithm FPA

Definition of the inequalities

Let $\hat{y}^k$ be a **non-dominated point** for (BOIP) found at iteration k

The Frontier Partitioner Algorithm FPA

Definition of the inequalities

Let $\hat{y}^k$ be a **non-dominated point** for (BOIP) found at iteration k

Let $\epsilon_i \in (0, \gamma_i]$, $i = 1, 2$.

The Frontier Partitioner Algorithm FPA

Definition of the inequalities

Let $\hat{y}^k$ be a **non-dominated point** for (BOIP) found at iteration k

Let $\epsilon_i \in (0, \gamma_i]$, $i = 1, 2$.

We consider the inequalities

$$f_i(x) \leq \hat{y}_i^k - \epsilon_i, \quad i = 1, 2$$

The Frontier Partitioner Algorithm FPA

Definition of the inequalities

Let $\hat{y}^k$ be a **non-dominated point** for (BOIP) found at iteration k

Let $\epsilon_i \in (0, \gamma_i]$, $i = 1, 2$.

We consider the inequalities

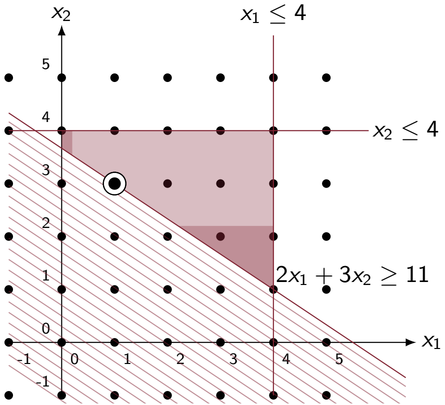
$$f_i(x) \leq \hat{y}_i^k - \epsilon_i, \quad i = 1, 2$$

Remark

The inequalities $f_i(x) \leq \hat{y}_i^k - \epsilon_i, \; i = 1, 2$
cut the non-dominated solution $\hat{y}^k$
and they are **linear in the criterion space**

The Frontier Partitioner Algorithm FPA

Definition of the inequalities



The Frontier Partitioner Algorithm FPA

Definition of the subproblems

Let $\hat{y}^0$ be a **non-dominated point** for (BOIP) found at the first iteration of FPA

The Frontier Partitioner Algorithm FPA

Definition of the subproblems

Let $\hat{y}^0$ be a **non-dominated point** for (BOIP) found at the first iteration of FPA

Starting from $\hat{y}^0$, FPA **defines the following two BOIPs**:

$$\min_{x \in \mathcal{X}_1 \cap \mathbb{Z}^n} (f_1(x), f_2(x)) \qquad \mathcal{X}_1 = \mathcal{X} \cap \{x \in \mathbb{R}^n \,:\, f_1(x) \leq \hat{y}_1^0 - \epsilon_1\}$$

$$\min_{x \in \mathcal{X}_2 \cap \mathbb{Z}^n} (f_1(x), f_2(x)) \qquad \mathcal{X}_2 = \mathcal{X} \cap \{x \in \mathbb{R}^n \,:\, f_2(x) \leq \hat{y}_2^0 - \epsilon_2\}$$

The Frontier Partitioner Algorithm FPA

Definition of the subproblems

Let $\hat{y}^0$ be a **non-dominated point** for (BOIP) found at the first iteration of FPA

Starting from $\hat{y}^0$, FPA **defines the following two BOIPs**:

$$\min_{x \in \mathcal{X}_1 \cap \mathbb{Z}^n} (f_1(x), f_2(x)) \qquad \mathcal{X}_1 = \mathcal{X} \cap \{x \in \mathbb{R}^n \,:\, f_1(x) \leq \hat{y}_1^0 - \epsilon_1\}$$

$$\min_{x \in \mathcal{X}_2 \cap \mathbb{Z}^n} (f_1(x), f_2(x)) \qquad \mathcal{X}_2 = \mathcal{X} \cap \{x \in \mathbb{R}^n \,:\, f_2(x) \leq \hat{y}_2^0 - \epsilon_2\}$$

...and goes on **producing iteratively a finite lists of BOIPs!**

The Frontier Partitioner Algorithm FPA

Convergence analysis

Proposition

At every iteration FPA either states that the **BOIP** *considered* **is infeasible** *or finds a* **yet unknown non-dominated solution***.*

The Frontier Partitioner Algorithm FPA

Convergence analysis

Proposition

At every iteration FPA either states that the **BOIP** *considered* **is infeasible** *or finds a* **yet unknown non-dominated solution**.

Theorem

The Frontier Partitioner Algorithm **returns the complete Pareto frontier** $\mathcal{Y}_N$ *of (BOIP) after having addressed* $2|\mathcal{Y}_N| + 1$ *single-objective integer programs.*

Improving the complexity of FPA

Use smart weights

In order to identify all $|\mathcal{Y}_N|$ non-dominated points of a BOIP by solving a sequence of subproblems, **any criterion space algorithm for BOIPs must solve at least $|\mathcal{Y}_N|$ subproblems**

Improving the complexity of FPA

Use smart weights

In order to identify all $|\mathcal{Y}_N|$ non-dominated points of a BOIP by solving a sequence of subproblems, **any criterion space algorithm for BOIPs must solve at least $|\mathcal{Y}_N|$ subproblems**

$$\Downarrow$$

The complexity of any criterion space algorithm is $O(|\mathcal{Y}_N|)$

Improving the complexity of FPA

Use smart weights

In order to identify all $|\mathcal{Y}_N|$ non-dominated points of a BOIP by solving a sequence of subproblems, **any criterion space algorithm for BOIPs must solve at least $|\mathcal{Y}_N|$ subproblems**

$$\Downarrow$$

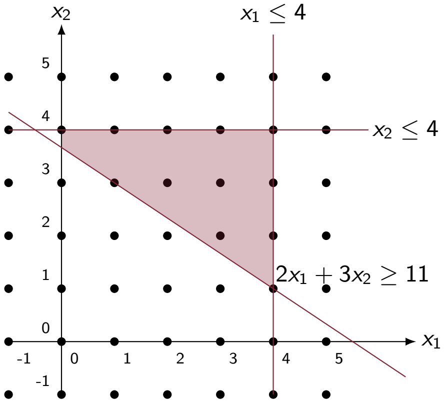
The complexity of any criterion space algorithm is $O(|\mathcal{Y}_N|)$

Remark

We can **drop down the complexity of** *FPA*
from $2|\mathcal{Y}_N| + 1$ *to* $|\mathcal{Y}_N| + 1$ *using smart weights!*

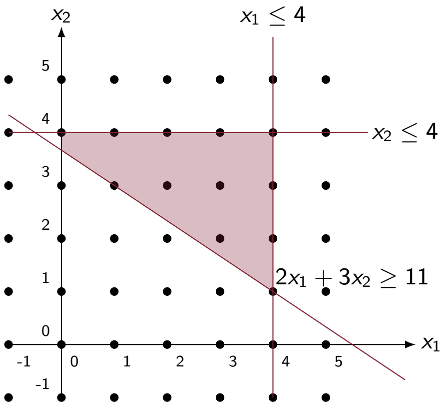
FPA applied to the example

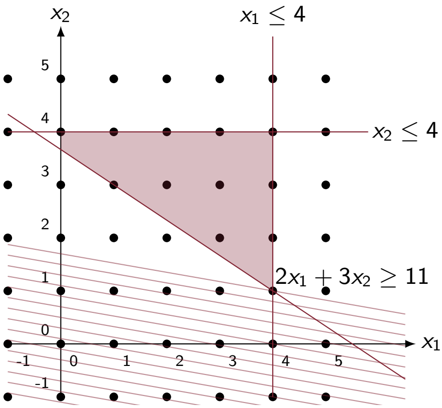
smart weights



FPA applied to the example

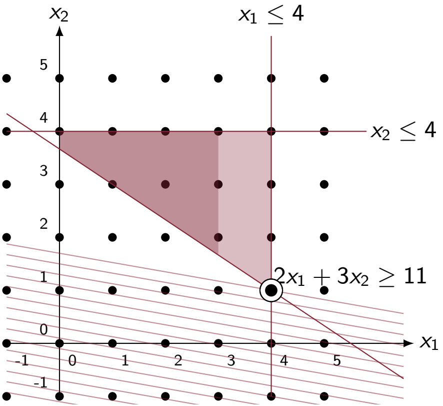
smart weights





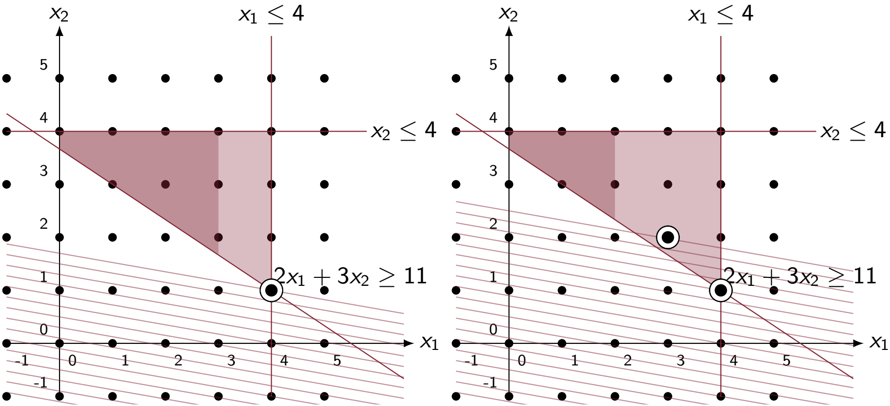
FPA applied to the example

smart weights



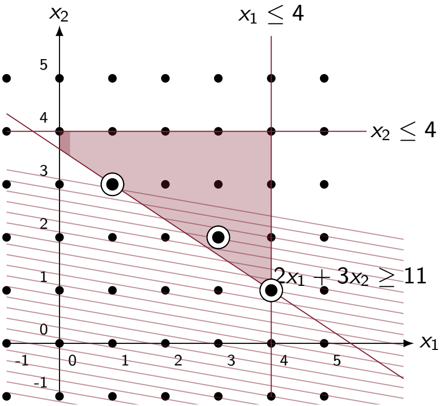
FPA applied to the example

smart weights



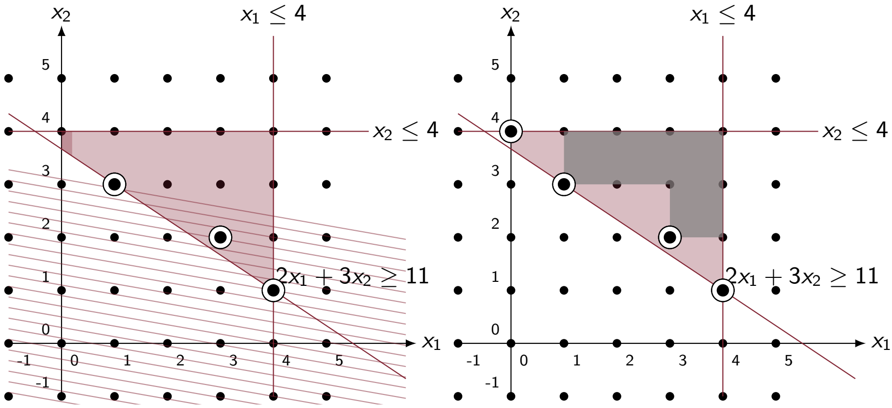
FPA applied to the example

smart weights



FPA applied to the example

smart weights



Which BOIPs can be addressed by FPA?

Which BOIPs can be addressed by FPA?

$f_i(x) =$	γ	INLP oracle
$c^\intercal x$ with $c \in \mathbb{Z}^n$	1	*ILP*
$c^\intercal x$ with $c \in \mathbb{Q}^n$	$\frac{1}{r}$	*ILP*
$x^\intercal Qx + c^\intercal x$ with $Q \succeq 0,\ Q \in \mathbb{Z}^{n\times n},\ c \in \mathbb{Z}^n$	1	*QCQIP*
$x^\intercal Qx + c^\intercal x$ with $Q \succeq 0,\ Q \in \mathbb{Q}^{n\times n},\ c \in \mathbb{Q}^n$	$\frac{1}{r}$	*QCQIP*
$: \mathbb{Z}^n \to \mathbb{Z}$, convex	1	*CIP*

Table: Classes of functions that satisfy the positive gap value assumption.

Numerical results

Algorithm `FPA`

- is implemented in Java
- uses `CPLEX 12.7.1` to address the scalarized problem (INLP)

Numerical results

Algorithm `FPA`

- is implemented in Java
- uses `CPLEX 12.7.1` to address the scalarized problem (INLP)

We took instances available at
http://home.ku.edu.tr/~moolibrary/

Numerical results

Algorithm FPA

- is implemented in Java
- uses CPLEX 12.7.1 to address the scalarized problem (INLP)

We took instances available at
http://home.ku.edu.tr/~moolibrary/

We tested FPA on

- biobjective integer linear instances
  we compare FPA with the *Balanced Box Method*
  [Boland et al. (2015) INFORMS Journal on Computing, 27(4), 735-754]

Numerical results

Algorithm FPA

- is implemented in Java
- uses CPLEX 12.7.1 to address the scalarized problem (INLP)

We took instances available at
http://home.ku.edu.tr/~moolibrary/

We tested FPA on

- biobjective integer linear instances
  we compare FPA with the *Balanced Box Method*
  [Boland et al. (2015) INFORMS Journal on Computing, 27(4), 735-754]
- biobjective integer convex quadratic instances

Comparison via performance profiles

[Dolan, E. and Moré, J. (2002). *Benchmarking optimization software with performance profiles.* Mathematical Programming, 91, 201–213.]

Given

- a **set of solvers** $\mathcal{S}$
- a **set of problems** $\mathcal{P}$

Comparison via performance profiles

[Dolan, E. and Moré, J. (2002). *Benchmarking optimization software with performance profiles.* Mathematical Programming, 91, 201–213.]

Given

- a **set of solvers** $\mathcal{S}$
- a **set of problems** $\mathcal{P}$

We define the **performance ratio**

$$r_{p,s} = t_{p,s} / \min\{t_{p,s'} : s' \in \mathcal{S}\},$$

where $t_{p,s}$ is the computational time

Comparison via performance profiles

[Dolan, E. and Moré, J. (2002). *Benchmarking optimization software with performance profiles.* Mathematical Programming, 91, 201–213.]

Given

- a **set of solvers** $\mathcal{S}$
- a **set of problems** $\mathcal{P}$

We define the **performance ratio**

$$r_{p,s} = t_{p,s} / \min\{t_{p,s'} : s' \in \mathcal{S}\},$$

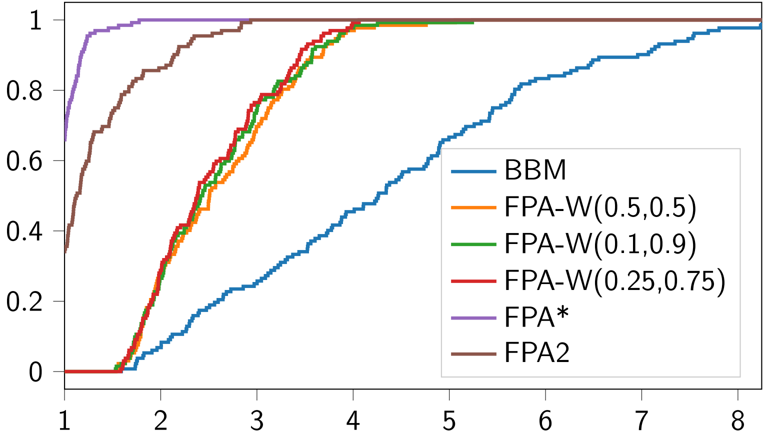
where $t_{p,s}$ is the computational time

The **performance profile** for $s \in S$ is the plot of the **cumulative distribution function** ρ_s:

$$\rho_s(\tau) = |\{p \in \mathcal{P} : \; r_{p,s} \leq \tau\}| / |\mathcal{P}|$$

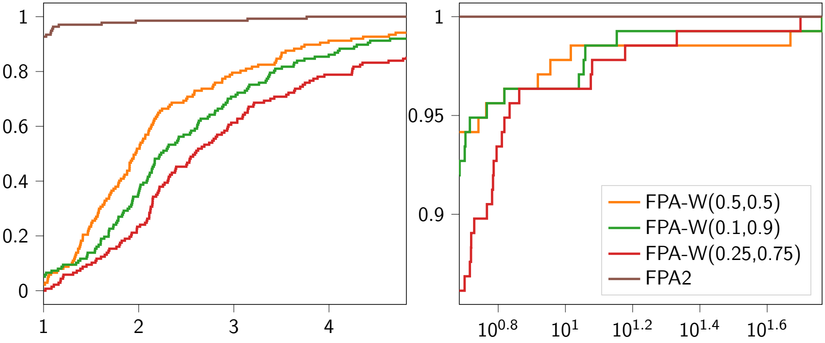
Results on biobjective integer linear instances

Performance profiles related to the CPU time



Results on biobjective integer quadratic instances

Performance profiles related to the CPU time



FPA summary

FPA is a **criterion space algorithm** for **biobjective integer programming problems** that

FPA summary

FPA is a **criterion space algorithm** for **biobjective integer programming problems** that

- Can handle several classes of biobjective integer **nonlinear** programming problems

FPA summary

FPA is a **criterion space algorithm** for **biobjective integer programming problems** that

- Can handle several classes of biobjective integer **nonlinear** programming problems
- It is based on the use of **properly defined inequalities**

FPA summary

FPA is a **criterion space algorithm** for **biobjective integer programming problems** that

- Can handle several classes of biobjective integer **nonlinear** programming problems
- It is based on the use of **properly defined inequalities**
- Has the complexity of $|\mathcal{Y}_N| + 1$

FPA summary

FPA is a **criterion space algorithm** for **biobjective integer programming problems** that

- Can handle several classes of biobjective integer **nonlinear** programming problems
- It is based on the use of **properly defined inequalities**
- Has the complexity of $|\mathcal{Y}_N| + 1$
- On biobjective integer linear programming problems outperforms existing state-of-the art methods

MOMIX: a decision space search method for multi-objective mixed integer convex programming problems

M. De Santis, G. Eichfelder, J. Niebling, S. Rocktäschel
Solving multiobjective mixed integer convex optimization problems,
SIAM Journal on Optimization 30 (4), 3122-3145 (2020)

MOMIX: a decision space search method for (MOMIC)

MOMIX adresses **Multiobjective Mixed Integer Nonlinear** programming problems of the following form:

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

MOMIX: a decision space search method for (MOMIC)

MOMIX adresses **Multiobjective Mixed Integer Nonlinear** programming problems of the following form:

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

where

- $f_j, g_k : B \to \mathbb{R};\ j = 1, \ldots, m;\ k = 1, \ldots, p$ **convex and differentiable**
- $l, u \in \mathbb{R}^n$ are lower and upper bounds on the decision variables
- the index set $I \subseteq \{1, \ldots, n\}$ specifies which variables have to take integer values

MOMIX: a decision space search method for (MOMIC)

main ingredients

MOMIX is a branch-and-bound method based on partitioning the feasible set of (MOMIC)

MOMIX: a decision space search method for (MOMIC)

main ingredients

MOMIX is a branch-and-bound method based on partitioning the feasible set of (MOMIC)

- **Branching rule:** based on bisections of the box B

MOMIX: a decision space search method for (MOMIC)

main ingredients

MOMIX is a branch-and-bound method based on partitioning the feasible set of (MOMIC)

- **Branching rule:** based on bisections of the box B
- **Upper bound computation:** evaluation of the objective functions on feasible points

`MOMIX`: a decision space search method for (MOMIC)

main ingredients

`MOMIX` is a branch-and-bound method based on partitioning the feasible set of (MOMIC)

- **Branching rule:** based on bisections of the box B
- **Upper bound computation:** evaluation of the objective functions on feasible points
- **Lower bound computation:** linear outer approximation of the image set

Some notation

By B^g, $B^{\mathbb{Z}}$ and $B^{g,\mathbb{Z}}$ we denote the following sets related to the constraints in (MOMIC):

$$B^g := \{x \in B \mid g(x) \leq 0\}$$

$$B^{\mathbb{Z}} := \{x \in B \mid x_i \in \mathbb{Z} \text{ for all } i \in I\}$$

$$B^{g,\mathbb{Z}} := B^g \cap B^{\mathbb{Z}}$$

Some notation

By B^g, $B^{\mathbb{Z}}$ and $B^{g,\mathbb{Z}}$ we denote the following sets related to the constraints in (MOMIC):

$$B^g := \{x \in B \mid g(x) \leq 0\}$$

$$B^{\mathbb{Z}} := \{x \in B \mid x_i \in \mathbb{Z} \text{ for all } i \in I\}$$

$$B^{g,\mathbb{Z}} := B^g \cap B^{\mathbb{Z}}$$

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

$$\min\ f(x)$$

$$\text{s.t. } x \in B^{g,\mathbb{Z}}$$

Upper Bounds and Local Upper Bounds

Two lists of points are kept updated and used for pruning:

- $\mathcal{L}_{PNS} \subseteq f(B^{g,\mathbb{Z}})$: *potentially nondominated solutions*

Upper Bounds and Local Upper Bounds

Two lists of points are kept updated and used for pruning:

- $\mathcal{L}_{PNS} \subseteq f(B^{g,\mathbb{Z}})$: *potentially nondominated solutions*
- $\mathcal{L}_{LUB} \subseteq \mathbb{R}^m$: *local upper bounds*
  [Klamroth et al., On the representation of the search region in multi-objective optimization., EJOR (2015)]

Upper Bounds and Local Upper Bounds

Two lists of points are kept updated and used for pruning:

- $\mathcal{L}_{PNS} \subseteq f(B^{g,\mathbb{Z}})$: *potentially nondominated solutions*
- $\mathcal{L}_{LUB} \subseteq \mathbb{R}^m$: *local upper bounds*
  [Klamroth et al., On the representation of the search region in multi-objective optimization., EJOR (2015)]

Theorem

Consider a subbox $\tilde{B} \subseteq B$

Upper Bounds and Local Upper Bounds

Two lists of points are kept updated and used for pruning:

- $\mathcal{L}_{PNS} \subseteq f(B^{g,\mathbb{Z}})$: *potentially nondominated solutions*
- $\mathcal{L}_{LUB} \subseteq \mathbb{R}^m$: *local upper bounds*
  [Klamroth et al., On the representation of the search region in multi-objective optimization., EJOR (2015)]

Theorem

Consider a subbox $\tilde{B} \subseteq B$
Let $\mathcal{L}_{LUB}$ *be the local upper bound set w.r.t.* $\mathcal{L}_{PNS}$

Upper Bounds and Local Upper Bounds

Two lists of points are kept updated and used for pruning:

- $\mathcal{L}_{PNS} \subseteq f(B^{g,\mathbb{Z}})$: *potentially nondominated solutions*
- $\mathcal{L}_{LUB} \subseteq \mathbb{R}^m$: *local upper bounds*
  [Klamroth et al., On the representation of the search region in multi-objective optimization., EJOR (2015)]

Theorem

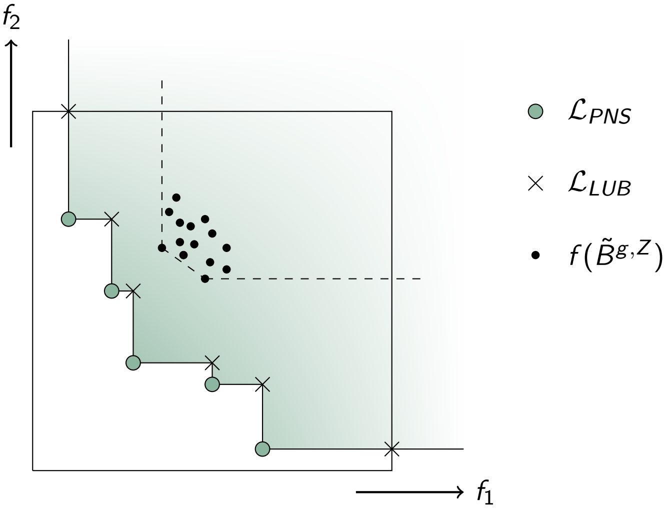
Consider a subbox $\tilde{B} \subseteq B$
Let $\mathcal{L}_{LUB}$ be the local upper bound set w.r.t. $\mathcal{L}_{PNS}$

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

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

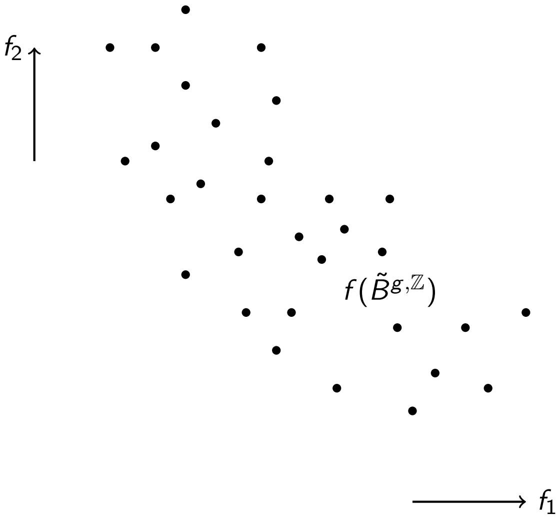
Pruning of the node

example on a bi-objective purely integer instance



Lower bounds

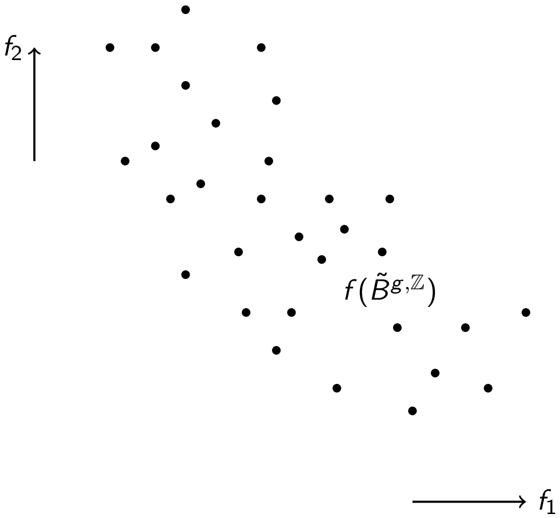
image set of a bi-objective purely integer instance



At every node of the branch-and-bound tree a **subbox $\tilde{B} \subseteq B$ is selected**

Lower bounds

image set of a bi-objective purely integer instance



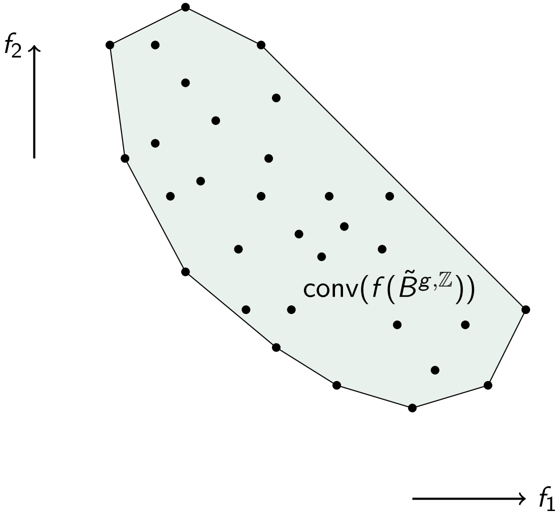
At every node of the branch-and-bound tree a **subbox $\tilde{B} \subseteq B$ is selected**

a lower bound is any set $L_{\tilde{B}} \subseteq \mathbb{R}^m$ such that

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

Lower bounds

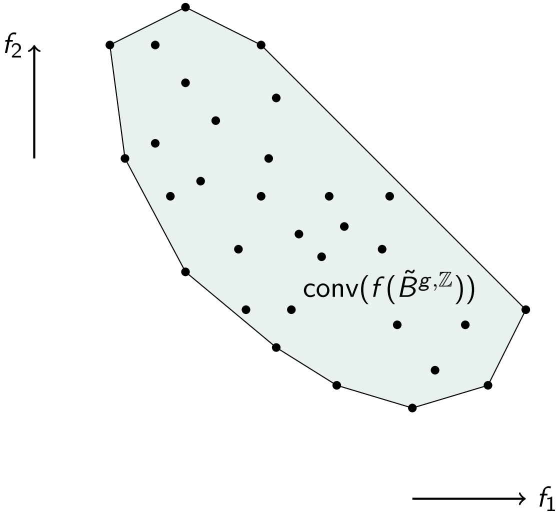
convex hull of the image set



In particular **$\mathbf{conv}(f(\tilde{B}^{g,\mathbb{Z}}))$ is a lower bound**

Lower bounds

convex hull of the image set



In particular **$\mathbf{conv}(f(\tilde{B}^{g,\mathbb{Z}}))$ is a lower bound**

we look for sets $L_{\tilde{B}}$:

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

Lower bounds computation

At every node a **subbox $\tilde{B} \subseteq B$ is selected**

Lower bounds computation

At every node a **subbox $\tilde{B} \subseteq B$ is selected**

and a **linear outer approximation $L_{\tilde{B}}$ of conv**$(f(\tilde{B}^{g,\mathbb{Z}}))$ is built:

Lower bounds computation

At every node a **subbox $\tilde{B} \subseteq B$ is selected**

and a **linear outer approximation $L_{\tilde{B}}$ of conv$(f(\tilde{B}^{g,\mathbb{Z}}))$** is built:

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

Lower bounds computation

At every node a **subbox $\tilde{B} \subseteq B$ is selected**

and a **linear outer approximation $L_{\tilde{B}}$ of conv$(f(\tilde{B}^{g,\mathbb{Z}}))$** is built:

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

$$\Downarrow$$

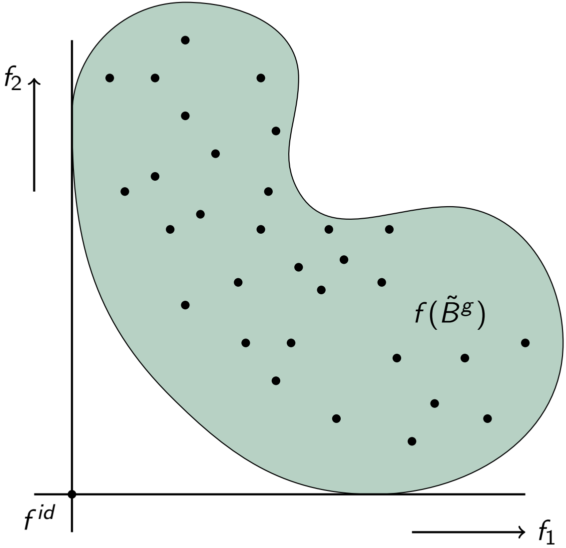
if $\quad p \notin L_{\tilde{B}} + \mathbb{R}^m_+ \quad$ holds for all $p \in \mathcal{L}_{LUB}$

the node can be pruned (or the box $\tilde{B}$ can be discarded) as

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

Lower bounding procedure: Step 1

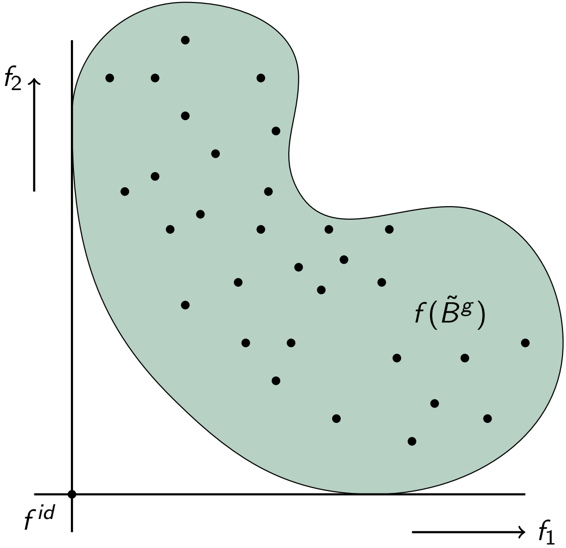
computation of the ideal point



As a first step, we compute the **ideal point** $f^{id} \in \mathbb{R}^m$ of $f(\tilde{B}^g)$

Lower bounding procedure: Step 1

computation of the ideal point



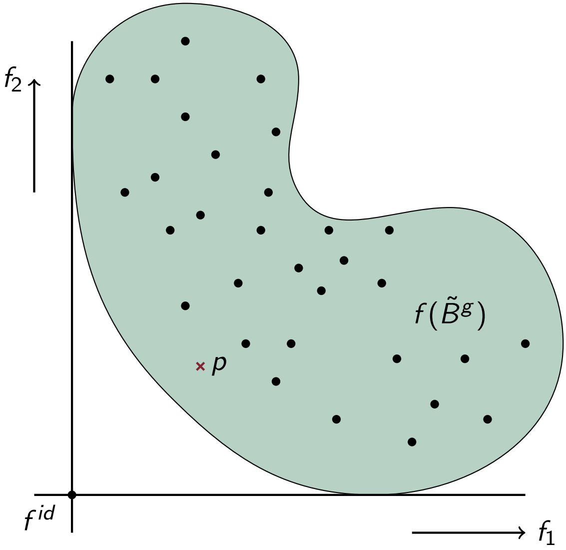
As a first step, we compute the **ideal point** $f^{id} \in \mathbb{R}^m$ of $f(\tilde{B}^g)$

$$f_j^{id} := \min_{x \in \tilde{B}^g} f_j(x)$$

$$j = 1, \ldots, m$$

Lower bounding procedure: Step 2

computation of supporting hyperplanes for $f(\tilde{B}^g)$

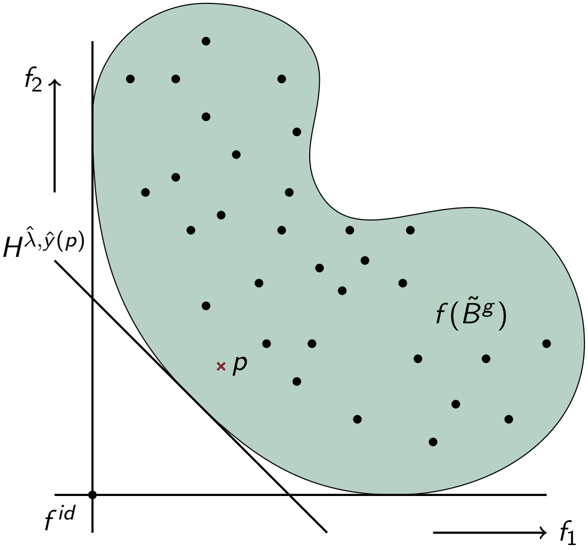


Let $p \in \mathcal{L}_{LUB}$

if $p \in L_{\tilde{B}} + \mathbb{R}^m_+$ we try to **improve** $L_{\tilde{B}}$ by **computing a further hyperplane**

Lower bounding procedure: Step 2

computation of supporting hyperplanes for $f(\tilde{B}^g)$



Let $p \in \mathcal{L}_{LUB}$

if $p \in L_{\tilde{B}} + \mathbb{R}^m_+$ we try to **improve** $L_{\tilde{B}}$ by **computing a further hyperplane**

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

Computation of supporting hyperplanes for $f(\tilde{B}^g)$

address a single-objective continuous convex problem

Let $(\hat{x}, \hat{t}) \in \tilde{B}^g \times \mathbb{R}$ be a minimal solution of the problem

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

Computation of supporting hyperplanes for $f(\tilde{B}^g)$

address a single-objective continuous convex problem

Let $(\hat{x}, \hat{t}) \in \tilde{B}^g \times \mathbb{R}$ be a minimal solution of the problem

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

Then a supporting hyperplane of $f(\tilde{B}^g)$ is given by

$$H^{\hat{\lambda}, \hat{y}(p)} := \{y \in \mathbb{R}^m \mid \hat{\lambda}^T y = \hat{\lambda}^T \hat{y}(p)\}$$

with

- $\hat{\lambda} \in \mathbb{R}^m_+$ a Lagrange multiplier for $f(\hat{x}) \leq p + \hat{t}e$
- $\hat{y}(p) := p + \hat{t}e$

see e.g. [Löhne et al., J. Global Optim. (2014)]

Computation of supporting hyperplanes for $f(\tilde{B}^g)$

Implications

There exist two possibilities:

(i) $\hat{t} > 0$

Computation of supporting hyperplanes for $f(\tilde{B}^g)$

Implications

There exist two possibilities:

(i) $\hat{t} > 0 \Longrightarrow p \notin L_{\tilde{B}} + \mathbb{R}^m_+$

we improve the lower bound adding by $H^{\hat{\lambda},\hat{y}(p)}$

and consider the next local upper bound

Computation of supporting hyperplanes for $f(\tilde{B}^g)$

Implications

There exist two possibilities:

(i) $\hat{t} > 0 \Longrightarrow p \notin L_{\tilde{B}} + \mathbb{R}^m_+$

we improve the lower bound adding by $H^{\hat{\lambda}, \hat{y}(p)}$

and consider the next local upper bound

(ii) $\hat{t} \leq 0$

Computation of supporting hyperplanes for $f(\tilde{B}^g)$

Implications

There exist two possibilities:

(i) $\hat{t} > 0 \implies p \notin L_{\tilde{B}} + \mathbb{R}^m_+$

we improve the lower bound adding by $H^{\hat{\lambda},\hat{y}(p)}$

and consider the next local upper bound

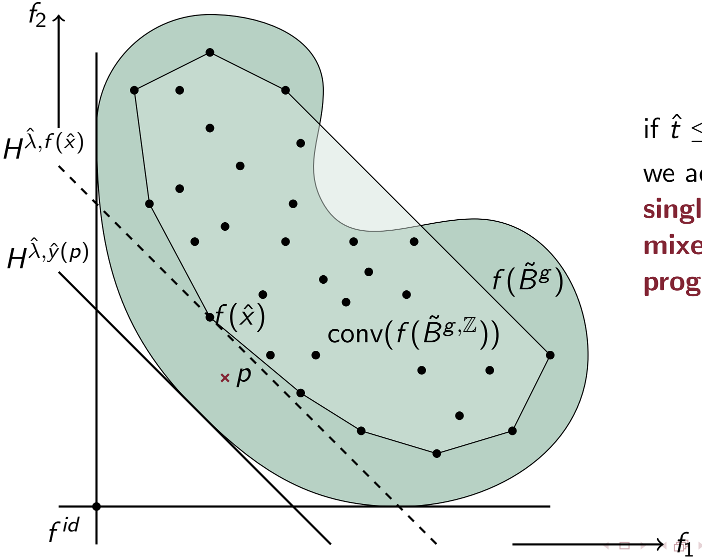
(ii) $\hat{t} \leq 0 \implies p \in L_{\tilde{B}} + \mathbb{R}^m_+$

we cannot prune the node

we refine the outer approximation of $conv(f(\tilde{B}^{g,\mathbb{Z}}))$

Lower bounding procedure: Step 3

computation of supporting hyperplanes for $conv(f(\tilde{B}^{g,\mathbb{Z}}))$

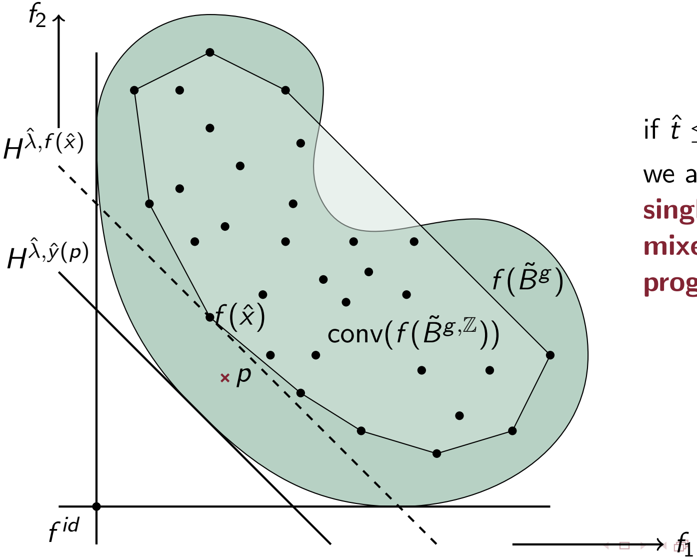


if $\hat{t} \leq 0$

we address a **single-objective mixed integer convex programming problem**

Lower bounding procedure: Step 3

computation of supporting hyperplanes for $conv(f(\tilde{B}^{g,\mathbb{Z}}))$



if $\hat{t} \leq 0$

we address a **single-objective mixed integer convex programming problem**

$$\min \hat{\lambda}^T f(x)$$
$$\text{s.t. } x \in \tilde{B}^{g,\mathbb{Z}}$$

Computation of supporting hyperplanes for $conv(f(\tilde{B}^{g,\mathbb{Z}}))$

address a single-objective mixed integer convex problem

Let $\hat{x} \in \tilde{B}^{g,\mathbb{Z}}$ be a minimal solution of

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

Computation of supporting hyperplanes for $conv(f(\tilde{B}^{g,\mathbb{Z}}))$

address a single-objective mixed integer convex problem

Let $\hat{x} \in \tilde{B}^{g,\mathbb{Z}}$ be a minimal solution of

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

- A supporting hyperplane of $\text{conv}(f(\tilde{B}^{g,\mathbb{Z}}))$ is given by

$$H^{\hat{\lambda}, f(\hat{x})} := \{y \in \mathbb{R}^m \mid \hat{\lambda}^T y = \hat{\lambda}^T f(\hat{x})\}$$

Computation of supporting hyperplanes for $conv(f(\tilde{B}^{g,\mathbb{Z}}))$

address a single-objective mixed integer convex problem

Let $\hat{x} \in \tilde{B}^{g,\mathbb{Z}}$ be a minimal solution of

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

- A supporting hyperplane of $\mathrm{conv}(f(\tilde{B}^{g,\mathbb{Z}}))$ is given by

$$H^{\hat{\lambda},f(\hat{x})} := \{y \in \mathbb{R}^m \mid \hat{\lambda}^T y = \hat{\lambda}^T f(\hat{x})\}$$

- $f(\hat{x})$ is an **upper bound** for (MOMIC)

Again two situations occur:

(i) $\hat{\lambda}^T p < \hat{\lambda}^T f(\hat{x})$

Again two situations occur:

(i) $\hat{\lambda}^T p < \hat{\lambda}^T f(\hat{x})$

we improve the outer approximation by $H^{\hat{\lambda}, f(\hat{x})}$

and consider the next local upper bound

Again two situations occur:

(i) $\hat{\lambda}^T p < \hat{\lambda}^T f(\hat{x})$

we improve the outer approximation by $H^{\hat{\lambda}, f(\hat{x})}$

and consider the next local upper bound

(ii) $\hat{\lambda}^T p \geq \hat{\lambda}^T f(\hat{x})$

Computation of supporting hyperplanes for $conv(f(\tilde{B}^{g,\mathbb{Z}}))$

Implications

Again two situations occur:

(i) $\hat{\lambda}^T p < \hat{\lambda}^T f(\hat{x})$

we improve the outer approximation by $H^{\hat{\lambda}, f(\hat{x})}$

and consider the next local upper bound

(ii) $\hat{\lambda}^T p \geq \hat{\lambda}^T f(\hat{x})$

the local upper bound p lies above

the hyperplane $H^{\hat{\lambda}, f(\hat{x})}$

Computation of supporting hyperplanes for $conv(f(\tilde{B}^{g,\mathbb{Z}}))$

Implications

Again two situations occur:

(i) $\hat{\lambda}^T p < \hat{\lambda}^T f(\hat{x})$

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

(ii) $\hat{\lambda}^T p \geq \hat{\lambda}^T f(\hat{x})$

the local upper bound p lies above
the hyperplane $H^{\hat{\lambda}, f(\hat{x})}$
and we branch the current node by bisecting $\tilde{B}$

Correctness results

detection of both the efficient and the nondominated set

Input of `MOMIX`: $\delta > 0$ prescribed precision

Correctness results

detection of both the efficient and the nondominated set

Input of MOMIX: $\delta > 0$ prescribed precision

Output of MOMIX:

- $\mathcal{L}_{\mathcal{S}}$: list of subboxes $\tilde{B} \subseteq B$ with width $\omega(\tilde{B}) < \delta$

Correctness results

detection of both the efficient and the nondominated set

Input of MOMIX: $\delta > 0$ prescribed precision

Output of MOMIX:

- $\mathcal{L}_{\mathcal{S}}$: list of subboxes $\tilde{B} \subseteq B$ with width $\omega(\tilde{B}) < \delta$
- $\mathcal{L}_{PNS}$: list of upper bounds

Correctness results

detection of both the efficient and the nondominated set

Theorem

Let $E \subseteq B^{g,\mathbb{Z}}$ be the efficient set of (MOMIC).
Let $\mathcal{L}_\mathcal{S}$ be the output of `MOMIX`. Then $\mathcal{L}_\mathcal{S}$ is a cover of E, namely

$$E \subseteq \bigcup_{\tilde{B} \in \mathcal{L}_\mathcal{S}} \tilde{B}$$

Correctness results

detection of both the efficient and the nondominated set

Theorem

Let $E \subseteq B^{g,\mathbb{Z}}$ be the efficient set of (MOMIC).
Let $\mathcal{L}_\mathcal{S}$ be the output of `MOMIX`. *Then $\mathcal{L}_\mathcal{S}$ is a cover of E, namely*

$$E \subseteq \bigcup_{\tilde{B} \in \mathcal{L}_\mathcal{S}} \tilde{B}$$

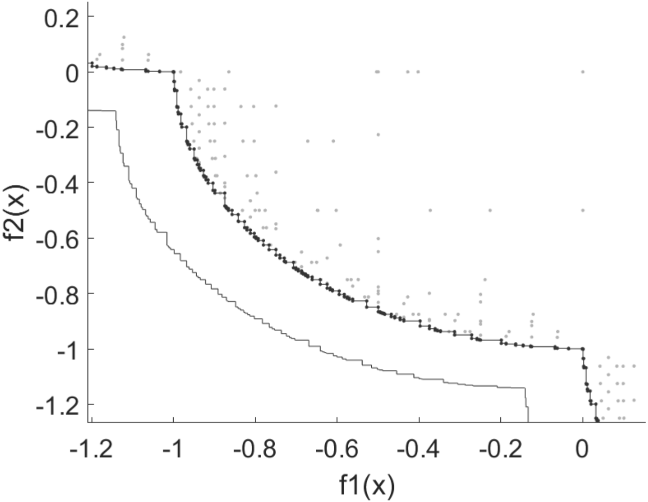
Theorem

Let $\delta > 0$ be the input parameter and $\mathcal{L}_{PNS}$, $\mathcal{L}_\mathcal{S}$ be the output of `MOMIX`. *Let $\mathcal{L}_{LUB}$ be the local upper bound set with respect to $\mathcal{L}_{PNS}$. Then*

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

Example - bi-objective instance with $L\delta = 0.1\sqrt{2}$

part of the image set



Numerical results

- Comparison between MOMIX and $\texttt{MOMIX}_{light}$ on three bi-objective scalable instances with convex quadratic objective functions and constraints

Numerical results

- Comparison between MOMIX and $\texttt{MOMIX}_{light}$ on three bi-objective scalable instances with convex quadratic objective functions and constraints
  - MOMIX and $\texttt{MOMIX}_{light}$ are implemented in MATLAB R2018a
  - within MOMIX we adopted the mixed integer quadratic solver of GUROBI

Numerical results

- Comparison between MOMIX and $\texttt{MOMIX}_{light}$

  on three bi-objective scalable instances with convex quadratic objective functions and constraints

  - MOMIX and $\texttt{MOMIX}_{light}$ are implemented in MATLAB R2018a
  - within MOMIX we adopted the mixed integer quadratic solver of GUROBI

- Comparison between MOMIX and the ε-constraint method on a bi-objective scalable instance

Numerical results

- Comparison between MOMIX and $\texttt{MOMIX}_{light}$ on three bi-objective scalable instances with convex quadratic objective functions and constraints
  - MOMIX and $\texttt{MOMIX}_{light}$ are implemented in MATLAB R2018a
  - within MOMIX we adopted the mixed integer quadratic solver of GUROBI
- Comparison between MOMIX and the ε-constraint method on a bi-objective scalable instance
- Plot of $\mathcal{L}_{PNS}$ obtained for an instance with three objectives

Branching strategies

Let $\tilde{B} = [\tilde{l}, \tilde{u}]$ be a subbox of B

Branching strategies

Let $\tilde{B} = [\tilde{l}, \tilde{u}]$ be a subbox of B

We consider the following two strategies to identify the branching variable $\hat{\imath} \in \{1, \ldots, n\}$:

Branching strategies

Let $\tilde{B} = [\tilde{l}, \tilde{u}]$ be a subbox of B

We consider the following two strategies to identify the branching variable $\hat{\imath} \in \{1, \dots, n\}$:

(br1) $J_1 = \text{argmax}\{\tilde{u}_i - \tilde{l}_i \mid i \in I\}$

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

Choose $\hat{\imath} \in J_1$

Branching strategies

Let $\tilde{B} = [\tilde{l}, \tilde{u}]$ be a subbox of B

We consider the following two strategies 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 $J_1 = \operatorname{argmax}\{\tilde{u}_i - \tilde{l}_i \mid i \in \{1, \ldots, n\} \setminus I\}$
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 $\hat{\imath} \in J_2 \cap I$
Otherwise, choose $\hat{\imath} \in J_2$

Numerical results

Comparison between MOMIX and MOMIX$_{light}$

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

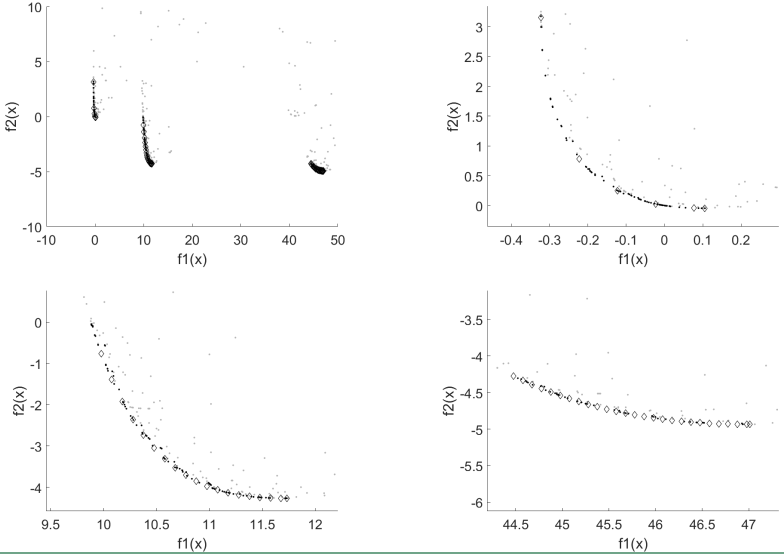
The ε-constraint method minimizes a sequence of parameter-dependent single-objective optimization problems of the following form:

$$\begin{aligned} \min \quad & f_2(x) \\ \text{s.t.} \quad & f_1(x) \leq \varepsilon \\ & x \in B^{g,\mathbb{Z}} \end{aligned} \tag{P$_\varepsilon$}$$

The minima of the functions f_1 and f_2 define the interval where the parameter ε belongs

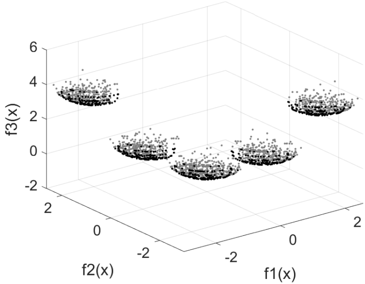
Comparison with the ε-constraint method

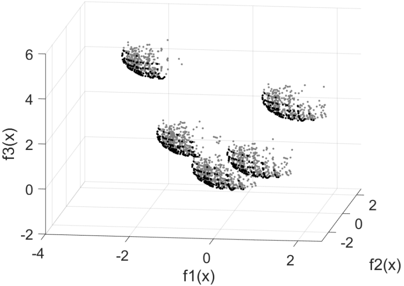
Instance T2 with $|I| = 5$, $n = 7$: $\mathcal{L}_{PNS}$ vs 52 solutions ($\diamond$) computed by ε-constraint method, solving 475 single-objective mixed integer problems



Results on a tri-objective instance

The set $\mathcal{L}_{PNS}$ from two different perspectives





MOMIX summary

- MOMIX is a branch-and-bound method for multiobjective mixed integer convex problems based on the use of properly defined lower bounds

MOMIX summary

- MOMIX is a branch-and-bound method for multiobjective mixed integer convex problems based on the use of properly defined lower bounds
- linear outer approximations of the image set are built in an adaptive way

MOMIX summary

- MOMIX is a branch-and-bound method for multiobjective mixed integer convex problems based on the use of properly defined lower bounds
- linear outer approximations of the image set are built in an adaptive way
- correctness guarantee in terms of detecting both the efficient and the nondominated set of multiobjective mixed integer convex problems according to a prescribed precision

Thanks for your attention!

References

- Boland, N., Charkhgard, H. and Savelsbergh, M. (2015). *A criterion space search algorithm for biobjective integer programming: The balanced box method.* INFORMS Journal on Computing, 27(4), 735–754
- Cacchiani, V. and D'Ambrosio, C. (2017). *A branch-and-bound based heuristic algorithm for convex multi-objective MINLPs.* European Journal of Operational Research, 260, 920-933
- Ehrgott, M., Waters, C., Kasimbeyli and R., Ustun, O. (2009). *Multiobjective programming and multiattribute utility functions in portfolio optimization.* INFOR, 47(1), 31-42
- Löhne, A., Rudloff, B., and Ulus, F. (2014), *Primal and dual approximation algorithms for convex vector optimization problems.* Journal of Global Optimization, 60, 713-736.

References

- Liu, Q., Zhang, C., Zhu, K. and Rao, Y. (2014). *Novel multi-objective resource allocation and activity scheduling for fourth party logistics.* Computers and Operations Research, 44, 42-51
- Klamroth, K., Lacour R. and Vanderpooten, D. (2015). *On the representation of the search region in multi-objective optimization.* European Journal of Operational Research, 245, 767-778
- Niebling, J. and Eichfelder, G. (2019). *A branch-and-bound-based algorithm for nonconvex multi-objective optimization* SIAM Journal Optimization, 29, 794-821
- Pecci F, Abraham E and Stoianov I (2018). *Global optimality bounds for the placement of control valves in water supply networks.* Optimization and Engineering 67(1):201-223, DOI 10.1007/s10589-016-9888-z

References

- Xidonas, P., Mavrotas, G. and Psarras, J. (2010). *Equity portfolio construction and selection using multiobjective mathematical programming*, Journal of Global Optimization, 47, 185-209
- Yenisey, M. M. and Yagmahan, B. (2014). *Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends.* Omega, 45, 119-135
- Yu, L., and Peng, Y. (2014). *Multiple criteria decision making in emergency management.* Computers and Operations Research, 42, 1-124