Multi-Objective Combinatorial Optimization

Anthony Przybylski

University of Nantes, Master 2 ORO

Overview

Overview

1. Scalarization
2. The ε-constraint Method with Adaptive Step
3. The Two Phase Method
4. Bound sets, Branch & Bound

Principle and Properties of Scalarization

Convert multi-objective problem to (parameterized) single objective problem and solve repeatedly with different parameter values

Desirable properties of scalarizations: (Wierzbicki 1984)

- Correctness: Optimal solutions are (weakly) efficient
- Completness: All efficient solutions can be found
- Computability: Scalarization is not harder than single objective version of problem (theory and practice)
- Linearity: Scalarization has linear formulation

Principle and Properties of Scalarization

Convert multi-objective problem to (parameterized) single objective problem and solve repeatedly with different parameter values

Desirable properties of scalarizations: (Wierzbicki 1984)

- Correctness: Optimal solutions are (weakly) efficient
- Completness: All efficient solutions can be found
- Computability: Scalarization is not harder than single objective version of problem (theory and practice)
- Linearity: Scalarization has linear formulation

Principle and Properties of Scalarization

Convert multi-objective problem to (parameterized) single objective problem and solve repeatedly with different parameter values

Desirable properties of scalarizations: (Wierzbicki 1984)

- Correctness: Optimal solutions are (weakly) efficient
- Completness: All efficient solutions can be found
- Computability: Scalarization is not harder than single objective version of problem (theory and practice)
- Linearity: Scalarization has linear formulation

Principle and Properties of Scalarization

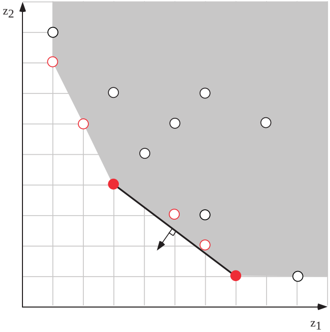
Convert multi-objective problem to (parameterized) single objective problem and solve repeatedly with different parameter values

Desirable properties of scalarizations: (Wierzbicki 1984)

- Correctness: Optimal solutions are (weakly) efficient
- Completness: All efficient solutions can be found
- Computability: Scalarization is not harder than single objective version of problem (theory and practice)
- Linearity: Scalarization has linear formulation

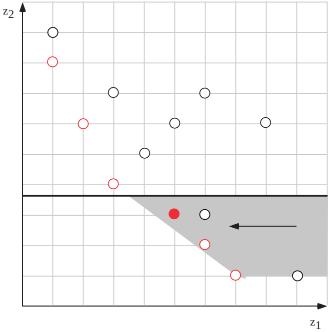
Scalarization Methods

- Weighted sum:
  $\min_{x \in X}\{\lambda^T z(x)\}$
- ε-constraint:
  $\min_{x \in X}\{z_l(x) : z_k(x) \leq \varepsilon_k, k \neq l\}$
- Weighted Chebychev:
  $\min_{x \in X}\left\{\max_{k=1,\ldots,p} \mu_k(z_k(x) - y_k^I)\right\}$



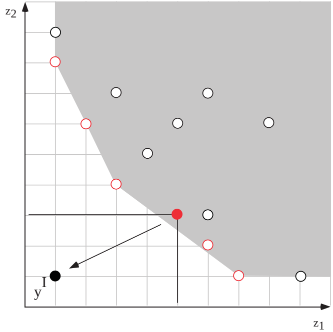
Scalarization Methods

- Weighted sum:
  $$\min_{x \in X}\{\lambda^T z(x)\}$$
- ε-constraint:
  $$\min_{x \in X}\{z_l(x) : z_k(x) \leq \varepsilon_k, k \neq l\}$$
- Weighted Chebychev:
  $$\min_{x \in X}\left\{\max_{k=1,\ldots,p} \mu_k(z_k(x) - y_k^I)\right\}$$



Scalarization Methods

- Weighted sum:
  $\min_{x \in X} \{\lambda^T z(x)\}$
- ε-constraint:
  $\min_{x \in X} \{z_l(x) : z_k(x) \leq \varepsilon_k, k \neq l\}$
- Weighted Chebychev:
  $\min_{x \in X} \left\{ \max_{k=1,\dots,p} \mu_k (z_k(x) - y_k^I) \right\}$



General Scalarization

$$\min_{x \in X} \left\{ \max_{k=1,\ldots,p} [\mu_k(c_k x - \rho_k)] + \sum_{k=1}^{p} [\lambda_k(c_k x - \rho_k)] \right\}$$

$$\text{subject to} \quad c_k x \leq \varepsilon_k \quad k = 1, \ldots, p$$

Includes	ρ	μ	λ	ε
Weighted Sum	0	0	λ	$\varepsilon_k = \infty$, for all k
ε-constraint	0	0	$\lambda_l = 1, \lambda_k = 0, k \neq l$	$\varepsilon_l = \infty, \varepsilon_k, k \neq l$
Chebychev	y^I	μ	0	$\varepsilon_k = \infty$, for all k

Method	Correct	Complete	Computable	Linear
Weighted Sum	+	-	+	+
ε-constraint	+	+	-	+
Chebychev	+	(+)	(-)	+

General Scalarization

$$\min_{x \in X} \left\{ \max_{k=1,\ldots,p} [\mu_k (c_k x - \rho_k)] + \sum_{k=1}^{p} [\lambda_k (c_k x - \rho_k)] \right\}$$

$$\text{subject to} \quad c_k x \leq \varepsilon_k \quad k = 1, \ldots, p$$

Includes	ρ	μ	λ	ε
Weighted Sum	0	0	λ	$\varepsilon_k = \infty$, for all k
ε-constraint	0	0	$\lambda_l = 1, \lambda_k = 0, k \neq l$	$\varepsilon_l = \infty, \varepsilon_k, k \neq l$
Chebychev	y^I	μ	0	$\varepsilon_k = \infty$, for all k

Method	Correct	Complete	Computable	Linear
Weighted Sum	+	−	+	+
ε-constraint	+	+	−	+
Chebychev	+	(+)	(-)	+

General Scalarization

$$\min_{x \in X} \left\{ \max_{k=1,\ldots,p} [\mu_k(c_k x - \rho_k)] + \sum_{k=1}^{p} [\lambda_k(c_k x - \rho_k)] \right\}$$
$$\text{subject to} \quad c_k x \leq \varepsilon_k \quad k = 1, \ldots, p$$

Includes	ρ	μ	λ	ε
Weighted Sum	0	0	λ	$\varepsilon_k = \infty$, for all k
ε-constraint	0	0	$\lambda_l = 1, \lambda_k = 0, k \neq l$	$\varepsilon_l = \infty, \varepsilon_k, k \neq l$
Chebychev	y^I	μ	0	$\varepsilon_k = \infty$, for all k

Method	Correct	Complete	Computable	Linear
Weighted Sum	+	−	+	+
ε-constraint	+	+	−	+
Chebychev	+	(+)	(-)	+

General Scalarization

$$\min_{x \in X} \quad \left\{ \max_{k=1,\ldots,p} [\mu_k (c_k x - \rho_k)] + \sum_{k=1}^{p} [\lambda_k (c_k x - \rho_k)] \right\}$$
$$\text{subject to} \quad c_k x \le \varepsilon_k \qquad k = 1, \ldots, p$$

Includes	ρ	μ	λ	ε
Weighted Sum	0	0	λ	$\varepsilon_k = \infty$, for all k
ε-constraint	0	0	$\lambda_l = 1, \lambda_k = 0, k \neq l$	$\varepsilon_l = \infty, \varepsilon_k, k \neq l$
Chebychev	y^I	μ	0	$\varepsilon_k = \infty$, for all k

Method	Correct	Complete	Computable	Linear
Weighted Sum	+	−	+	+
ε-constraint	+	+	−	+
Chebychev	+	(+)	(−)	−

General Scalarization

$$\begin{array}{ll} \min\limits_{x \in X} & \left\{ \max\limits_{k=1,\ldots,p} [\mu_k(c_k x - \rho_k)] + \sum\limits_{k=1}^{p} [\lambda_k(c_k x - \rho_k)] \right\} \\ \text{subject to} & c_k x \leq \varepsilon_k \quad k = 1, \ldots, p \end{array}$$

Includes	ρ	μ	λ	ε
Weighted Sum	0	0	λ	$\varepsilon_k = \infty$, for all k
ε-constraint	0	0	$\lambda_l = 1, \lambda_k = 0, k \neq l$	$\varepsilon_l = \infty, \varepsilon_k, k \neq l$
Chebychev	y^I	μ	0	$\varepsilon_k = \infty$, for all k

Method	Correct	Complete	Computable	Linear
Weighted Sum	+	−	+	+
ε-constraint	+	+	−	+
Chebychev	+	(+)	(-)	+

General Scalarization

$$\min_{x \in X} \left\{ \max_{k=1,\ldots,p} [\mu_k (c_k x - \rho_k)] + \sum_{k=1}^{p} [\lambda_k (c_k x - \rho_k)] \right\}$$
$$\text{subject to} \quad c_k x \leq \varepsilon_k \qquad k = 1, \ldots, p$$

Includes	ρ	μ	λ	ε
Weighted Sum	0	0	λ	$\varepsilon_k = \infty$, for all k
ε-constraint	0	0	$\lambda_l = 1, \lambda_k = 0, k \neq l$	$\varepsilon_l = \infty, \varepsilon_k, k \neq l$
Chebychev	y^I	μ	0	$\varepsilon_k = \infty$, for all k

Method	Correct	Complete	Computable	Linear
Weighted Sum	+	-	+	+
ε-constraint	+	+	-	+
Chebychev	+	(+)	(-)	+

General Scalarization

$$\min_{x \in X} \left\{ \max_{k=1,\ldots,p} [\mu_k(c_k x - \rho_k)] + \sum_{k=1}^{p} [\lambda_k(c_k x - \rho_k)] \right\}$$
$$\text{subject to} \quad c_k x \leq \varepsilon_k \qquad k = 1, \ldots, p$$

Includes	ρ	μ	λ	ε
Weighted Sum	0	0	λ	$\varepsilon_k = \infty$, for all k
ε-constraint	0	0	$\lambda_l = 1, \lambda_k = 0, k \neq l$	$\varepsilon_l = \infty, \varepsilon_k, k \neq l$
Chebychev	y^I	μ	0	$\varepsilon_k = \infty$, for all k

Method	Correct	Complete	Computable	Linear
Weighted Sum	+	-	+	+
ε-constraint	+	+	-	+
Chebychev	+	(+)	(-)	+

General Scalarization

$$\min_{x \in X} \left\{ \max_{k=1,\dots,p} [\mu_k(c_k x - \rho_k)] + \sum_{k=1}^{p} [\lambda_k(c_k x - \rho_k)] \right\}$$
$$\text{subject to} \quad c_k x \leq \varepsilon_k \quad k = 1, \dots, p$$

Includes	ρ	μ	λ	ε
Weighted Sum	0	0	λ	$\varepsilon_k = \infty$, for all k
ε-constraint	0	0	$\lambda_l = 1, \lambda_k = 0, k \neq l$	$\varepsilon_l = \infty, \varepsilon_k, k \neq l$
Chebychev	y^I	μ	0	$\varepsilon_k = \infty$, for all k

Method	Correct	Complete	Computable	Linear
Weighted Sum	+	-	+	+
ε-constraint	+	+	-	+
Chebychev	+	(+)	(-)	+

General Scalarization

$$\min_{x \in X} \left\{ \max_{k=1,\dots,p} [\mu_k(c_k x - \rho_k)] + \sum_{k=1}^{p} [\lambda_k (c_k x - \rho_k)] \right\}$$

$$\text{subject to} \quad c_k x \leq \varepsilon_k \quad k = 1, \dots, p$$

Includes	ρ	μ	λ	ε
Weighted Sum	0	0	λ	$\varepsilon_k = \infty$, for all k
ε-constraint	0	0	$\lambda_l = 1, \lambda_k = 0, k \neq l$	$\varepsilon_l = \infty, \varepsilon_k, k \neq l$
Chebychev	y^I	μ	0	$\varepsilon_k = \infty$, for all k

Method	Correct	Complete	Computable	Linear
Weighted Sum	+	-	+	+
ε-constraint	+	+	-	+
Chebychev	+	(+)	(-)	+

General scalarization

Theorem (Ehrgott 2005)

1. *The general scalarization is $\mathcal{NP}$-hard*
2. *An optimal solution of the Lagrangian dual of the linearized general scalarization is a supported efficient solution*

Notes

- The general scalarization includes other particular scalarizations
- Given a problem the single objective case is $\mathcal{NP}$-hard, a scalarization (like ε-constraint) of this problem is also $\mathcal{NP}$-hard.
  This does not imply a same practical difficulty

General scalarization

Theorem (Ehrgott 2005)

1. *The general scalarization is $\mathcal{NP}$-hard*
2. *An optimal solution of the Lagrangian dual of the linearized general scalarization is a supported efficient solution*

Notes

- The general scalarization includes other particular scalarizations
- Given a problem the single objective case is $\mathcal{NP}$-hard, a scalarization (like ε-constraint) of this problem is also $\mathcal{NP}$-hard.
  This does not imply a same practical difficulty

General scalarization

Theorem (Ehrgott 2005)

1. *The general scalarization is $\mathcal{NP}$-hard*
2. *An optimal solution of the Lagrangian dual of the linearized general scalarization is a supported efficient solution*

Notes

- The general scalarization includes other particular scalarizations
- Given a problem the single objective case is $\mathcal{NP}$-hard, a scalarization (like ε-constraint) of this problem is also $\mathcal{NP}$-hard.
  This does not imply a same practical difficulty

Overview

The ε-constraint Method

- Given an instance of a MOCO problem, a complete set can be computed using ε-constraint method
- All efficient solution $\bar{x}$ is an optimal solution of a problem

$$\min_{x \in X} \{z_l(x) : z_k(x) \leq \varepsilon_k, k \neq l\} \tag{1}$$

- A suitable parameter to find $\bar{x}$ (or an equivalent solution) by optimization of (1) could be $\varepsilon = z(\bar{x})$
- However, $\bar{x}$ is not know
- Determination of appropriate ε value?

The ε-constraint Method

- Given an instance of a MOCO problem, a complete set can be computed using ε-constraint method
- All efficient solution $\bar{x}$ is an optimal solution of a problem

$$\min_{x \in X} \{z_l(x) : z_k(x) \leq \varepsilon_k, k \neq l\} \tag{1}$$

- A suitable parameter to find $\bar{x}$ (or an equivalent solution) by optimization of (1) could be $\varepsilon = z(\bar{x})$
- However, $\bar{x}$ is not know
- Determination of appropriate ε value?

The ε-constraint Method

- Given an instance of a MOCO problem, a complete set can be computed using ε-constraint method
- All efficient solution $\bar{x}$ is an optimal solution of a problem

$$\min_{x \in X}\{z_l(x) : z_k(x) \leq \varepsilon_k, k \neq l\} \tag{1}$$

- A suitable parameter to find $\bar{x}$ (or an equivalent solution) by optimization of (1) could be $\varepsilon = z(\bar{x})$
- However, $\bar{x}$ is not know
- Determination of appropriate ε value?

The Bi-objective case

- Use of the "natural order" of non-dominated points in the bi-objective case:
  Let y^1, y^2 be two nondominated points with $y^1 \neq y^2$ then ($y_1^1 < y_1^2$ and $y_2^1 > y_2^2$), or ($y_1^1 > y_1^2$ and $y_2^1 < y_2^2$)
- With two objectives, the ε-constraint scalarization is

$$\min_{x \in X}\{z_1(x) : z_2(x) \leq \varepsilon_1\} \tag{2}$$

- Given a nondominated point y^i, the next (weakly) non-dominated point w.r.t. to z_1 (if is exists) can be found by solving

$$\min_{x \in X}\{z_1(x) : z_2(x) < y_2^i\}$$

  This is not an instance of (2)!!!

The Bi-objective case

- Use of the "natural order" of non-dominated points in the bi-objective case:
  Let y^1, y^2 be two nondominated points with $y^1 \neq y^2$ then $(y_1^1 < y_1^2$ and $y_2^1 > y_2^2)$, or $(y_1^1 > y_1^2$ and $y_2^1 < y_2^2)$
- With two objectives, the ε-constraint scalarization is

$$\min_{x \in X}\{z_1(x) : z_2(x) \leq \varepsilon_1\} \tag{2}$$

- Given a nondominated point y^i, the next (weakly) non-dominated point w.r.t. to z_1 (if is exists) can be found by solving

$$\min_{x \in X}\{z_1(x) : z_2(x) < y_2^i\}$$

This is not an instance of (2)!!!

The Bi-objective case

- Use of the "natural order" of non-dominated points in the bi-objective case:
  Let y^1, y^2 be two nondominated points with $y^1 \neq y^2$ then $(y_1^1 < y_1^2$ and $y_2^1 > y_2^2)$, or $(y_1^1 > y_1^2$ and $y_2^1 < y_2^2)$
- With two objectives, the ε-constraint scalarization is

$$\min_{x \in X}\{z_1(x) : z_2(x) \leq \varepsilon_1\} \tag{2}$$

- Given a nondominated point y^i, the next (weakly) non-dominated point w.r. to z_1 (if is exists) can be found by solving

$$\min_{x \in X}\{z_1(x) : z_2(x) < y_2^i\}$$

  This is not an instance of (2)!!!

The Bi-objective case

- Use of the "natural order" of non-dominated points in the bi-objective case:
  Let y^1, y^2 be two nondominated points with $y^1 \neq y^2$ then $(y_1^1 < y_1^2$ and $y_2^1 > y_2^2)$, or $(y_1^1 > y_1^2$ and $y_2^1 < y_2^2)$
- With two objectives, the ε-constraint scalarization is

$$\min_{x \in X} \{z_1(x) : z_2(x) \leq \varepsilon_1\} \tag{2}$$

- Given a nondominated point y^i, the next (weakly) non-dominated point w.r. to z_1 (if is exists) can be found by solving

$$\min_{x \in X} \{z_1(x) : z_2(x) \leq y_2^i - \epsilon\}$$

  where $\epsilon > 0$ is as small as possible!!!

Algorithm (ε-constraint with adaptive step)

1. Initialization:
   - Determine x^1 a lexicographic optimal solution for $z^{(1,2)}$
   - $\tilde{X} \leftarrow \{x^1\}$
   - $\varepsilon_1 \leftarrow z_2(x^1) - \epsilon$
2. While problem (2) is feasible do
   - Let $\hat{x}$ be an optimal solution of (2)
   - $\tilde{X} \leftarrow \tilde{X} \cup \{\hat{x}\}$
   - $\varepsilon_1 \leftarrow z_2(\hat{x}) - \epsilon$
3. Filter dominated solutions in $\tilde{X}$

Output: $\tilde{X}$ contains one solution for each nondominated point, i.e. a minimal complete set X_{E_m}

Algorithm (ε-constraint with adaptive step)

1. Initialization:
   - Determine x^1 a lexicographic optimal solution for $z^{(1,2)}$
   - $\tilde{X} \leftarrow \{x^1\}$
   - $\varepsilon_1 \leftarrow z_2(x^1) - \epsilon$
2. While problem (2) is feasible do
   - Let $\hat{x}$ be an optimal solution of (2)
   - $\tilde{X} \leftarrow \tilde{X} \cup \{\hat{x}\}$
   - $\varepsilon_1 \leftarrow z_2(\hat{x}) - \epsilon$
3. Filter dominated solutions in $\tilde{X}$

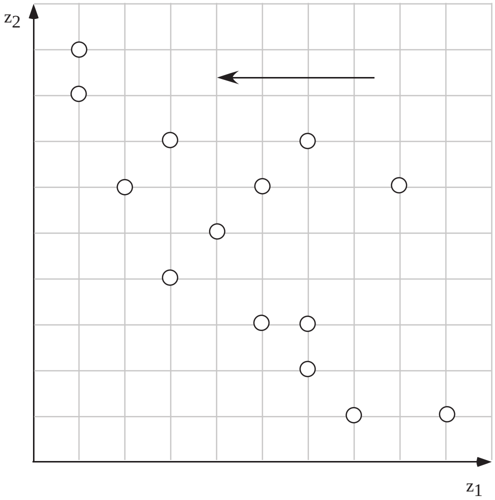
Output: $\tilde{X}$ contains one solution for each nondominated point, i.e. a minimal complete set X_{E_m}

Algorithm (ε-constraint with adaptive step)

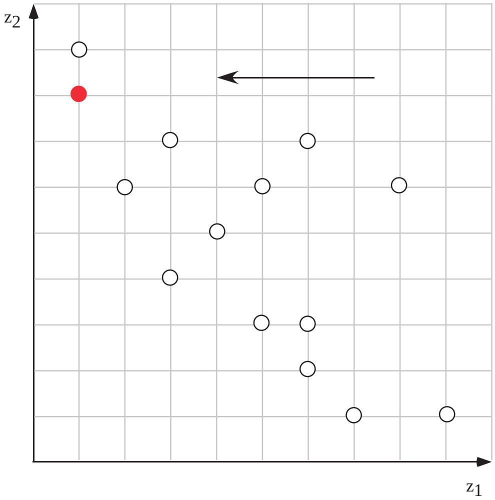
1. Initialization:
   - Determine x^1 a lexicographic optimal solution for $z^{(1,2)}$
   - $\tilde{X} \leftarrow \{x^1\}$
   - $\varepsilon_1 \leftarrow z_2(x^1) - \epsilon$
2. While problem (2) is feasible do
   - Let $\hat{x}$ be an optimal solution of (2)
   - $\tilde{X} \leftarrow \tilde{X} \cup \{\hat{x}\}$
   - $\varepsilon_1 \leftarrow z_2(\hat{x}) - \epsilon$
3. Filter dominated solutions in $\tilde{X}$

Output: $\tilde{X}$ contains one solution for each nondominated point, i.e. a minimal complete set X_{E_m}

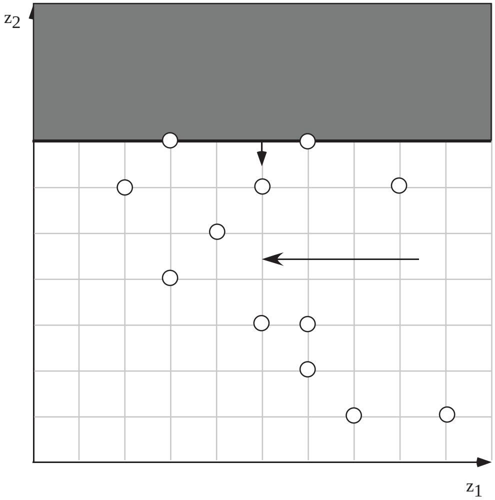
Illustration



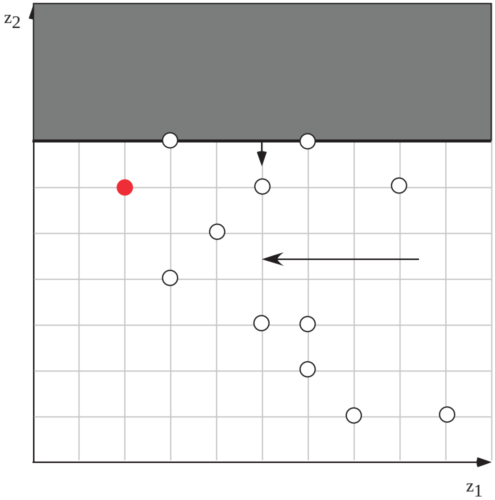
Illustration



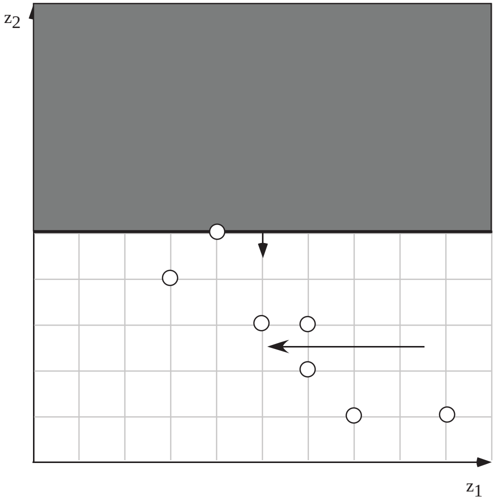
Illustration



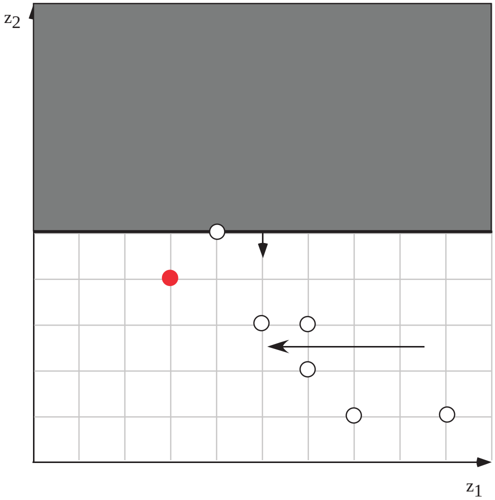
Illustration



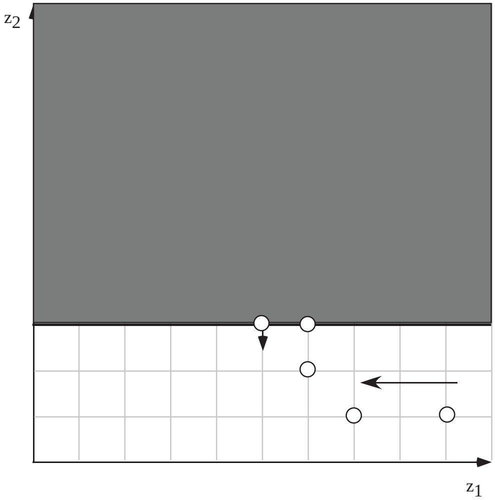
Illustration



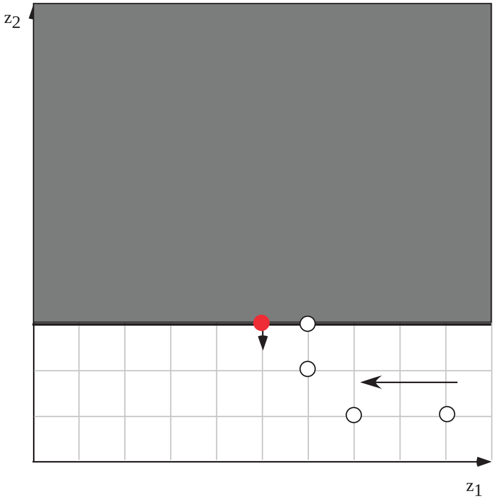
Illustration



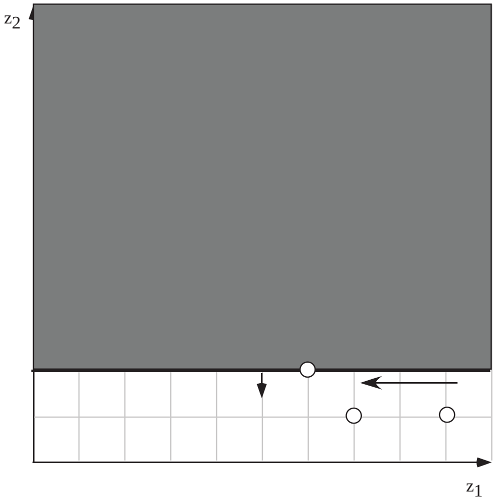
Illustration



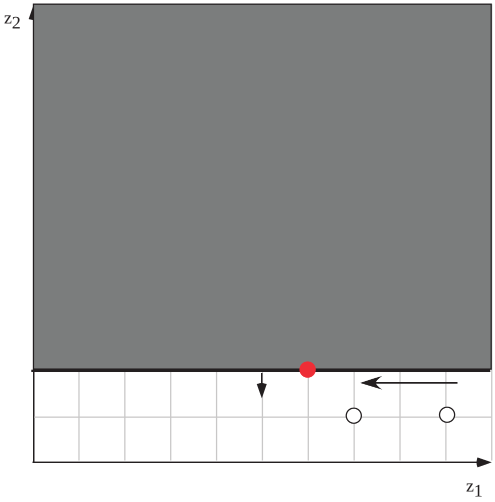
Illustration



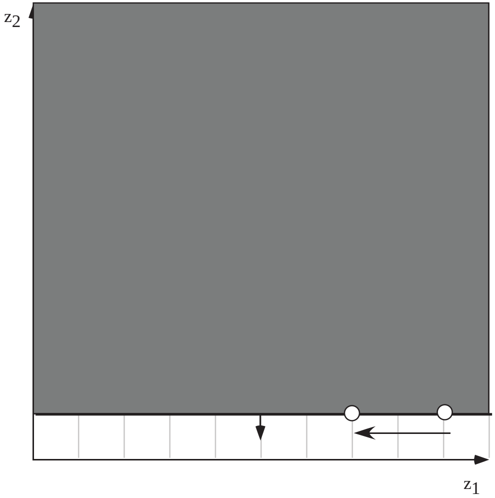
Illustration



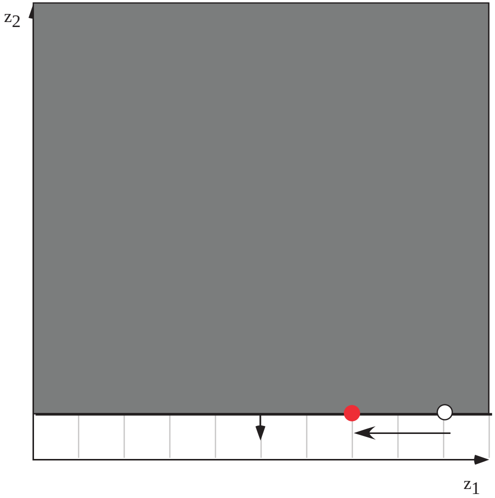
Illustration



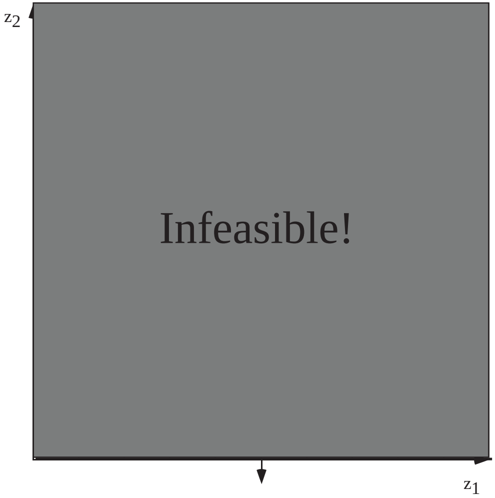
Illustration



Illustration



Illustration



Some Notes

- In order to avoid to filter dominated solutions (step 3), (2) is sometimes replaced by

$$\text{lexmin}_{x \in X}\{z_1(x) : z_2(x) \leq \varepsilon_1\} \tag{3}$$

- However, a lexicographic optimization may require several steps
  - Solve $\min_{x \in X} z_1(x)$ and obtain a solution x^1
  - Solve $\min_{x \in X} z_2(x)$ with the additional constraint $z_1(x) \leq z_1(x^1)$
- It seems to clarify the algorithm but could lead to an inefficient implementation

Some Notes

- In order to avoid to filter dominated solutions (step 3), (2) is sometimes replaced by

$$\text{lexmin}_{x \in X} \{z_1(x) : z_2(x) \leq \varepsilon_1\} \tag{3}$$

- However, a lexicographic optimization may require several steps
  - Solve $\min_{x \in X} z_1(x)$ and obtain a solution x^1
  - Solve $\min_{x \in X} z_2(x)$ with the additional constraint $z_1(x) \leq z_1(x^1)$
- It seems to clarify the algorithm but could lead to an inefficient implementation

Some Notes

- In order to avoid to filter dominated solutions (step 3), (2) is sometimes replaced by

$$\text{lexmin}_{x \in X}\{z_1(x) : z_2(x) \leq \varepsilon_1\} \tag{3}$$

- However, a lexicographic optimization may require several steps
    - Solve $\min_{x \in X} z_1(x)$ and obtain a solution x^1
    - Solve $\min_{x \in X} z_2(x)$ with the additional constraint $z_1(x) \leq z_1(x^1)$
- It seems to clarify the algorithm but could lead to an inefficient implementation

Some Notes

- In order to avoid to filter dominated solutions (step 3), (2) is sometimes replaced by

$$\text{lexmin}_{x \in X}\{z_1(x) : z_2(x) \leq \varepsilon_1\} \tag{3}$$

- However, a lexicographic optimization may require several steps
  - Solve $\min_{x \in X} z_1(x)$ and obtain a solution x^1
  - Solve $\min_{x \in X} z_2(x)$ with the additional constraint $z_1(x) \leq z_1(x^1)$
- It seems to clarify the algorithm but could lead to an inefficient implementation

Some Notes

- The choice of the step ϵ can be difficult in a general context
  - With a step too large, a nondominated point may be "jumped"
  - A step too small may cause numerical imprecisions in practice
- In a MOCO problem, $C \in \mathbb{Z}^{2\times n} \Longrightarrow Y \subset \mathbb{Z}^2$
  Consequently, the step ϵ can be fixed to 1
- Using the integrity of cost vectors, lexicographic optimization can be solved with a weighted sum scalarization

Some Notes

- The choice of the step ϵ can be difficult in a general context
  - With a step too large, a nondominated point may be "jumped"
  - A step too small may cause numerical imprecisions in practice
- In a MOCO problem, $C \in \mathbb{Z}^{2\times n} \Longrightarrow Y \subset \mathbb{Z}^2$
  Consequently, the step ϵ can be fixed to 1
- Using the integrity of cost vectors, lexicographic optimization can be solved with a weighted sum scalarization

Some Notes

- The choice of the step ϵ can be difficult in a general context
  - With a step too large, a nondominated point may be "jumped"
  - A step too small may cause numerical imprecisions in practice
- In a MOCO problem, $C \in \mathbb{Z}^{2\times n} \Longrightarrow Y \subset \mathbb{Z}^2$
  Consequently, the step ϵ can be fixed to 1
- Using the integrity of cost vectors, lexicographic optimization can be solved with a weighted sum scalarization

Some Notes

- The choice of the step ϵ can be difficult in a general context
  - With a step too large, a nondominated point may be "jumped"
  - A step too small may cause numerical imprecisions in practice
- In a MOCO problem, $C \in \mathbb{Z}^{2\times n} \Longrightarrow Y \subset \mathbb{Z}^2$
  Consequently, the step ϵ can be fixed to 1
- Using the integrity of cost vectors, lexicographic optimization can be solved with a weighted sum scalarization

Some Notes

- The choice of the step ϵ can be difficult in a general context
  - With a step too large, a nondominated point may be "jumped"
  - A step too small may cause numerical imprecisions in practice
- In a MOCO problem, $C \in \mathbb{Z}^{2\times n} \Longrightarrow Y \subset \mathbb{Z}^2$
  Consequently, the step ϵ can be fixed to 1
- Using the integrity of cost vectors, lexicographic optimization can be solved with a weighted sum scalarization

Bottleneck Objective

- ε-constraint method with adaptive step is a powerful method to solve $(1-\sum, 1-\max)$ and $(2-\max)$ MOCO problems
- It is judicious to convert a bottleneck objective (z_2 here) to a constraint

$$\min_{x \in X} \left\{ z_1(x) : \max_{i=1,\dots,n} c_i^2 x^i \leq \epsilon_2 \right\}$$

is equivalent to

$$\min_{x \in X} z_1(x)$$

with a modified cost vector c'^1 where
  - If $c_i^2 > \epsilon_2$ then $c'^1_i := \infty$
  - Else $c'^1_i := c_i^1$
- The ε-constraint scalarization is an instance of the single-objective problem with the same (practical and theoretical) difficulty

Bottleneck Objective

- ε-constraint method with adaptive step is a powerful method to solve $(1-\sum, 1-\max)$ and $(2-\max)$ MOCO problems
- It is judicious to convert a bottleneck objective (z_2 here) to a constraint

$$\min_{x \in X} \left\{ z_1(x) : \max_{i=1,\dots,n} c_i^2 x^i \leq \epsilon_2 \right\}$$

  is equivalent to

$$\min_{x \in X} z_1(x)$$

  with a modified cost vector ${c'}^1$ where
  - If $c_i^2 > \epsilon_2$ then ${c'}_i^1 := \infty$
  - Else ${c'}_i^1 := c_i^1$
- The ε-constraint scalarization is an instance of the single-objective problem with the same (practical and theoretical) difficulty

Bottleneck Objective

- ε-constraint method with adaptive step is a powerful method to solve $(1-\sum, 1-\max)$ and $(2-\max)$ MOCO problems
- It is judicious to convert a bottleneck objective (z_2 here) to a constraint

$$\min_{x \in X} \left\{ z_1(x) : \max_{i=1,\dots,n} c_i^2 x^i \leq \epsilon_2 \right\}$$

  is equivalent to

$$\min_{x \in X} z_1(x)$$

  with a modified cost vector ${c'}^1$ where
  - If $c_i^2 > \epsilon_2$ then ${c'}_i^1 := \infty$
  - Else ${c'}_i^1 := c_i^1$
- The ε-constraint scalarization is an instance of the single-objective problem with the same (practical and theoretical) difficulty

Bottleneck Objective

- ε-constraint method with adaptive step is a powerful method to solve $(1-\sum, 1-\max)$ and $(2-\max)$ MOCO problems
- It is judicious to convert a bottleneck objective (z_2 here) to a constraint

$$\min_{x \in X} \left\{ z_1(x) : \max_{i=1,\dots,n} c_i^2 x^i \leq \epsilon_2 \right\}$$

  is equivalent to

$$\min_{x \in X} z_1(x)$$

  with a modified cost vector ${c'}^1$ where
    - If $c_i^2 > \epsilon_2$ then ${c'}_i^1 := \infty$
    - Else ${c'}_i^1 := c_i^1$
- The ε-constraint scalarization is an instance of the single-objective problem with the same (practical and theoretical) difficulty

Conclusion for the Bi-objective Case

- ε-constraint with adaptive step is a generic method to compute a set X_{E_m} of an instance of a MOCO problem with two objectives (or a bounded bi-objective integer programme)
- The method is particularly efficient in presence of a bottleneck objective
- With two sum objectives, the constraint structure of the problem is modified
- A MIP solver is generally required and the scalarization is often difficult to solve $\Longrightarrow$ The size of solved instance is generally moderate
- However, the method can be implemented easily and rapidly $\Longrightarrow$ Interesting for a first feedback about a problem

Conclusion for the Bi-objective Case

- ε-constraint with adaptive step is a generic method to compute a set X_{E_m} of an instance of a MOCO problem with two objectives (or a bounded bi-objective integer programme)
- The method is particularly efficient in presence of a bottleneck objective
- With two sum objectives, the constraint structure of the problem is modified
- A MIP solver is generally required and the scalarization is often difficult to solve $\Longrightarrow$ The size of solved instance is generally moderate
- However, the method can be implemented easily and rapidly $\Longrightarrow$ Interesting for a first feedback about a problem

Conclusion for the Bi-objective Case

- ε-constraint with adaptive step is a generic method to compute a set X_{E_m} of an instance of a MOCO problem with two objectives (or a bounded bi-objective integer programme)
- The method is particularly efficient in presence of a bottleneck objective
- With two sum objectives, the constraint structure of the problem is modified
- A MIP solver is generally required and the scalarization is often difficult to solve $\Longrightarrow$ The size of solved instance is generally moderate
- However, the method can be implemented easily and rapidly $\Longrightarrow$ Interesting for a first feedback about a problem

Conclusion for the Bi-objective Case

- ε-constraint with adaptive step is a generic method to compute a set X_{E_m} of an instance of a MOCO problem with two objectives (or a bounded bi-objective integer programme)
- The method is particularly efficient in presence of a bottleneck objective
- With two sum objectives, the constraint structure of the problem is modified
- A MIP solver is generally required and the scalarization is often difficult to solve $\Longrightarrow$ The size of solved instance is generally moderate
- However, the method can be implemented easily and rapidly $\Longrightarrow$ Interesting for a first feedback about a problem

Multi-objective case: ε-constraint (like) with adaptive step

- M. Laumanns, L. Thiele, E. Zitzler. An adaptive scheme to generate the Pareto front based on the ε-constraint method. In: J. Branke. K. Deb, K. Miettinen, R.E. Steuer (eds.) Practical Approaches to Multi-Objective Optimization, number 04461 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005. Internationales Begegnungs und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.
- M. Ozlen, M. Azizoglu. Multi-objective integer programming: a general approach for generating all non-dominated solutions. Eur. J. Oper. Res. 199, 25-35 (2009)
- M. Ozlen, B.A. Burton, C.A.G MacRae. Multi-objective integer programming: an improved recursive algorithm. J. Optim. Theory and Appl. 160(2), 470-482 (2014)
- B. Lokman, M. Köksalan. Finding all nondominated points of multi-objective integer programs. J. Global Optim. 57, 347-365 (2013)
- G. Kirlik, S. Sayin. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. Eur. J. Oper. Res. 232, 479-488 (2014)
- K. Dächert, K. Klamroth. A linear bound on the number of scalarizations needed to solve tricriteria optimization problems. J. Global Optim. 62, 643-676 (2014)

Overview

The Two Phase Method

- General solving scheme for multi-objective combinatorial optimization problems (Ulungu and Teghem, 1995)
- Observation: There exists efficient algorithm for single-objective combinatorial optimization problems
- Idea: Intensively use these algorithms
- Consequence: The constraint structure cannot be modified, the only usable scalarization is the weighted sum

The Two Phase Method

- General solving scheme for multi-objective combinatorial optimization problems (Ulungu and Teghem, 1995)
- Observation: There exists efficient algorithm for single-objective combinatorial optimization problems
- Idea: Intensively use these algorithms
- Consequence: The constraint structure cannot be modified, the only usable scalarization is the weighted sum

The Two Phase Method

- General solving scheme for multi-objective combinatorial optimization problems (Ulungu and Teghem, 1995)
- Observation: There exists efficient algorithm for single-objective combinatorial optimization problems
- Idea: Intensively use these algorithms
- Consequence: The constraint structure cannot be modified, the only usable scalarization is the weighted sum

Phase 1: Compute Supported Solutions

- Using weighted sum scalarization, we solve

$$\min_{x \in X} \{\lambda_1 z_1(x) + \lambda_2 z_2(x)\} \tag{4}$$

  where $\lambda \in \mathbb{R}^2_>$
- Only supported solutions can be found by optimization of (4)
- How can we choose appropriate parameters λ to find *all* supported solutions ?
- By dichotomy using the "natural order" of nondominated points

Phase 1: Compute Supported Solutions

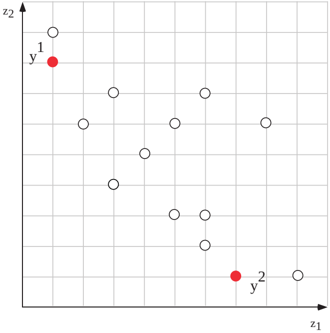
- Using weighted sum scalarization, we solve

$$\min_{x \in X} \{\lambda_1 z_1(x) + \lambda_2 z_2(x)\} \tag{4}$$

  where $\lambda \in \mathbb{R}^2_{>}$
- Only supported solutions can be found by optimization of (4)
- How can we choose appropriate parameters λ to find *all* supported solutions ?
- By dichotomy using the "natural order" of nondominated points

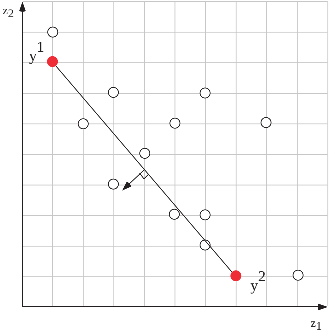
Dichotomic Method

- Given two supported solutions x^1, x^2 with $y^1 = z(x^1)$ and $y^2 = z(x^2)$ such that $(y_1^1 < y_1^2$ and $y_2^1 > y_2^2)$
- Let $\lambda = (\lambda_1, \lambda_2)$ be the weight defined by $\lambda_1 = y_2^1 - y_2^2$ and $\lambda_2 = y_1^2 - y_1^1$
- $\lambda_1 y_1^1 + \lambda_2 y_2^1 = \lambda_1 y_1^2 + \lambda_2 y_2^2$ λ defines the normal to the line segment connecting y^1 and y^2



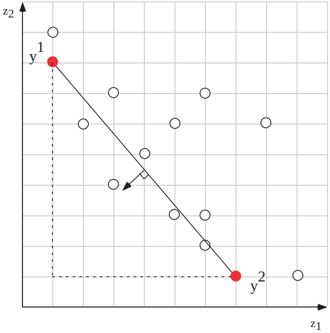
Dichotomic Method

- Given two supported solutions x^1, x^2 with $y^1 = z(x^1)$ and $y^2 = z(x^2)$ such that $(y_1^1 < y_1^2$ and $y_2^1 > y_2^2)$
- Let $\lambda = (\lambda_1, \lambda_2)$ be the weight defined by $\lambda_1 = y_2^1 - y_2^2$ and $\lambda_2 = y_1^2 - y_1^1$
- $\lambda_1 y_1^1 + \lambda_2 y_2^1 = \lambda_1 y_1^2 + \lambda_2 y_2^2$ λ defines the normal to the line segment connecting y^1 and y^2



Dichotomic Method

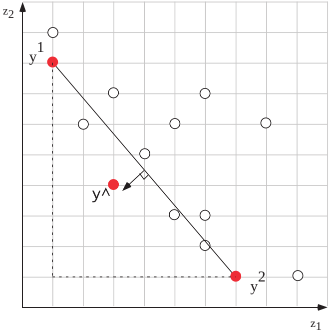
- Let $\hat{x}$ be an optimal solution of (4) with $\hat{y} = z(\hat{x})$
- $\hat{y}$ is necessarily located "between" y^1 and y^2
- Suppose $\lambda_1 \hat{y}_1 + \lambda_2 \hat{y}_2 < \lambda_1 y_1^2 + \lambda_2 y_2^2$ Then $\hat{y}$ is located below the line segment connecting y^1 and y^2



Question: Is there supported solutions between y^1 and $\hat{y}$, $\hat{y}$ and y^2? Recursive solution to check it

Dichotomic Method

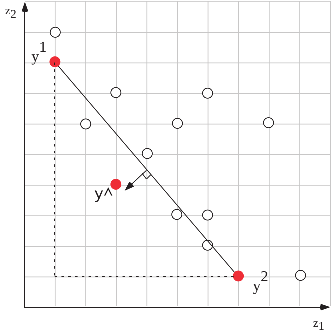
- Let $\hat{x}$ be an optimal solution of (4) with $\hat{y} = z(\hat{x})$
- $\hat{y}$ is necessarily located "between" y^1 and y^2
- Suppose $\lambda_1 \hat{y}_1 + \lambda_2 \hat{y}_2 < \lambda_1 y_1^2 + \lambda_2 y_2^2$ Then $\hat{y}$ is located below the line segment connecting y^1 and y^2



Question: Is there supported solutions between y^1 and $\hat{y}$, $\hat{y}$ and y^2? Recursive solution to check it

Dichotomic Method

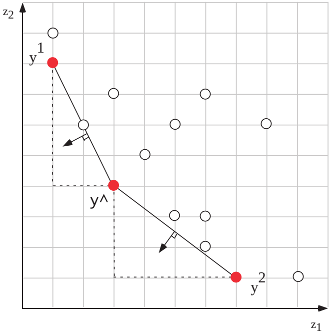
- Let $\hat{x}$ be an optimal solution of (4) with $\hat{y} = z(\hat{x})$
- $\hat{y}$ is necessarily located "between" y^1 and y^2
- Suppose $\lambda_1 \hat{y}_1 + \lambda_2 \hat{y}_2 < \lambda_1 y_1^2 + \lambda_2 y_2^2$
  Then $\hat{y}$ is located below the line segment connecting y^1 and y^2



Question: Is there supported solutions between y^1 and $\hat{y}$, $\hat{y}$ and y^2? Recursive solution to check it

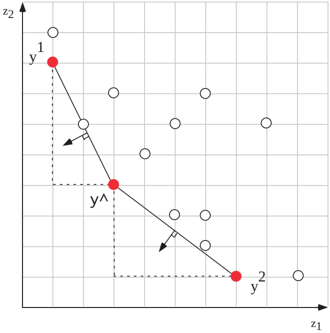
Dichotomic Method

- Let $\lambda = (\lambda_1, \lambda_2)$ be the weight vector defined by y^1 and $\hat{y}$
- Let $\bar{x}$ be an optimal solution of (4) with $\bar{y} = z(\bar{x})$
- Suppose $\lambda_1 \bar{y}_1 + \lambda_2 \bar{y}_2 = \lambda_1 \hat{y}_1 + \lambda_2 \hat{y}_2$ Is the solution/point necessarily new?
- We can obtain an equivalent solution to x^1 or $\bar{x}$ or a non-extreme supported solution



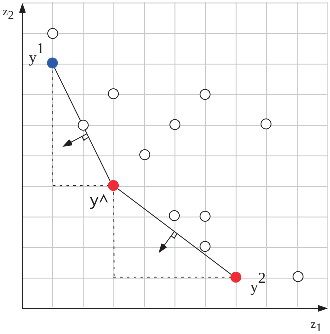
Dichotomic Method

- Let $\lambda = (\lambda_1, \lambda_2)$ be the weight vector defined by y^1 and $\hat{y}$
- Let $\bar{x}$ be an optimal solution of (4) with $\bar{y} = z(\bar{x})$
- Suppose $\lambda_1 \bar{y}_1 + \lambda_2 \bar{y}_2 = \lambda_1 \hat{y}_1 + \lambda_2 \hat{y}_2$ Is the solution/point necessarily new?
- We can obtain an equivalent solution to x^1 or $\bar{x}$ or a non-extreme supported solution



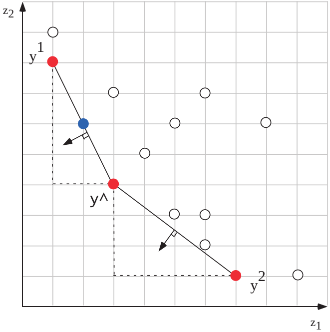
Dichotomic Method

- Let $\lambda = (\lambda_1, \lambda_2)$ be the weight vector defined by y^1 and $\hat{y}$
- Let $\bar{x}$ be an optimal solution of (4) with $\bar{y} = z(\bar{x})$
- Suppose
  $\lambda_1 \bar{y}_1 + \lambda_2 \bar{y}_2 = \lambda_1 \hat{y}_1 + \lambda_2 \hat{y}_2$
  Is the solution/point necessarily new?
- We can obtain an equivalent solution to x^1 or $\bar{x}$ or a non-extreme supported solution



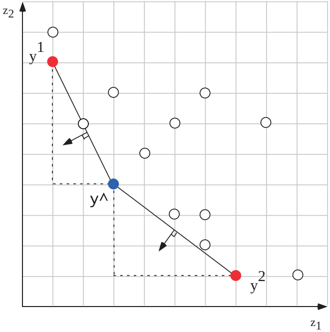
Dichotomic Method

- Let $\lambda = (\lambda_1, \lambda_2)$ be the weight vector defined by y^1 and $\hat{y}$
- Let $\bar{x}$ be an optimal solution of (4) with $\bar{y} = z(\bar{x})$
- Suppose
  $\lambda_1 \bar{y}_1 + \lambda_2 \bar{y}_2 = \lambda_1 \hat{y}_1 + \lambda_2 \hat{y}_2$
  Is the solution/point necessarily new?
- We can obtain an equivalent solution to x^1 or $\bar{x}$ or a non-extreme supported solution



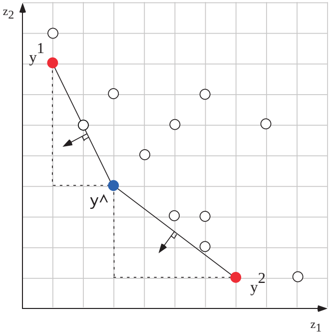
Dichotomic Method

- Let $\lambda = (\lambda_1, \lambda_2)$ be the weight vector defined by y^1 and $\hat{y}$
- Let $\bar{x}$ be an optimal solution of (4) with $\bar{y} = z(\bar{x})$
- Suppose
  $\lambda_1 \bar{y}_1 + \lambda_2 \bar{y}_2 = \lambda_1 \hat{y}_1 + \lambda_2 \hat{y}_2$
  Is the solution/point necessarily new?
- We can obtain an equivalent solution to x^1 or $\bar{x}$
  or a non-extreme supported solution



Dichotomic Method

- Let $\lambda = (\lambda_1, \lambda_2)$ be the weight vector defined by y^1 and $\hat{y}$
- Let $\bar{x}$ be an optimal solution of (4) with $\bar{y} = z(\bar{x})$
- Suppose
  $\lambda_1\bar{y}_1 + \lambda_2\bar{y}_2 = \lambda_1\hat{y}_1 + \lambda_2\hat{y}_2$
  Is the solution/point necessarily new?
- We can obtain an equivalent solution to x^1 or $\bar{x}$
  or a non-extreme supported solution



Algorithm (Aneja and Nair 1979)

- Compute $x^{(1,2)}$ and $x^{(2,1)}$ two lexicographic optimal solutions for resp. $z^{(1,2)}$ and $z^{(2,1)}$
- $\tilde{X} \leftarrow \{x^{(1,2)}, x^{(2,1)}\}$
- $\tilde{X} \leftarrow \texttt{SolveRecursion}(x^{(1,2)}, x^{(2,1)}, \tilde{X})$

Algorithm (solveRecursion)

Input : $x^1, x^2 \in X_{SE}$, $\tilde{X} \subseteq X_{SE}$

- $\lambda_1 \leftarrow z_2(x^1) - z_2(x^2)$; $\lambda_2 \leftarrow z_1(x^2) - z_1(x^1)$
- $x \leftarrow \texttt{solveWeightedSum}(\lambda)$; $\tilde{X} \leftarrow \tilde{X} \cup \{x\}$
- If $(\lambda_1 z_1(x) + \lambda_2 z_2(x) < \lambda_1 z_1(x^1) + \lambda_2 z_2(x^1))$ Then
  - $\tilde{X} \leftarrow \texttt{SolveRecursion}(x^1, x, \tilde{X})$
  - $\tilde{X} \leftarrow \texttt{SolveRecursion}(x, x^2, \tilde{X})$

Algorithm (Aneja and Nair 1979)

- Compute $x^{(1,2)}$ and $x^{(2,1)}$ two lexicographic optimal solutions for resp. $z^{(1,2)}$ and $z^{(2,1)}$
- $\tilde{X} \leftarrow \{x^{(1,2)}, x^{(2,1)}\}$
- $\tilde{X} \leftarrow \texttt{SolveRecursion}(x^{(1,2)}, x^{(2,1)}, \tilde{X})$

Algorithm (solveRecursion)

Input : $x^1, x^2 \in X_{SE}$, $\tilde{X} \subseteq X_{SE}$

- $\lambda_1 \leftarrow z_2(x^1) - z_2(x^2)$; $\lambda_2 \leftarrow z_1(x^2) - z_1(x^1)$
- $x \leftarrow \texttt{solveWeightedSum}(\lambda)$; $\tilde{X} \leftarrow \tilde{X} \cup \{x\}$
- If $(\lambda_1 z_1(x) + \lambda_2 z_2(x) < \lambda_1 z_1(x^1) + \lambda_2 z_2(x^1))$ Then
  - $\tilde{X} \leftarrow \texttt{SolveRecursion}(x^1, x, \tilde{X})$
  - $\tilde{X} \leftarrow \texttt{SolveRecursion}(x, x^2, \tilde{X})$

Algorithm (Aneja and Nair 1979)

- Compute $x^{(1,2)}$ and $x^{(2,1)}$ two lexicographic optimal solutions for resp. $z^{(1,2)}$ and $z^{(2,1)}$
- $\tilde{X} \leftarrow \{x^{(1,2)}, x^{(2,1)}\}$
- $\tilde{X} \leftarrow \texttt{SolveRecursion}(x^{(1,2)}, x^{(2,1)}, \tilde{X})$

Algorithm (solveRecursion)

Input : $x^1, x^2 \in X_{SE}$, $\tilde{X} \subseteq X_{SE}$

- $\lambda_1 \leftarrow z_2(x^1) - z_2(x^2)$; $\lambda_2 \leftarrow z_1(x^2) - z_1(x^1)$
- $x \leftarrow \texttt{solveWeightedSum}(\lambda)$; $\tilde{X} \leftarrow \tilde{X} \cup \{x\}$
- If $(\lambda_1 z_1(x) + \lambda_2 z_2(x) < \lambda_1 z_1(x^1) + \lambda_2 z_2(x^1))$ Then
  - $\tilde{X} \leftarrow \texttt{SolveRecursion}(x^1, x, \tilde{X})$
  - $\tilde{X} \leftarrow \texttt{SolveRecursion}(x, x^2, \tilde{X})$

Algorithm (Aneja and Nair 1979)

- Compute $x^{(1,2)}$ and $x^{(2,1)}$ two lexicographic optimal solutions for resp. $z^{(1,2)}$ and $z^{(2,1)}$
- $\tilde{X} \leftarrow \{x^{(1,2)}, x^{(2,1)}\}$
- $\tilde{X} \leftarrow \texttt{SolveRecursion}(x^{(1,2)}, x^{(2,1)}, \tilde{X})$

Algorithm (solveRecursion)

Input : $x^1, x^2 \in X_{SE}$, $\tilde{X} \subseteq X_{SE}$

- $\lambda_1 \leftarrow z_2(x^1) - z_2(x^2)$; $\lambda_2 \leftarrow z_1(x^2) - z_1(x^1)$
- $x \leftarrow \texttt{solveWeightedSum}(\lambda)$; $\tilde{X} \leftarrow \tilde{X} \cup \{x\}$
- If $(\lambda_1 z_1(x) + \lambda_2 z_2(x) < \lambda_1 z_1(x^1) + \lambda_2 z_2(x^1))$ Then
  - $\tilde{X} \leftarrow \texttt{SolveRecursion}(x^1, x, \tilde{X})$
  - $\tilde{X} \leftarrow \texttt{SolveRecursion}(x, x^2, \tilde{X})$

Algorithm (Aneja and Nair 1979)

- Compute $x^{(1,2)}$ and $x^{(2,1)}$ two lexicographic optimal solutions for resp. $z^{(1,2)}$ and $z^{(2,1)}$
- $\tilde{X} \leftarrow \{x^{(1,2)}, x^{(2,1)}\}$
- $\tilde{X} \leftarrow \texttt{SolveRecursion}(x^{(1,2)}, x^{(2,1)}, \tilde{X})$

Algorithm (solveRecursion)

Input : $x^1, x^2 \in X_{SE}, \tilde{X} \subseteq X_{SE}$

- $\lambda_1 \leftarrow z_2(x^1) - z_2(x^2)$; $\lambda_2 \leftarrow z_1(x^2) - z_1(x^1)$
- $x \leftarrow \texttt{solveWeightedSum}(\lambda)$; $\tilde{X} \leftarrow \tilde{X} \cup \{x\}$
- If $(\lambda_1 z_1(x) + \lambda_2 z_2(x) < \lambda_1 z_1(x^1) + \lambda_2 z_2(x^1))$ Then
  - $\tilde{X} \leftarrow \texttt{SolveRecursion}(x^1, x, \tilde{X})$
  - $\tilde{X} \leftarrow \texttt{SolveRecursion}(x, x^2, \tilde{X})$

Conclusion for Phase 1

- The dichotomic method is general
- At termination of this method, $\tilde{X}$ contains a minimal complete set X_{SE1_m} plus possibly some non-extreme and/or equivalent supported solutions
- A complete set of supported solutions X_{SE} is not necessarily obtained
- Given a weight λ, the procedure `solveWeightedSum` returns one optimal solution
- Instead, by using an algorithm enumerating all optimal solution of a single-objective problem,
  all supported solutions can be found (including equivalent ones), i.e. the set X_{SE_M}

Conclusion for Phase 1

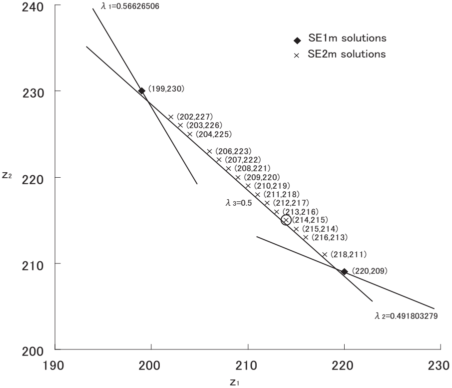
- The dichotomic method is general
- At termination of this method, $\tilde{X}$ contains a minimal complete set X_{SE1_m} plus possibly some non-extreme and/or equivalent supported solutions
- A complete set of supported solutions X_{SE} is not necessarily obtained
- Given a weight λ, the procedure `solveWeightedSum` returns one optimal solution
- Instead, by using an algorithm enumerating all optimal solution of a single-objective problem, all supported solutions can be found (including equivalent ones), i.e. the set X_{SE_M}

Conclusion for Phase 1

- The dichotomic method is general
- At termination of this method, $\tilde{X}$ contains a minimal complete set X_{SE1_m} plus possibly some non-extreme and/or equivalent supported solutions
- A complete set of supported solutions X_{SE} is not necessarily obtained
- Given a weight λ, the procedure `solveWeightedSum` returns one optimal solution
- Instead, by using an algorithm enumerating all optimal solution of a single-objective problem,
  all supported solutions can be found (including equivalent ones), i.e. the set X_{SE_M}

Illustration

However, even if an efficient algorithm exists for the considered problem, this remains an enumeration problem



Multi-objective case

- A. Przybylski, X. Gandibleux, M. Ehrgott. A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. INFORMS Journal on Computing, 22:371-386, 2010.
- Ö. Özpeynirci and M. Köksalan. An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs. Management Science, 56:2302-2315, 2010.

Phase 2: Compute other efficient solutions

- It remains to determine non-supported efficient solutions (and missing non-extreme supported efficient solutions)
- However, it is forbidden to modify the constraint structure
- We can only use weighted sum scalarization, that computes only supported solutions
- Phase 2 is therefore enumerative

Phase 2: Compute other efficient solutions

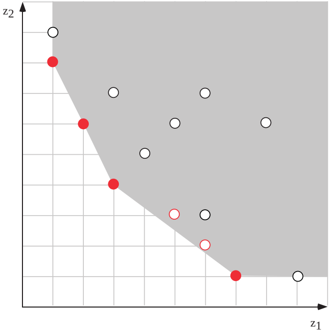
- It remains to determine non-supported efficient solutions (and missing non-extreme supported efficient solutions)
- However, it is forbidden to modify the constraint structure
- We can only use weighted sum scalarization, that computes only supported solutions
- Phase 2 is therefore enumerative

Phase 2: Compute other efficient solutions

- It remains to determine non-supported efficient solutions (and missing non-extreme supported efficient solutions)
- However, it is forbidden to modify the constraint structure
- We can only use weighted sum scalarization, that computes only supported solutions
- Phase 2 is therefore enumerative

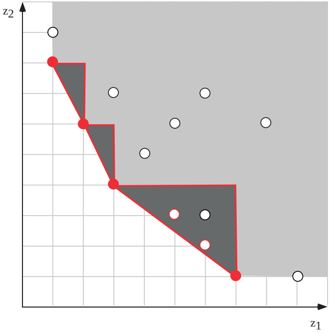
The Search Area

- All known feasible point is used to define a search area where potentially nondominated points may exist
- Using Y_{SN}, the initial search area is given by triangles defined by consecutive supported points w.r. to z_1



The Search Area

- All known feasible point is used to define a search area where potentially nondominated points may exist
- Using Y_{SN}, the initial search area is given by triangles defined by consecutive supported points w.r. to z_1



Exploration of the Search Area

- The initial search area is naturally partitionned
- Each triangle is explored separately with a problem-specific enumeration
- Using the weight λ defining the normal to the hypothenuse of the triangle, solutions x with $y = z(x)$ in the triangle are explored
- However, an enumeration of all solution x with $y = z(x)$ in the explored triangle is not acceptable
  $\Longrightarrow$ Required pruning tests in the enumeration
- Contrary to Phase 1, Phase 2 is not general

Exploration of the Search Area

- The initial search area is naturally partitionned
- Each triangle is explored separately with a problem-specific enumeration
- Using the weight λ defining the normal to the hypothenuse of the triangle, solutions x with $y = z(x)$ in the triangle are explored
- However, an enumeration of all solution x with $y = z(x)$ in the explored triangle is not acceptable
  $\Longrightarrow$ Required pruning tests in the enumeration
- Contrary to Phase 1, Phase 2 is not general

Exploration of the Search Area

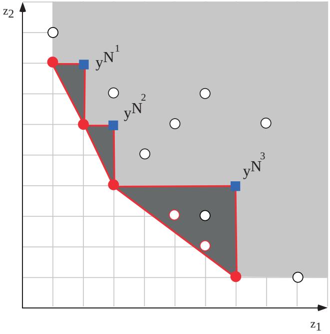
- The initial search area is naturally partitionned
- Each triangle is explored separately with a problem-specific enumeration
- Using the weight λ defining the normal to the hypothenuse of the triangle, solutions x with $y = z(x)$ in the triangle are explored
- However, an enumeration of all solution x with $y = z(x)$ in the explored triangle is not acceptable
  $\Longrightarrow$ Required pruning tests in the enumeration
- Contrary to Phase 1, Phase 2 is not general

Exploration of the Search Area

- The initial search area is naturally partitionned
- Each triangle is explored separately with a problem-specific enumeration
- Using the weight λ defining the normal to the hypothenuse of the triangle, solutions x with $y = z(x)$ in the triangle are explored
- However, an enumeration of all solution x with $y = z(x)$ in the explored triangle is not acceptable
  $\Longrightarrow$ Required pruning tests in the enumeration
- Contrary to Phase 1, Phase 2 is not general

Local Nadir Points

- *Definition:* Points defined with maximum entries of two consecutive (potentially) nondominated points
- *Properties:*
  - Used to define the search area initially and during the exploration of a triangle
  - Search area $\equiv$ area located "below" the local nadir points: $(\text{conv } Y + \mathbb{R}^2_{\geqq}) \cap \bigcup_i (y^{N^i} - \mathbb{R}^2_{\geqq})$



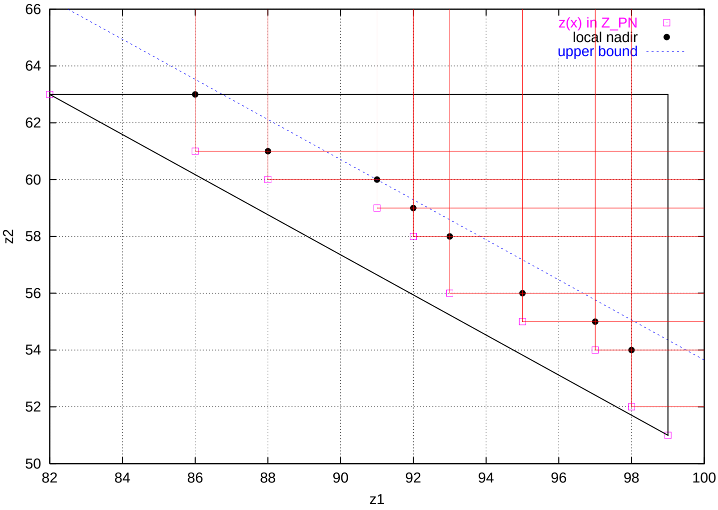
Use Local Nadir Points to Compute Upper Bounds

- Each known feasible point in the triangle can be used to reduce the search area
- It is done by updating the local nadir points
- Upper bounds β_i on $\lambda_1 z_1(x) + \lambda_2 z_2(x)$ for a solution $x \in X$ with $z(x)$ in a triangle $\Delta(y^r, y^s)$ to be efficient can be computed using these points

Use Local Nadir Points to Compute Upper Bounds

- Each known feasible point in the triangle can be used to reduce the search area
- It is done by updating the local nadir points
- Upper bounds β_i on $\lambda_1 z_1(x) + \lambda_2 z_2(x)$ for a solution $x \in X$ with $z(x)$ in a triangle $\Delta(y^r, y^s)$ to be efficient can be computed using these points

Use Local Nadir Points to Compute Upper Bounds



Use Local Nadir Points to Compute Upper Bounds

- Let $\{y^i : 1 \leq i \leq q\}$ be the set of found (potentially) non-dominated points in $\Delta(y^r, y^s)$ sorted by increasing z_1 values ($y^1 = y^r$ and $y^q = y^s$)

$$\beta_0 = \max_{i=1}^{q-1}\{\lambda_1 y_1^{i+1} + \lambda_2 y_2^i\}$$

- All $x \in X$ with $z(x)$ in the triangle and $\lambda_1 z_1(x) + \lambda_2 z_2(x) \geq \beta_0$ is dominated
- Enumerating all solution $x \in X$ such that $\lambda_1 z_1(x) + \lambda_2 z_2(x) < \beta_0$ in each triangle, we find all efficient solution, i.e. the complete set X_{E_M}

Use Local Nadir Points to Compute Upper Bounds

- Let $\{y^i : 1 \leq i \leq q\}$ be the set of found (potentially) non-dominated points in $\Delta(y^r, y^s)$ sorted by increasing z_1 values ($y^1 = y^r$ and $y^q = y^s$)

$$\beta_0 = \max_{i=1}^{q-1}\{\lambda_1 y_1^{i+1} + \lambda_2 y_2^i\}$$

- All $x \in X$ with $z(x)$ in the triangle and $\lambda_1 z_1(x) + \lambda_2 z_2(x) \geq \beta_0$ is dominated
- Enumerating all solution $x \in X$ such that $\lambda_1 z_1(x) + \lambda_2 z_2(x) < \beta_0$ in each triangle, we find all efficient solution, i.e. the complete set X_{E_M}

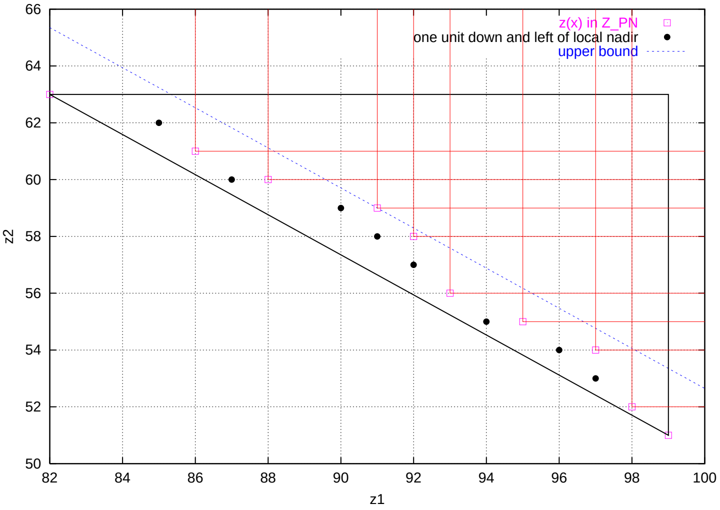
Use Local Nadir Points to Compute Upper Bounds

- Let $\{y^i : 1 \leq i \leq q\}$ be the set of found (potentially) non-dominated points in $\Delta(y^r, y^s)$ sorted by increasing z_1 values ($y^1 = y^r$ and $y^q = y^s$)

$$\beta_0 = \max_{i=1}^{q-1}\{\lambda_1 y_1^{i+1} + \lambda_2 y_2^i\}$$

- All $x \in X$ with $z(x)$ in the triangle and $\lambda_1 z_1(x) + \lambda_2 z_2(x) \geq \beta_0$ is dominated
- Enumerating all solution $x \in X$ such that $\lambda_1 z_1(x) + \lambda_2 z_2(x) < \beta_0$ in each triangle, we find all efficient solution, i.e. the complete set X_{E_M}

Use Local Nadir Points to Compute Upper Bounds



Use Local Nadir Points to Compute Upper Bounds

- Let $\{y^i : 1 \leq i \leq q\}$ be the set of found (potentially) non-dominated points in $\Delta(y^r, y^s)$ sorted by increasing z_1 values ($y^1 = y^r$ and $y^q = y^s$)

$$\delta_1 = \max_{i=1}^{q}\{\lambda_1 y_1^i + \lambda_2 y_2^i\}$$

$$\delta_2 = \max_{i=1}^{q-1}\{\lambda_1(y_1^{i+1} - 1) + \lambda_2(y_2^i - 1)\}$$

$$\beta_1 = \max\{\delta_1, \delta_2\}$$

- All $x \in X$ with $z(x)$ in the triangle and $\lambda_1 z_1(x) + \lambda_2 z_2(x) > \beta_1$ is dominated
- Enumerating all solution $x \in X$ such that $\lambda_1 z_1(x) + \lambda_2 z_2(x) \leq \beta_1$ in each triangle, we find all efficient solution, i.e. the complete set X_{E_M}

Use Local Nadir Points to Compute Upper Bounds

- Let $\{y^i : 1 \leq i \leq q\}$ be the set of found (potentially) non-dominated points in $\Delta(y^r, y^s)$ sorted by increasing z_1 values ($y^1 = y^r$ and $y^q = y^s$)

$$\delta_1 = \max_{i=1}^{q}\{\lambda_1 y_1^i + \lambda_2 y_2^i\}$$

$$\delta_2 = \max_{i=1}^{q-1}\{\lambda_1(y_1^{i+1} - 1) + \lambda_2(y_2^i - 1)\}$$

$$\beta_1 = \max\{\delta_1, \delta_2\}$$

- All $x \in X$ with $z(x)$ in the triangle and $\lambda_1 z_1(x) + \lambda_2 z_2(x) > \beta_1$ is dominated
- Enumerating all solution $x \in X$ such that $\lambda_1 z_1(x) + \lambda_2 z_2(x) \leq \beta_1$ in each triangle, we find all efficient solution, i.e. the complete set X_{E_M}

Use Local Nadir Points to Compute Upper Bounds

- Let $\{y^i : 1 \leq i \leq q\}$ be the set of found (potentially) non-dominated points in $\Delta(y^r, y^s)$ sorted by increasing z_1 values ($y^1 = y^r$ and $y^q = y^s$)

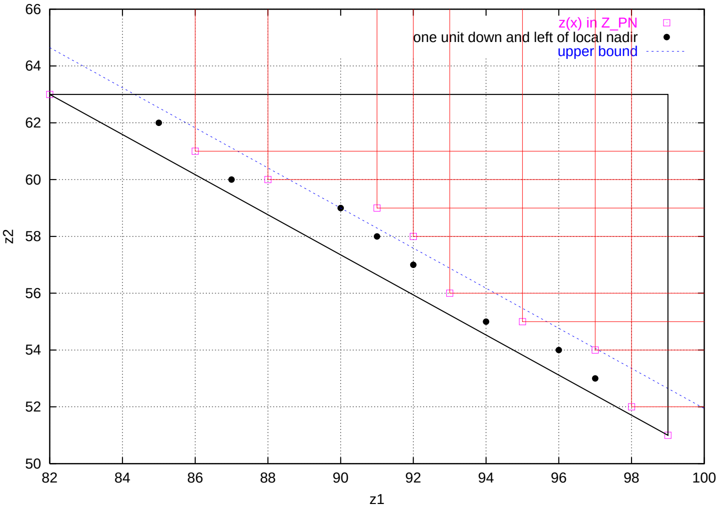
$$\delta_1 = \max_{i=1}^{q}\{\lambda_1 y_1^i + \lambda_2 y_2^i\}$$

$$\delta_2 = \max_{i=1}^{q-1}\{\lambda_1(y_1^{i+1} - 1) + \lambda_2(y_2^i - 1)\}$$

$$\beta_1 = \max\{\delta_1, \delta_2\}$$

- All $x \in X$ with $z(x)$ in the triangle and $\lambda_1 z_1(x) + \lambda_2 z_2(x) > \beta_1$ is dominated
- Enumerating all solution $x \in X$ such that $\lambda_1 z_1(x) + \lambda_2 z_2(x) \leq \beta_1$ in each triangle, we find all efficient solution, i.e. the complete set X_{E_M}

Use Local Nadir Points to Compute Upper Bounds



Use Local Nadir Points to Compute Upper Bounds

- Let $\{y^i : 1 \leq i \leq q\}$ be the set of found (potentially) non-dominated points in $\Delta(y^r, y^s)$ sorted by increasing z_1 values ($y^1 = y^r$ and $y^q = y^s$)

$$\delta_2 = \max_{i=1}^{q-1}\{\lambda_1(y_1^{i+1} - 1) + \lambda_2(y_2^i - 1)\}$$

$$\beta_2 = \delta_2$$

- All $x \in X$ with $z(x)$ in the triangle and $\lambda_1 z_1(x) + \lambda_2 z_2(x) > \beta_2$ is dominated or equivalent to a solution in X_{PE}
- Enumerating all solution $x \in X$ such that $\lambda_1 z_1(x) + \lambda_2 z_2(x) \leq \beta_2$ in each triangle, we find a complete set X_E

Use Local Nadir Points to Compute Upper Bounds

- Let $\{y^i : 1 \leq i \leq q\}$ be the set of found (potentially) non-dominated points in $\Delta(y^r, y^s)$ sorted by increasing z_1 values ($y^1 = y^r$ and $y^q = y^s$)

$$\delta_2 = \max_{i=1}^{q-1}\{\lambda_1(y_1^{i+1} - 1) + \lambda_2(y_2^i - 1)\}$$

$$\beta_2 = \delta_2$$

- All $x \in X$ with $z(x)$ in the triangle and $\lambda_1 z_1(x) + \lambda_2 z_2(x) > \beta_2$ is dominated or equivalent to a solution in X_{PE}
- Enumerating all solution $x \in X$ such that $\lambda_1 z_1(x) + \lambda_2 z_2(x) \leq \beta_2$ in each triangle, we find a complete set X_E

Use Local Nadir Points to Compute Upper Bounds

- Let $\{y^i : 1 \leq i \leq q\}$ be the set of found (potentially) non-dominated points in $\Delta(y^r, y^s)$ sorted by increasing z_1 values ($y^1 = y^r$ and $y^q = y^s$)

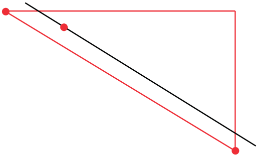
$$\delta_2 = \max_{i=1}^{q-1}\{\lambda_1(y_1^{i+1} - 1) + \lambda_2(y_2^i - 1)\}$$

$$\beta_2 = \delta_2$$

- All $x \in X$ with $z(x)$ in the triangle and $\lambda_1 z_1(x) + \lambda_2 z_2(x) > \beta_2$ is dominated or equivalent to a solution in X_{PE}
- Enumerating all solution $x \in X$ such that $\lambda_1 z_1(x) + \lambda_2 z_2(x) \leq \beta_2$ in each triangle, we find a complete set X_E

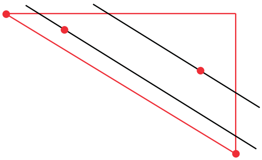
Strategy for Enumeration: Variable-Fixing

- Fix (well-chosen) variables $x_i = 1$
  Solve the weighted sum reduced problem
- Necessity to simultaneously fix a number of variables to find all non-supported solutions of the triangle
- Difficulties to order the exploration and to avoid redundancy
- Use of a list of potentially efficient solutions X_{PE} updated during all the process



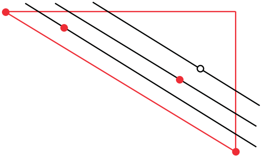
Strategy for Enumeration: Variable-Fixing

- Fix (well-chosen) variables $x_i = 1$
  Solve the weighted sum reduced problem
- Necessity to simultaneously fix a number of variables to find all non-supported solutions of the triangle
- Difficulties to order the exploration and to avoid redundancy
- Use of a list of potentially efficient solutions X_{PE} updated during all the process



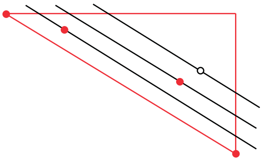
Strategy for Enumeration: Variable-Fixing

- Fix (well-chosen) variables $x_i = 1$
  Solve the weighted sum reduced problem
- Necessity to simultaneously fix a number of variables to find all non-supported solutions of the triangle
- Difficulties to order the exploration and to avoid redundancy
- Use of a list of potentially efficient solutions X_{PE} updated during all the process



Strategy for Enumeration: Variable-Fixing

- Fix (well-chosen) variables $x_i = 1$
  Solve the weighted sum reduced problem
- Necessity to simultaneously fix a number of variables to find all non-supported solutions of the triangle
- Difficulties to order the exploration and to avoid redundancy
- Use of a list of potentially efficient solutions X_{PE} updated during all the process



Strategy for Enumeration: Ranking

- Use a ranking method, i.e. an algorithm to find the K-best solution of a single objective problem

  $\Longrightarrow$
  - No modification of the problem structure
  - No repetition of solutions
  - Exploration naturally ordered
- Use of a list of efficient solutions completed during the process
  No need to remove solutions from the list
- Ranking algorithms are available for most polynomially solvable problems

- The same algorithm is applied to all triangle $\Delta(y^r, y^s)$
- $\tilde{X}$ denotes the set of collected efficient solutions in $\Delta(y^r, y^s)$

Three functions/procedures are used

- `SingleOpt`: Given $\lambda \in \mathbb{R}^2_>$, solve the weighted sum single-objective problem and return an optimal solution
- `UpdateUB`: Given $\tilde{X}$, return an upper bound β_l on $\lambda_1 z_1(x) + \lambda_2 z_2(x)$
- `Kbest`: Given $\lambda \in \mathbb{R}^2_>$ and $k \in \mathbb{N}_0$, return the $k - best$ solution of the weighted sum single-objective problem

- The same algorithm is applied to all triangle $\Delta(y^r, y^s)$
- $\tilde{X}$ denotes the set of collected efficient solutions in $\Delta(y^r, y^s)$

Three functions/procedures are used

- `SingleOpt`: Given $\lambda \in \mathbb{R}^2_>$, solve the weighted sum single-objective problem and return an optimal solution
- `UpdateUB`: Given $\tilde{X}$, return an upper bound β_i on $\lambda_1 z_1(x) + \lambda_2 z_2(x)$
- `Kbest`: Given $\lambda \in \mathbb{R}^2_>$ and $k \in \mathbb{N}_0$, return the $k - best$ solution of the weighted sum single-objective problem

- The same algorithm is applied to all triangle $\Delta(y^r, y^s)$
- $\tilde{X}$ denotes the set of collected efficient solutions in $\Delta(y^r, y^s)$

Three functions/procedures are used

- `SingleOpt`: Given $\lambda \in \mathbb{R}^2_>$, solve the weighted sum single-objective problem and return an optimal solution
- `UpdateUB`: Given $\tilde{X}$, return an upper bound β_i on $\lambda_1 z_1(x) + \lambda_2 z_2(x)$
- `Kbest`: Given $\lambda \in \mathbb{R}^2_>$ and $k \in \mathbb{N}_0$, return the $k-$*best* solution of the weighted sum single-objective problem

- The same algorithm is applied to all triangle $\Delta(y^r, y^s)$
- $\tilde{X}$ denotes the set of collected efficient solutions in $\Delta(y^r, y^s)$

Three functions/procedures are used

- `SingleOpt`: Given $\lambda \in \mathbb{R}^2_>$, solve the weighted sum single-objective problem and return an optimal solution
- `UpdateUB`: Given $\tilde{X}$, return an upper bound β_i on $\lambda_1 z_1(x) + \lambda_2 z_2(x)$
- `Kbest`: Given $\lambda \in \mathbb{R}^2_>$ and $k \in \mathbb{N}_0$, return the $k-$*best* solution of the weighted sum single-objective problem

Algorithm (Phase 2 using Ranking)

- $\lambda \leftarrow (y_2^r - y_2^s, y_1^s - y_1^r)$; $k \leftarrow 1$
- $x^1 \leftarrow \texttt{SingleOpt}(\lambda)$; $\tilde{X} \leftarrow \{x^1\}$
- $UB \leftarrow \texttt{UpdateUB}(\tilde{X})$
- While $(z^\lambda(x^k) \leq UB)$
  - $k \leftarrow k + 1$
  - $x^k \leftarrow \texttt{KBest}(k, \lambda)$;
  - If (x^k is not dominated) Then
    - $\tilde{X} \leftarrow \tilde{X} \cup \{x^k\}$
    - $UB \leftarrow \texttt{UpdateUB}(\tilde{X})$

At termination, $\tilde{X}$ contains a (maximum) complete set restricted to $\Delta(y^r, y^s)$ (depending on the used upper bound)

Algorithm (Phase 2 using Ranking)

- $\lambda \leftarrow (y_2^r - y_2^s, y_1^s - y_1^r)$; $k \leftarrow 1$
- $x^1 \leftarrow \texttt{SingleOpt}(\lambda)$; $\tilde{X} \leftarrow \{x^1\}$
- $UB \leftarrow \texttt{UpdateUB}(\tilde{X})$
- While $(z^\lambda(x^k) \leq UB)$
    - $k \leftarrow k + 1$
    - $x^k \leftarrow \texttt{KBest}(k, \lambda)$;
    - If (x^k is not dominated) Then
        - $\tilde{X} \leftarrow \tilde{X} \cup \{x^k\}$
        - $UB \leftarrow \texttt{UpdateUB}(\tilde{X})$

At termination, $\tilde{X}$ contains a (maximum) complete set restricted to $\Delta(y^r, y^s)$ (depending on the used upper bound)

Algorithm (Phase 2 using Ranking)

- $\lambda \leftarrow (y_2^r - y_2^s, y_1^s - y_1^r)$; $k \leftarrow 1$
- $x^1 \leftarrow \texttt{SingleOpt}(\lambda)$; $\tilde{X} \leftarrow \{x^1\}$
- $UB \leftarrow \texttt{UpdateUB}(\tilde{X})$
- While $(z^\lambda(x^k) \leq UB)$
  - $k \leftarrow k + 1$
  - $x^k \leftarrow \texttt{KBest}(k, \lambda)$;
  - If (x^k is not dominated) Then
    - $\tilde{X} \leftarrow \tilde{X} \cup \{x^k\}$
    - $UB \leftarrow \texttt{UpdateUB}(\tilde{X})$

At termination, $\tilde{X}$ contains a (maximum) complete set restricted to $\Delta(y^r, y^s)$ (depending on the used upper bound)

Algorithm (Phase 2 using Ranking)

- $\lambda \leftarrow (y_2^r - y_2^s, y_1^s - y_1^r)$; $k \leftarrow 1$
- $x^1 \leftarrow \texttt{SingleOpt}(\lambda)$; $\tilde{X} \leftarrow \{x^1\}$
- $UB \leftarrow \texttt{UpdateUB}(\tilde{X})$
- While $(z^\lambda(x^k) \leq UB)$
    - $k \leftarrow k + 1$
    - $x^k \leftarrow \texttt{KBest}(k, \lambda)$;
    - If (x^k is not dominated) Then
        - $\tilde{X} \leftarrow \tilde{X} \cup \{x^k\}$
        - $UB \leftarrow \texttt{UpdateUB}(\tilde{X})$

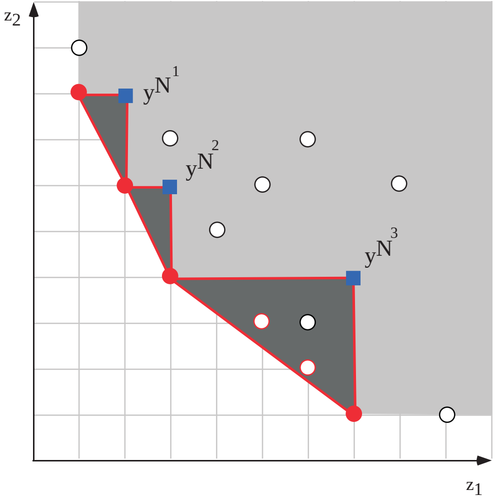
At termination, $\tilde{X}$ contains a (maximum) complete set restricted to $\Delta(y^r, y^s)$ (depending on the used upper bound)

Algorithm (Phase 2 using Ranking)

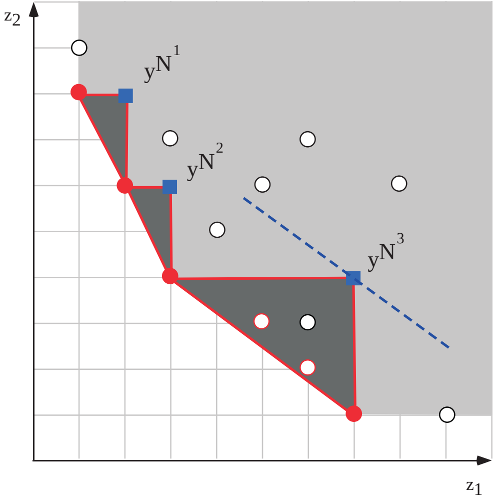
- $\lambda \leftarrow (y_2^r - y_2^s, y_1^s - y_1^r)$; $k \leftarrow 1$
- $x^1 \leftarrow \texttt{SingleOpt}(\lambda)$; $\tilde{X} \leftarrow \{x^1\}$
- $UB \leftarrow \texttt{UpdateUB}(\tilde{X})$
- While $(z^\lambda(x^k) \leq UB)$
  - $k \leftarrow k + 1$
  - $x^k \leftarrow \texttt{KBest}(k, \lambda)$;
  - If (x^k is not dominated) Then
    - $\tilde{X} \leftarrow \tilde{X} \cup \{x^k\}$
    - $UB \leftarrow \texttt{UpdateUB}(\tilde{X})$

At termination, $\tilde{X}$ contains a (maximum) complete set restricted to $\Delta(y^r, y^s)$ (depending on the used upper bound)

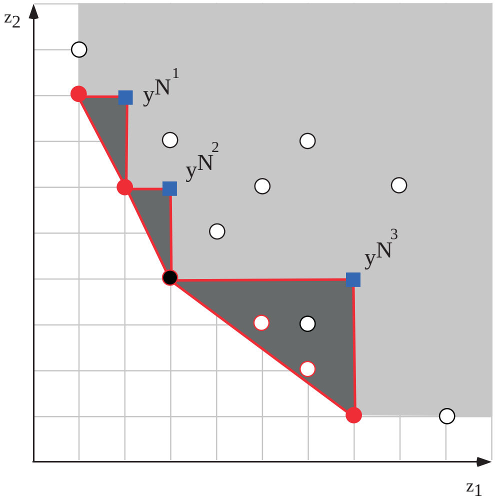
Illustration



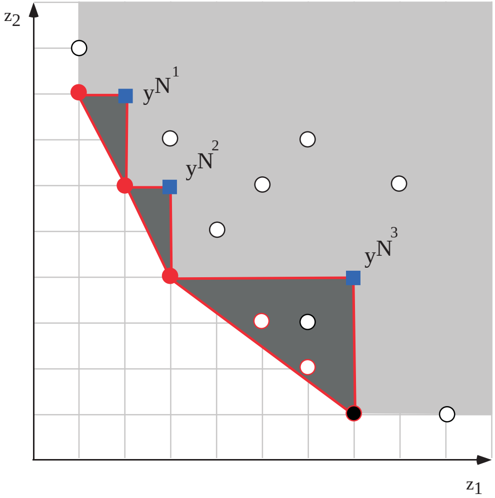
Illustration



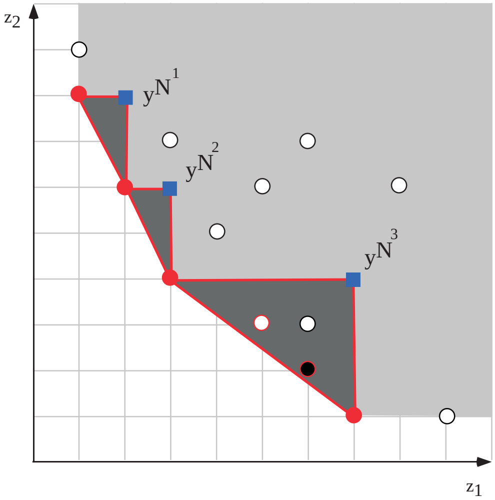
Illustration



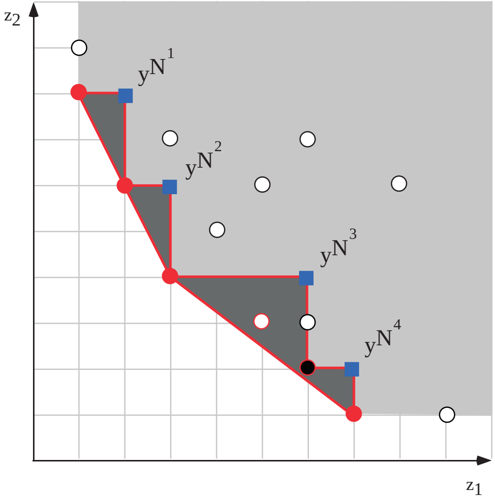
Illustration



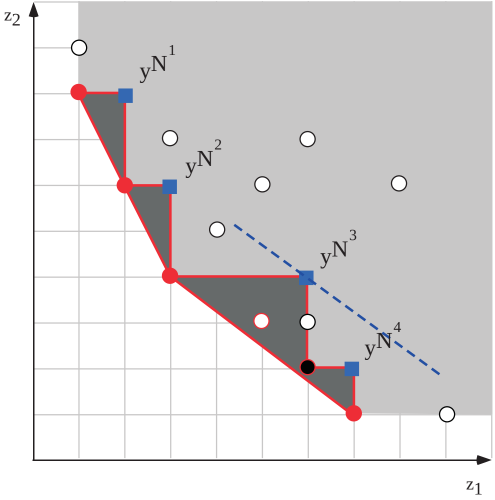
Illustration



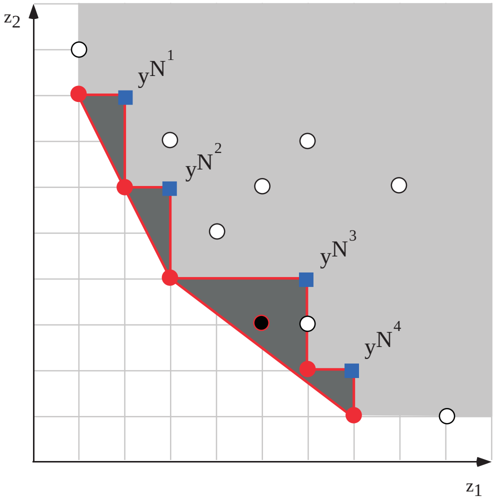
Illustration



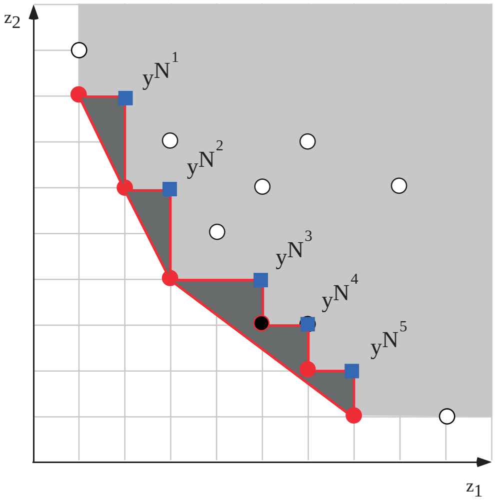
Illustration



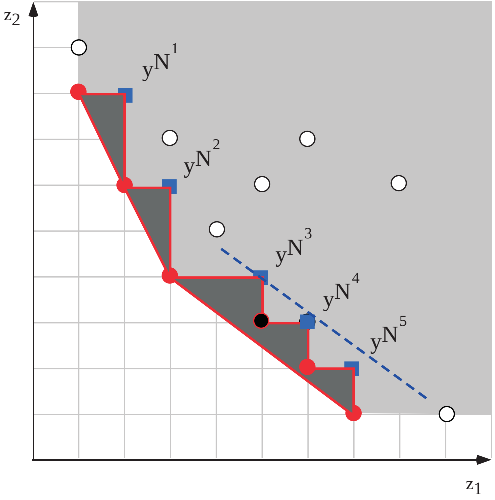
Illustration



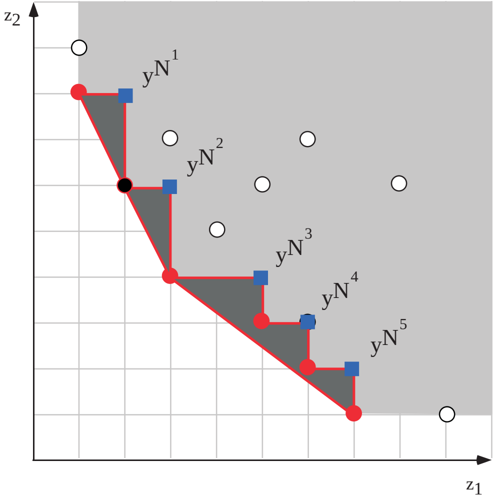
Illustration



Illustration



Illustration



Conclusion for Phase 2

- Contrary to Phase 1, the enumeration in Phase 2 is not general
- But some features are specific
  - The enumeration is done triangle by triangle
  - General upper bounds can be used
  - A general enumeration algorithm exists with the assumption that a ranking algorithm is available

Conclusion for Phase 2

- Contrary to Phase 1, the enumeration in Phase 2 is not general
- But some features are specific
  - The enumeration is done triangle by triangle
  - General upper bounds can be used
  - A general enumeration algorithm exists with the assumption that a ranking algorithm is available

Conclusion for Phase 2

- Contrary to Phase 1, the enumeration in Phase 2 is not general
- But some features are specific
  - The enumeration is done triangle by triangle
  - General upper bounds can be used
  - A general enumeration algorithm exists with the assumption that a ranking algorithm is available

Conclusion for the Two Phase Method

- Two phase method respects problem structure
- Successful for assignment, spanning tree, shortest path, knapsack and network flow problem using a ranking algorithm
- Branch and Bound algorithms have also been used successively for Phase 2
- In both phases, two phase method strongly use specific features of the bi-objective case
- Note: Two phase method is not appropriate for problem the single-objective problem is hard to solve

Conclusion for the Two Phase Method

- Two phase method respects problem structure
- Successful for assignment, spanning tree, shortest path, knapsack and network flow problem using a ranking algorithm
- Branch and Bound algorithms have also been used successively for Phase 2
- In both phases, two phase method strongly use specific features of the bi-objective case
- Note: Two phase method is not appropriate for problem the single-objective problem is hard to solve

Conclusion for the Two Phase Method

- Two phase method respects problem structure
- Successful for assignment, spanning tree, shortest path, knapsack and network flow problem using a ranking algorithm
- Branch and Bound algorithms have also been used successively for Phase 2
- In both phases, two phase method strongly use specific features of the bi-objective case
- Note: Two phase method is not appropriate for problem the single-objective problem is hard to solve

Conclusion for the Two Phase Method

- Two phase method respects problem structure
- Successful for assignment, spanning tree, shortest path, knapsack and network flow problem using a ranking algorithm
- Branch and Bound algorithms have also been used successively for Phase 2
- In both phases, two phase method strongly use specific features of the bi-objective case
- Note: Two phase method is not appropriate for problem the single-objective problem is hard to solve

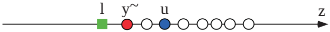
Multi-objective case

- A. Przybylski, X. Gandibleux, M. Ehrgott. A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. Discrete Optimization, 7:149-165, 2010.

Overview

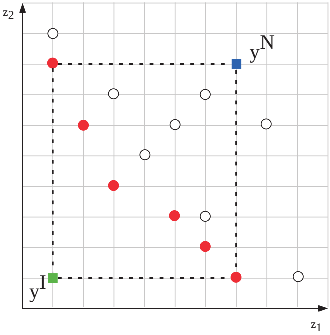
Ideal and Nadir Point

- Single-objective case:
  (Single) lower and upper bounds l and u on the (single) optimal value $\tilde{y}$
- Multi-objective case:
  (Single) bounds on the whole set Y_N naturally defined by the ideal point y^I and the nadir point y^N
- y^I and y^N generally located "far away" from Y_N
  $\Rightarrow$ Need to use several points to bound!



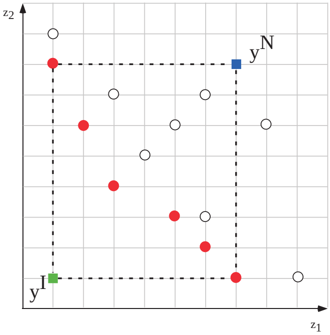
Ideal and Nadir Point

- Single-objective case:
  (Single) lower and upper bounds l and u on the (single) optimal value $\tilde{y}$
- Multi-objective case:
  (Single) bounds on the whole set Y_N naturally defined by the ideal point y^I and the nadir point y^N
- y^I and y^N generally located “far away” from Y_N
  $\Rightarrow$ Need to use several points to bound!



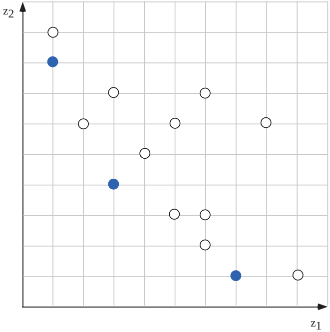
Ideal and Nadir Point

- Single-objective case:
  (Single) lower and upper bounds l and u on the (single) optimal value $\tilde{y}$
- Multi-objective case:
  (Single) bounds on the whole set Y_N naturally defined by the ideal point y^I and the nadir point y^N
- y^I and y^N generally located "far away" from Y_N
  $\Rightarrow$ Need to use several points to bound!



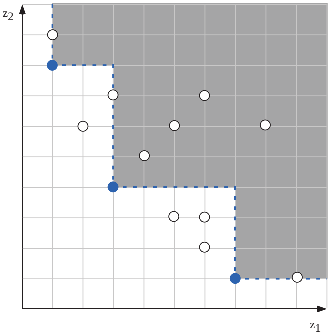
Implicit use of particular bound sets

- Early algorithms: Use of the set of all known feasible points to apply a dominance test (Incumbent list)
- Two Phase Method: Knowing Y_{SN1}, use of two sets of points to define a search area containing Y_N
  - $(\text{conv } Y_{SN1})_N$ bounds Y_N from "below" (Convex relaxation)
  - Y_{SN_1} bounds Y_N from "above" (Incumbent list)



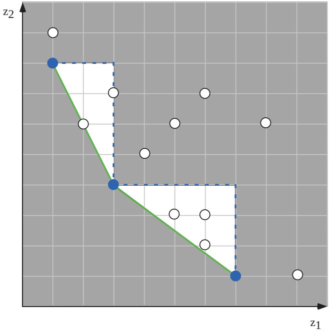
Implicit use of particular bound sets

- Early algorithms: Use of the set of all known feasible points to apply a dominance test (Incumbent list)
- Two Phase Method: Knowing Y_{SN1}, use of two sets of points to define a search area containing Y_N
  - $(\text{conv } Y_{SN1})_N$ bounds Y_N from "below" (Convex relaxation)
  - Y_{SN_1} bounds Y_N from "above" (Incumbent list)



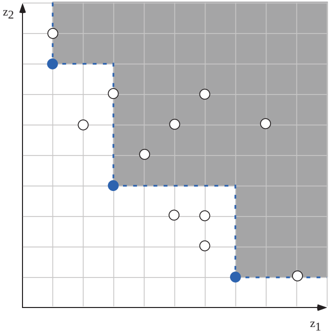
Implicit use of particular bound sets

- Early algorithms: Use of the set of all known feasible points to apply a dominance test (Incumbent list)
- Two Phase Method: Knowing Y_{SN1}, use of two sets of points to define a search area containing Y_N
  - $(\text{conv } Y_{SN1})_N$ bounds Y_N from "below" (Convex relaxation)
  - Y_{SN_1} bounds Y_N from "above" (Incumbent list)



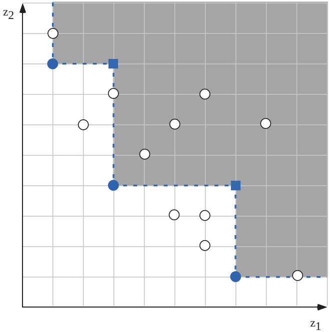
Upper bound set: Feasible points or local nadir points?

- Is Y_N bounded from "above" by the incumbent list?
- Is Y_N bounded from "above" by deduced local nadir points?
- No distinction in the single-objective case, but
  - Incumbent list is a more natural extension of Incumbent
  - Local nadir points deduced from Incumbent list
- Single-objective case: $u = l \Rightarrow u = l = \tilde{y}$
- Intersection between Incumbent list and convex relaxation $\Rightarrow$ Nondominated point!



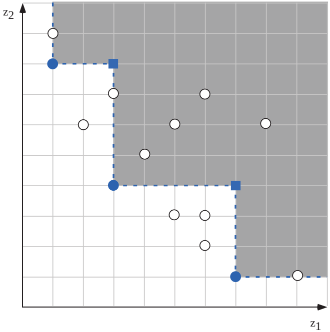
Upper bound set: Feasible points or local nadir points?

- Is Y_N bounded from "above" by the incumbent list?
- Is Y_N bounded from "above" by deduced local nadir points?
- No distinction in the single-objective case, but
  - Incumbent list is a more natural extension of Incumbent
  - Local nadir points deduced from Incumbent list
- Single-objective case: $u = l \Rightarrow u = l = \tilde{y}$
- Intersection between Incumbent list and convex relaxation $\Rightarrow$ Nondominated point!



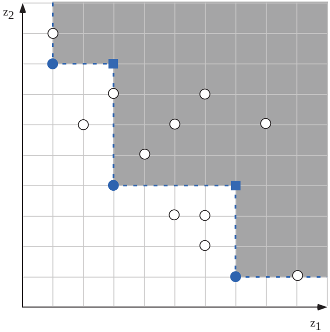
Upper bound set: Feasible points or local nadir points?

- Is Y_N bounded from "above" by the incumbent list?
- Is Y_N bounded from "above" by deduced local nadir points?
- No distinction in the single-objective case, but
    - Incumbent list is a more natural extension of Incumbent
    - Local nadir points deduced from Incumbent list
- Single-objective case: $u = l \Rightarrow u = l = \tilde{y}$
- Intersection between Incumbent list and convex relaxation $\Rightarrow$ Nondominated point!



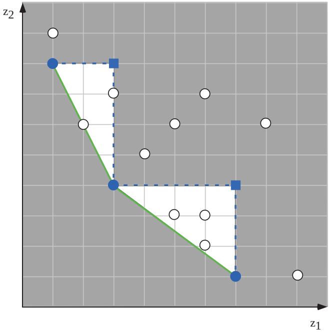
Upper bound set: Feasible points or local nadir points?

- Is Y_N bounded from "above" by the incumbent list?
- Is Y_N bounded from "above" by deduced local nadir points?
- No distinction in the single-objective case, but
  - Incumbent list is a more natural extension of Incumbent
  - Local nadir points deduced from Incumbent list
- Single-objective case: $u = l \Rightarrow u = l = \tilde{y}$
- Intersection between Incumbent list and convex relaxation $\Rightarrow$ Nondominated point!



Upper bound set: Feasible points or local nadir points?

- Is Y_N bounded from "above" by the incumbent list?
- Is Y_N bounded from "above" by deduced local nadir points?
- No distinction in the single-objective case, but
  - Incumbent list is a more natural extension of Incumbent
  - Local nadir points deduced from Incumbent list
- Single-objective case: $u = l \Rightarrow u = l = \tilde{y}$
- Intersection between Incumbent list and convex relaxation $\Rightarrow$ Nondominated point!



Desirable Properties of Bound Sets

Desirable properties of a general extension of bounds

- Single-objective case:
  - Upper bound value u given by the incumbent
  - Lower bound value l obtained by the solution of a relaxation (linear, surrogate,...)
  - Optimal value $\tilde{y}$ such that $l \leq y \leq u$
  - Equality of upper and lower bound values $\Rightarrow \tilde{y}$ found
- Multi-objective case:
  - Upper bound set U given by the incumbent list
  - Lower bound set L obtained by the solution of a relaxation (linear, convex,...)
  - $Y_N \subseteq (L + \mathbb{R}^p_{\geqq}) \setminus (U + \mathbb{R}^p_{>})$
  - $L \cap U \subseteq Y_N$
  - Equality of upper and lower bound sets $\Rightarrow Y_N$ found

Desirable Properties of Bound Sets

Desirable properties of a general extension of bounds

- Single-objective case:
  - Upper bound value u given by the incumbent
  - Lower bound value l obtained by the solution of a relaxation (linear, surrogate,...)
  - Optimal value $\tilde{y}$ such that $l \leq y \leq u$
  - Equality of upper and lower bound values $\Rightarrow$ $\tilde{y}$ found
- Multi-objective case:
  - Upper bound set U given by the incumbent list
  - Lower bound set L obtained by the solution of a relaxation (linear, convex,...)
  - $Y_N \subseteq (L + \mathbb{R}^p_{\geq}) \setminus (U + \mathbb{R}^p_{\geq})$
  - $L \cap U \subseteq Y_N$
  - Equality of upper and lower bound sets $\Rightarrow$ Y_N found

Desirable Properties of Bound Sets

Desirable properties of a general extension of bounds

- Single-objective case:
  - Upper bound value u given by the incumbent
  - Lower bound value l obtained by the solution of a relaxation (linear, surrogate,...)
  - Optimal value $\tilde{y}$ such that $l \leq y \leq u$
  - Equality of upper and lower bound values $\Rightarrow \tilde{y}$ found
- Multi-objective case:
  - Upper bound set U given by the incumbent list
  - Lower bound set L obtained by the solution of a relaxation (linear, convex,...)
  - $Y_N \subseteq (L + \mathbb{R}^p_{\geqq}) \setminus (U + \mathbb{R}^p_{\geq})$
  - $L \cap U \subseteq Y_N$
  - Equality of upper and lower bound sets $\Rightarrow Y_N$ found

Desirable Properties of Bound Sets

Desirable properties of a general extension of bounds

- Single-objective case:
  - Upper bound value u given by the incumbent
  - Lower bound value l obtained by the solution of a relaxation (linear, surrogate,...)
  - Optimal value $\tilde{y}$ such that $l \leq y \leq u$
  - Equality of upper and lower bound values $\Rightarrow$ $\tilde{y}$ found
- Multi-objective case:
  - Upper bound set U given by the incumbent list
  - Lower bound set L obtained by the solution of a relaxation (linear, convex,...)
  - $Y_N \subseteq (L + \mathbb{R}^p_{\geqq}) \setminus (U + \mathbb{R}^p_{\geq})$
  - $L \cap U \subseteq Y_N$
  - Equality of upper and lower bound sets $\Rightarrow$ Y_N found

Properties of Main Formal Definitions of Bound Sets

- Pioneer definition (Villareal and Karwan, 1981)
  - Does not support continuous sets of points as bound sets (like linear or convex relaxation)
- General definition (Ehrgott and Gandibleux, 2001)
  - Proposed to bound any subset $\bar{Y}$ of feasible points
  - Support continuous sets of points as bound sets
  - Does not support the incumbent list as an upper bound set (on Y_N)
  - Consider local nadir points as bound sets
- Modern definition (Ehrgott and Gandibleux, 2007)
  - Include the incumbent list as an upper bound set
  - Verify all desirable properties of bound sets

Properties of Main Formal Definitions of Bound Sets

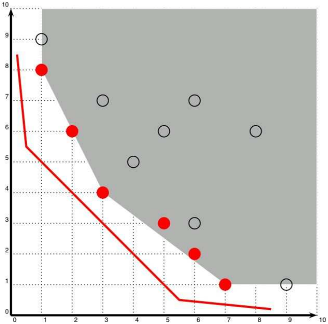
- Pioneer definition (Villareal and Karwan, 1981)
  - Does not support continuous sets of points as bound sets (like linear or convex relaxation)
- General definition (Ehrgott and Gandibleux, 2001)
  - Proposed to bound any subset $\bar{Y}$ of feasible points
  - Support continuous sets of points as bound sets
  - Does not support the incumbent list as an upper bound set (on Y_N)
  - Consider local nadir points as bound sets
- Modern definition (Ehrgott and Gandibleux, 2007)
  - Include the incumbent list as an upper bound set
  - Verify all desirable properties of bound sets

Properties of Main Formal Definitions of Bound Sets

- Pioneer definition (Villareal and Karwan, 1981)
  - Does not support continuous sets of points as bound sets (like linear or convex relaxation)
- General definition (Ehrgott and Gandibleux, 2001)
  - Proposed to bound any subset $\bar{Y}$ of feasible points
  - Support continuous sets of points as bound sets
  - Does not support the incumbent list as an upper bound set (on Y_N)
  - Consider local nadir points as bound sets
- Modern definition (Ehrgott and Gandibleux, 2007)
  - Include the incumbent list as an upper bound set
  - Verify all desirable properties of bound sets

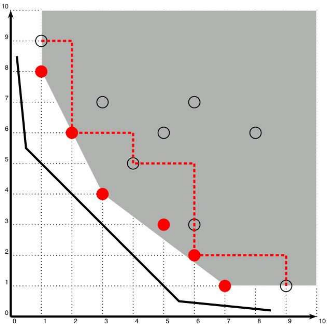
Bound sets: Definition (Ehrgott and Gandibleux, 2007)

1. Lower bound set L (for Y_N)
    - is $\mathbb{R}^p_{\geqq}$-closed
    - is $\mathbb{R}^p_{\geqq}$-bounded
    - $Y_N \subset L + \mathbb{R}^p_{\geqq}$
    - $L \subset \left(L + \mathbb{R}^p_{\geqq}\right)_N$
2. Upper bound set U (for Y_N)
    - is $\mathbb{R}^p_{\geqq}$-closed
    - is $\mathbb{R}^p_{\geqq}$-bounded
    - $Y_N \subset \mathrm{cl}\left[\left(U + \mathbb{R}^p_{\geqq}\right)^c\right]$
    - $U \subset \left(U + \mathbb{R}^p_{\geqq}\right)_N$



Bound sets: Definition (Ehrgott and Gandibleux, 2007)

1. Lower bound set L (for Y_N)
   - is $\mathbb{R}^p_{\geqq}$-closed
   - is $\mathbb{R}^p_{\geqq}$-bounded
   - $Y_N \subset L + \mathbb{R}^p_{\geqq}$
   - $L \subset \left(L + \mathbb{R}^p_{\geqq}\right)_N$
2. Upper bound set U (for Y_N)
   - is $\mathbb{R}^p_{\geqq}$-closed
   - is $\mathbb{R}^p_{\geqq}$-bounded
   - $Y_N \subset \mathrm{cl}\left[\left(U + \mathbb{R}^p_{\geqq}\right)^c\right]$
   - $U \subset \left(U + \mathbb{R}^p_{\geqq}\right)_N$



Branch and Bound: Main Components

Main component of a single-objective branch and bound

- Upper and lower bound values
- Separation procedure
- Choice of the active node
- Generation of feasible solution

Extension

- Upper and lower bound sets
- No change in the partitioning strategy (concerns the feasible set)... But how to choose a "good" variable to apply the separation?
- Immediate adaptation of depth-first search, breadth-first search... But how to apply a dynamic strategy?
- Generation of feasible solutions

Branch and Bound: Main Components

Main component of a single-objective branch and bound

- Upper and lower bound values
- Separation procedure
- Choice of the active node
- Generation of feasible solution

Extension

- Upper and lower bound sets
- No change in the partitioning strategy (concerns the feasible set)... But how to choose a “good” variable to apply the separation?
- Immediate adaptation of depth-first search, breadth-first search... But how to apply a dynamic strategy?
- Generation of feasible solutions

Branch and Bound: Main Components

Main component of a single-objective branch and bound

- Upper and lower bound values
- Separation procedure
- Choice of the active node
- Generation of feasible solution

Extension

- Upper and lower bound sets
- No change in the partitioning strategy (concerns the feasible set)... But how to choose a "good" variable to apply the separation?
- Immediate adaptation of depth-first search, breadth-first search... But how to apply a dynamic strategy?
- Generation of feasible solutions

Branch and Bound: Main Components

Main component of a single-objective branch and bound

- Upper and lower bound values
- Separation procedure
- Choice of the active node
- Generation of feasible solution

Extension

- Upper and lower bound sets
- No change in the partitioning strategy (concerns the feasible set)... But how to choose a "good" variable to apply the separation?
- Immediate adaptation of depth-first search, breadth-first search... But how to apply a dynamic strategy?
- Generation of feasible solutions

Fathoming nodes by infeasibility and optimality

- Fathoming by infeasibility: Identical to the single-objective case
  (concerns only the feasible set)
- Fathoming by optimality:
  - Early algorithms: Lower bound set → Ideal feasible point
  - Possibility to do better (?):
    Let $\overline{Y}$ be the feasible set of a subproblem
    1. Compute upper bound set on $\overline{Y}_N$
    2. Compute lower bound set on $\overline{Y}_N$
    3. Check equality between both bound sets (Unlikely to happen!)

Fathoming nodes by infeasibility and optimality

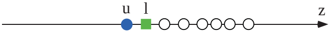
- Fathoming by infeasibility: Identical to the single-objective case
  (concerns only the feasible set)
- Fathoming by optimality:
  - Early algorithms: Lower bound set $\rightarrow$ Ideal feasible point
  - Possibility to do better (?):
    Let $\bar{Y}$ be the feasible set of a subproblem
    1. Compute upper bound set on $\bar{Y}_N$
    2. Compute lower bound set on $\bar{Y}_N$
    3. Check equality between both bound sets (Unlikely to happen!)

Fathoming nodes by infeasibility and optimality

- Fathoming by infeasibility: Identical to the single-objective case
  (concerns only the feasible set)
- Fathoming by optimality:
  - Early algorithms: Lower bound set $\rightarrow$ Ideal feasible point
  - Possibility to do better (?):
    Let $\bar{Y}$ be the feasible set of a subproblem
    1. Compute upper bound set on $\bar{Y}_N$
    2. Compute lower bound set on $\bar{Y}_N$
    3. Check equality between both bound sets (Unlikely to happen!)

Fathoming nodes by dominance

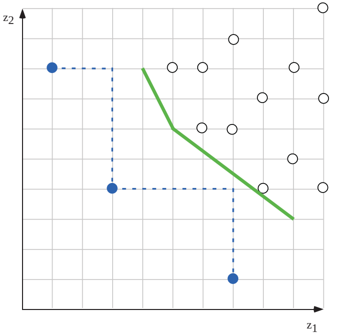
- Single-objective case
  - Upper bound u: Incumbent
  - Lower bound l for a subproblem: Relaxation of this subproblem
  - Node fathomed if $u \leq l$
- Multi-objective case
  - Upper bound set U: Incumbent List
  - Lower bound set L for $\bar{Y}_N$: relaxation of a subproblem
  - Node fathomed if $\forall l \in L$, there is $u \in U$ with $u \leqq l$
  - Main question: How to apply this dominance test?



Node fathomed!

Fathoming nodes by dominance

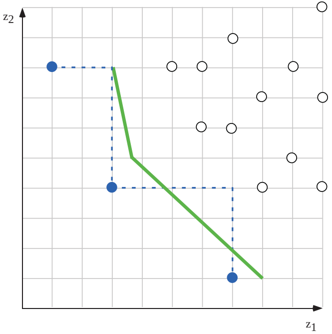
- Single-objective case
  - Upper bound u: Incumbent
  - Lower bound l for a subproblem: Relaxation of this subproblem
  - Node fathomed if $u \leq l$
- Multi-objective case
  - Upper bound set U: Incumbent List
  - Lower bound set L for $\bar{Y}_N$: relaxation of a subproblem
  - Node fathomed if $\forall l \in L$, there is $u \in U$ with $u \leqq l$
  - Main question: How to apply this dominance test?



Node fathomed!

Fathoming nodes by dominance

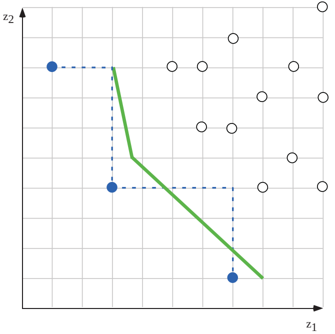
- Single-objective case
  - Upper bound u: Incumbent
  - Lower bound l for a subproblem: Relaxation of this subproblem
  - Node fathomed if $u \leq l$
- Multi-objective case
  - Upper bound set U: Incumbent List
  - Lower bound set L for $\bar{Y}_N$: relaxation of a subproblem
  - Node fathomed if $\forall l \in L$, there is $u \in U$ with $u \leqq l$
  - Main question: How to apply this dominance test?



Node not fathomed!

Fathoming nodes by dominance

- Single-objective case
  - Upper bound u: Incumbent
  - Lower bound l for a subproblem: Relaxation of this subproblem
  - Node fathomed if $u \leq l$
- Multi-objective case
  - Upper bound set U: Incumbent List
  - Lower bound set L for $\bar{Y}_N$: relaxation of a subproblem
  - Node fathomed if $\forall l \in L$, there is $u \in U$ with $u \leqq l$
  - Main question: How to apply this dominance test?



Node not fathomed!

Practical Application of dominance test

(Assumption: U is discrete)

- Lower bound set L composed of a finite set of points:
  - Pairwise comparison between points of L and U
- Lower bound set L composed of an infinite set of points:
  - Pairwise comparison between all pairs of points not possible!
  - (Sourd and Spaanjard, 2008): Proposition for the particular case for which $L + \mathbb{R}^2_{\geqq}$ is a polyhedron

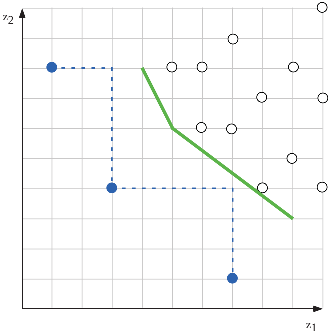
Practical Application of dominance test

(Assumption: U is discrete)

- Lower bound set L composed of a finite set of points:
  - Pairwise comparison between points of L and U
- Lower bound set L composed of an infinite set of points:
  - Pairwise comparison between all pairs of points not possible!
  - (Sourd and Spaanjard, 2008): Proposition for the particular case for which $L + \mathbb{R}^2_{\geqq}$ is a polyhedron

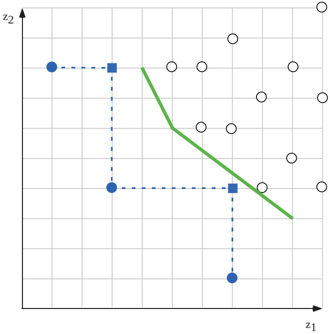
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



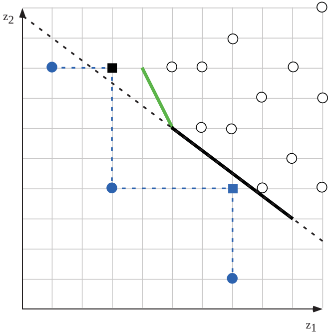
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



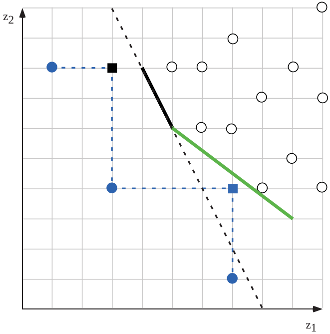
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



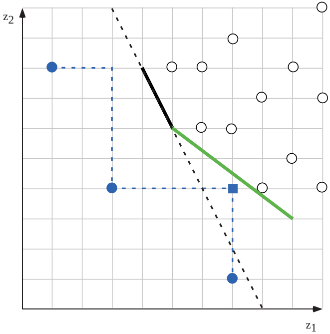
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



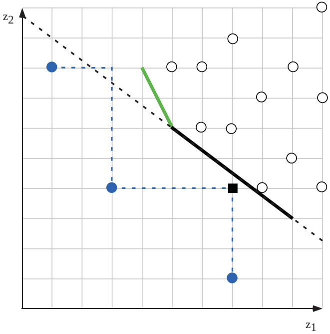
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



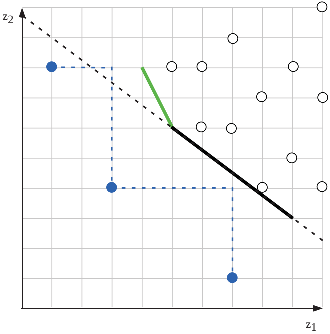
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



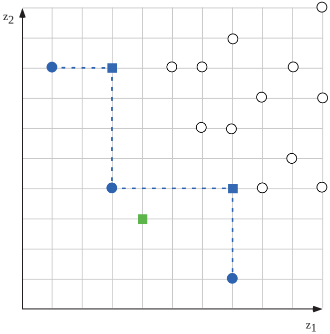
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



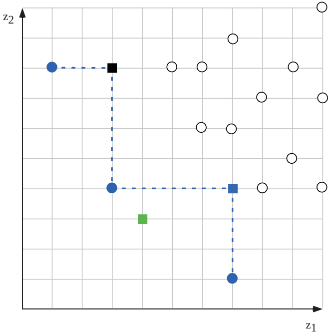
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



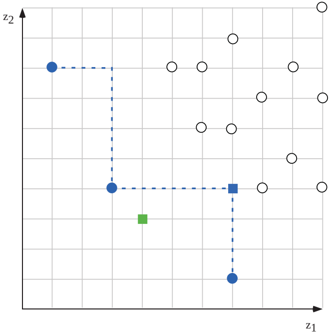
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



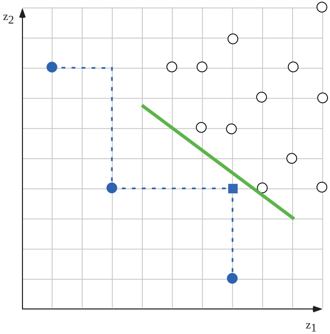
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



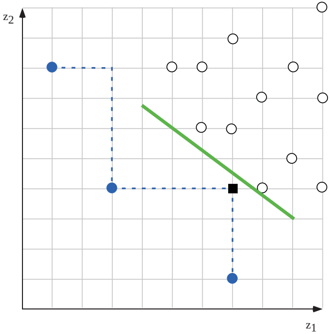
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



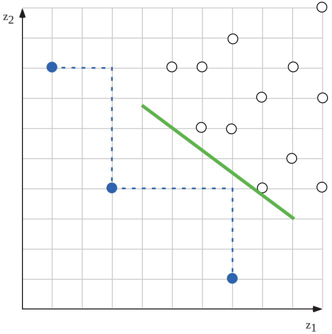
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



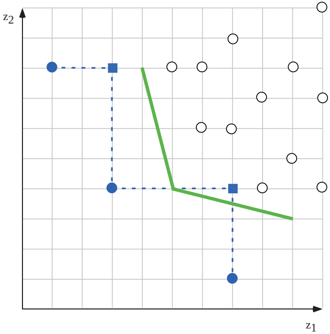
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



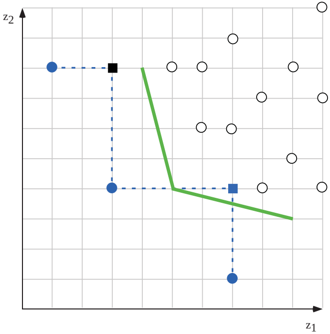
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



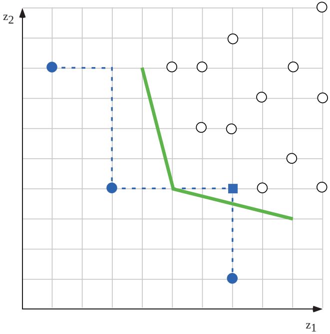
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



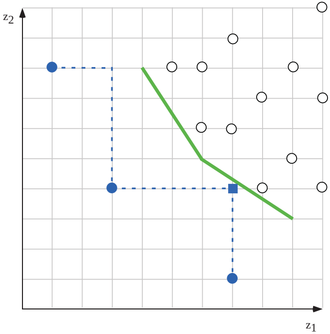
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



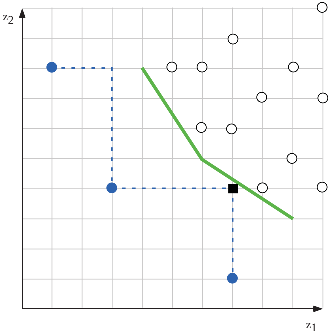
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



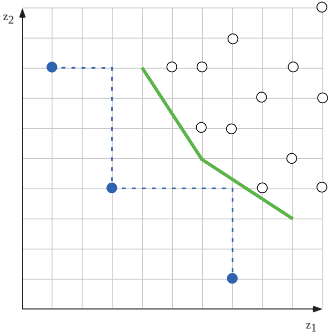
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



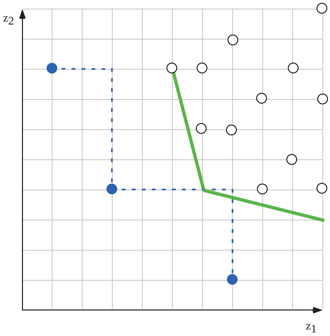
Practical Application of dominance test

- Use of corner points
  - For each local nadir point u, is there an edge of L such that u is located below?
- Possible partial computation of lower bound set when node fathomed
- Partial computation thanks to reoptimization from the father node to the child node



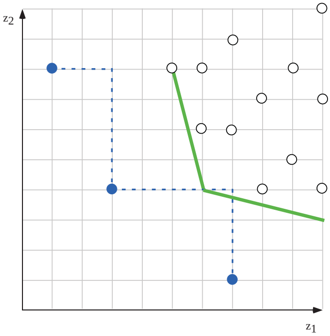
Global Branch & Bound versus Local Branch & Bound

- Fewer nodes fathomed by dominance with increasing number of objectives
- Partition objective space to apply several local B & B or apply a global B& B algorithm ?
- (Visée et al., 1997) proposed the use of B&B as Phase 2 of Two Phase Method
- Advantages:
  - More nodes fathomed by dominance
  - Preprocessing more effective
- Drawback
  - Possible redundant computations



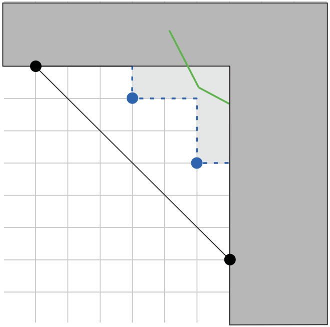
Global Branch & Bound versus Local Branch & Bound

- Fewer nodes fathomed by dominance with increasing number of objectives
- Partition objective space to apply several local B & B or apply a global B& B algorithm ?
- (Visée et al., 1997) proposed the use of B&B as Phase 2 of Two Phase Method
- Advantages:
  - More nodes fathomed by dominance
  - Preprocessing more effective
- Drawback
  - Possible redundant computations



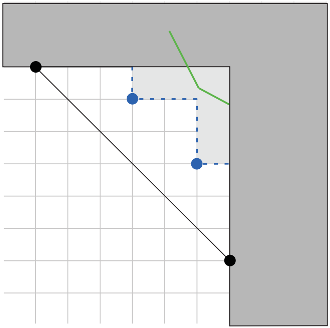
Global Branch & Bound versus Local Branch & Bound

- Fewer nodes fathomed by dominance with increasing number of objectives
- Partition objective space to apply several local B & B or apply a global B& B algorithm ?
- (Visée et al., 1997) proposed the use of B&B as Phase 2 of Two Phase Method
- Advantages:
  - More nodes fathomed by dominance
  - Preprocessing more effective
- Drawback
  - Possible redundant computations



Global Branch & Bound versus Local Branch & Bound

- Fewer nodes fathomed by dominance with increasing number of objectives
- Partition objective space to apply several local B & B or apply a global B& B algorithm ?
- (Visée et al., 1997) proposed the use of B&B as Phase 2 of Two Phase Method
- Advantages:
  - More nodes fathomed by dominance
  - Preprocessing more effective
- Drawback
  - Possible redundant computations



MOB&B: the 01 Case

Presentation

$$\begin{array}{rcl} \min z(x) & = & Cx \\ \text{subject to } Ax & = & b \\ x & \in & \{0,1\}^n \end{array}$$

$$\begin{array}{rcl} x \in \{0,1\}^n & \longrightarrow & n \text{ variables, } i = 1, \ldots, n \\ A \in \mathbb{Z}^{m \times n} & \longrightarrow & m \text{ constraints, } j = 1, \ldots, m \\ C \in \mathbb{Z}^{p \times n} & \longrightarrow & p \text{ (sum) objective vectors, } k = 1, \ldots, p \end{array}$$

The pionner algorithm (Kiziltan and Yucaoglu, 1983)

- Problem: MO01LP in maximization case, $C \in \mathbb{Z}^{p \times n}$
- Lower bound set on Y_N: Incumbent list
- Upper bound set on $\bar{Y}_N$: Ideal point of the unconstrained problem
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of solution by fixing all free variable to 0 before a feasibility check
- Separation procedure:
  - One variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (dynamic strategy):

Additional dominance test: Before to branch on a new variable, test if any possible child node can be fathomed to reduce the number of variables
Idea still applied in recent algorithm as a preprocessing at the root node

The pionner algorithm (Kiziltan and Yucaoglu, 1983)

- Problem: MO01LP in maximization case, $C \in \mathbb{Z}^{p \times n}$
- Lower bound set on Y_N: Incumbent list
- Upper bound set on $\bar{Y}_N$: Ideal point of the unconstrained problem
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of solution by fixing all free variable to 0 before a feasibility check
- Separation procedure:
  - One variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (dynamic strategy):

Additional dominance test: Before to branch on a new variable, test if any possible child node can be fathomed to reduce the number of variables
Idea still applied in recent algorithm as a preprocessing at the root node

The pionner algorithm (Kiziltan and Yucaoglu, 1983)

- Problem: MO01LP in maximization case, $C \in \mathbb{Z}^{p \times n}$
- Lower bound set on Y_N: Incumbent list
- Upper bound set on $\bar{Y}_N$: Ideal point of the unconstrained problem
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of solution by fixing all free variable to 0 before a feasibility check
- Separation procedure:
  - One variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (dynamic strategy):
    - Constructed solution feasible $\rightarrow$ any free variable x_j such that $c_j \not\leq 0$
    - Otherwise, consideration of each possible child node (by fixing $x_j = 1$) until a feasible solution is constructed
    - If no feasible solution can be constructed in a child node, infeasibility is measured by summing the slack variables of unsatisfied constraint, to branch on the variable nearest to feasibility

Additional dominance test: Before to branch on a new variable, test if any possible child node can be fathomed to reduce the number of variables

Idea still applied in recent algorithm as a preprocessing at the root node

The pionner algorithm (Kiziltan and Yucaoglu, 1983)

- Problem: MO01LP in maximization case, $C \in \mathbb{Z}^{p \times n}$
- Lower bound set on Y_N: Incumbent list
- Upper bound set on $\bar{Y}_N$: Ideal point of the unconstrained problem
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of solution by fixing all free variable to 0 before a feasibility check
- Separation procedure:
  - One variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (dynamic strategy):
    - Constructed solution feasible $\rightarrow$ any free variable x_j such that $c_j \nleqq 0$
    - Otherwise, consideration of each possible child node (by fixing $x_j = 1$) until a feasible solution is constructed
    - If no feasible solution can be constructed in a child node, infeasibility is measured by summing the slack variables of unsatisfied constraint, to branch on the variable nearest to feasibility

Additional dominance test: Before to branch on a new variable, test if any possible child node can be fathomed to reduce the number of variables
Idea still applied in recent algorithm as a preprocessing at the root node

The pionner algorithm (Kiziltan and Yucaoglu, 1983)

- Problem: MO01LP in maximization case, $C \in \mathbb{Z}^{p \times n}$
- Lower bound set on Y_N: Incumbent list
- Upper bound set on $\bar{Y}_N$: Ideal point of the unconstrained problem
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of solution by fixing all free variable to 0 before a feasibility check
- Separation procedure:
  - One variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (dynamic strategy):
    - Constructed solution feasible $\rightarrow$ any free variable x_j such that $c_j \nleq 0$
    - Otherwise, consideration of each possible child node (by fixing $x_j = 1$) until a feasible solution is constructed
    - If no feasible solution can be constructed in a child node, infeasibility is measured by summing the slack variables of unsatisfied constraint, to branch on the variable nearest to feasibility

Additional dominance test: Before to branch on a new variable, test if any possible child node can be fathomed to reduce the number of variables
Idea still applied in recent algorithm as a preprocessing at the root node

The pionner algorithm (Kiziltan and Yucaoglu, 1983)

- Problem: MO01LP in maximization case, $C \in \mathbb{Z}^{p \times n}$
- Lower bound set on Y_N: Incumbent list
- Upper bound set on $\bar{Y}_N$: Ideal point of the unconstrained problem
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of solution by fixing all free variable to 0 before a feasibility check
- Separation procedure:
  - One variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (dynamic strategy):
    - Constructed solution feasible $\rightarrow$ any free variable x_j such that $c_j \nleqq 0$
    - Otherwise, consideration of each possible child node (by fixing $x_j = 1$) until a feasible solution is constructed
    - If no feasible solution can be constructed in a child node, infeasibility is measured by summing the slack variables of unsatisfied constraint, to branch on the variable nearest to feasibility

Additional dominance test: Before to branch on a new variable, test if any possible child node can be fathomed to reduce the number of variables
Idea still applied in recent algorithm as a preprocessing at the root node

The pionner algorithm (Kiziltan and Yucaoglu, 1983)

- Problem: MO01LP in maximization case, $C \in \mathbb{Z}^{p \times n}$
- Lower bound set on Y_N: Incumbent list
- Upper bound set on $\bar{Y}_N$: Ideal point of the unconstrained problem
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of solution by fixing all free variable to 0 before a feasibility check
- Separation procedure:
  - One variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (dynamic strategy):
    - Constructed solution feasible $\rightarrow$ any free variable x_j such that $c_j \nleqq 0$
    - Otherwise, consideration of each possible child node (by fixing $x_j = 1$) until a feasible solution is constructed
    - If no feasible solution can be constructed in a child node, infeasibility is measured by summing the slack variables of unsatisfied constraint, to branch on the variable nearest to feasibility

Additional dominance test: Before to branch on a new variable, test if any possible child node can be fathomed to reduce the number of variables
Idea still applied in recent algorithm as a preprocessing at the root node

Embedded into the two phase method

Ulungu and Teghem (1997)

- Problem: bi-objective uni-dimensional binary knapsack problem
- Lower bound set on Y_N: Incumbent list
- Upper bound set on $\bar{Y}_N$: For each objective function z^k, one single-objective upper bound is computed (Martello and Toth)
  $\rightarrow$ One single point is obtained as an upper bound set
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of one feasible solution by fixing all free variables to 0
- Separation procedure:
  - One free variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (static strategy):
    Order O_k defined by sorting c_j^k/ω_j by decreasing order for $k \in \{1,2\}$, and definition of the rank r_j^k of each item j according to O_k
    Consideration of items j by decreasing value of the sum $r_j^1 + r_j^2$

Embedded into the two phase method

Ulungu and Teghem (1997)

- Problem: bi-objective uni-dimensional binary knapsack problem
- Lower bound set on Y_N: Incumbent list
- Upper bound set on $\bar{Y}_N$: For each objective function z^k, one single-objective upper bound is computed (Martello and Toth)
  $\rightarrow$ One single point is obtained as an upper bound set
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of one feasible solution by fixing all free variables to 0
- Separation procedure:
  - One free variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (static strategy):
    Order O_k defined by sorting c_j^k/ω_j by decreasing order for $k \in \{1, 2\}$, and definition of the rank r_j^k of each item j according to O_k
    Consideration of items j by decreasing value of the sum $r_j^1 + r_j^2$

Embedded into the two phase method

Ulungu and Teghem (1997)

- Problem: bi-objective uni-dimensional binary knapsack problem
- Lower bound set on Y_N: Incumbent list
- Upper bound set on $\bar{Y}_N$: For each objective function z^k, one single-objective upper bound is computed (Martello and Toth)
  $\rightarrow$ One single point is obtained as an upper bound set
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of one feasible solution by fixing all free variables to 0
- Separation procedure:
  - One free variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (static strategy):
    Order $\mathcal{O}_k$ defined by sorting c_j^k/ω_j by decreasing order for $k \in \{1, 2\}$, and definition of the rank r_j^k of each item j according to $\mathcal{O}_k$
    Consideration of items j by decreasing value of the sum $r_j^1 + r_j^2$

Embedded into the two phase method

Visée et al. (1998)

Application of a B&B algorithm in each triangle Δ with weight λ

- Problem: bi-objective uni-dimensional binary knapsack problem
- Lower bound set on $Y_N \cap \Delta$: Incumbent list, restricted to Δ
- Upper bound set on $\bar{Y}_N \cap \Delta$: For each objective z^k and for the weighted sum $\lambda^T z$, a single-objective upper bound is computed $\rightarrow$ One edge is obtained as an upper bound set
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of one feasible solution by fixing all free variables to 0, need to test if it belongs to Δ
- Separation procedure:
  - One free variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (static strategy):

    Consideration of items j by decreasing value of $\lambda^T \left(\frac{c_j^1}{w_j}, \frac{c_j^2}{w_j}\right)^T$

Preprocessing applied at the root node for each triangle Δ

Embedded into the two phase method

Visée et al. (1998)

Application of a B&B algorithm in each triangle Δ with weight λ

- Problem: bi-objective uni-dimensional binary knapsack problem
- Lower bound set on $Y_N \cap \Delta$: Incumbent list, restricted to Δ
- Upper bound set on $\bar{Y}_N \cap \Delta$: For each objective z^k and for the weighted sum $\lambda^T z$, a single-objective upper bound is computed
  $\rightarrow$ One edge is obtained as an upper bound set
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of one feasible solution by fixing all free variables to 0, need to test if it belongs to Δ
- Separation procedure:
  - One free variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (static strategy):

    Consideration of items j by decreasing value of $\lambda^T \left(\frac{c_j^1}{w_j}, \frac{c_j^2}{w_j}\right)^T$

Preprocessing applied at the root node for each triangle Δ

Embedded into the two phase method

Visée et al. (1998)

Application of a B&B algorithm in each triangle Δ with weight λ

- Problem: bi-objective uni-dimensional binary knapsack problem
- Lower bound set on $Y_N \cap \Delta$: Incumbent list, restricted to Δ
- Upper bound set on $\bar{Y}_N \cap \Delta$: For each objective z^k and for the weighted sum $\lambda^T z$, a single-objective upper bound is computed $\rightarrow$ One edge is obtained as an upper bound set
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of one feasible solution by fixing all free variables to 0, need to test if it belongs to Δ
- Separation procedure:
  - One free variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (static strategy):

    Consideration of items j by decreasing value of $\lambda^T \left(\frac{c_j^1}{w_j}, \frac{c_j^2}{w_j}\right)^T$

Preprocessing applied at the root node for each triangle Δ

Embedded into the two phase method

Visée et al. (1998)

Application of a B&B algorithm in each triangle Δ with weight λ

- Problem: bi-objective uni-dimensional binary knapsack problem
- Lower bound set on $Y_N \cap \Delta$: Incumbent list, restricted to Δ
- Upper bound set on $\bar{Y}_N \cap \Delta$: For each objective z^k and for the weighted sum $\lambda^T z$, a single-objective upper bound is computed $\rightarrow$ One edge is obtained as an upper bound set
- Choice of the active node: Depth-first search
- Construction of feasible points: At each node, construction of one feasible solution by fixing all free variables to 0, need to test if it belongs to Δ
- Separation procedure:
  - One free variable x_j is fixed at the value 1 first and next to 0
  - Choice of x_j (static strategy):

    Consideration of items j by decreasing value of $\lambda^T \left(\frac{c_j^1}{w_j}, \frac{c_j^2}{w_j}\right)^T$

Preprocessing applied at the root node for each triangle Δ

Using convex relaxation as a lower bound set

Sourd and Spanjaard (2008)

- Problem: bi-objective minimum weight spanning tree problem
- Upper Bound set on Y_N: Incumbent list
- Lower Bound Set on $\bar{Y}_N$: convex relaxation
- Choice of the active node: depth-first search
- Separation procedure:
  - One free edge e is mandatory first and next forbidden
  - Choice of e (static strategy) : e such that $\min\{w_e^1, w_e^2\}$ is minimal
- Construction of feasible points
  - Initialization of the incumbent with the computation of Y_{SN1}, completed with a local search
  - Convex relaxation always generates feasible points

Preprocessing: adaptations of the cut optimality condition and the cycle optimality condition to reduce the size of the graph

Using convex relaxation as a lower bound set

Sourd and Spanjaard (2008)

- Problem: bi-objective minimum weight spanning tree problem
- Upper Bound set on Y_N: Incumbent list
- Lower Bound Set on $\bar{Y}_N$: convex relaxation
- Choice of the active node: depth-first search
- Separation procedure:
  - One free edge e is mandatory first and next forbidden
  - Choice of e (static strategy) : e such that $\min\{w_e^1, w_e^2\}$ is minimal
- Construction of feasible points
  - Initialization of the incumbent with the computation of Y_{SN1}, completed with a local search
  - Convex relaxation always generates feasible points

Preprocessing: adaptations of the cut optimality condition and the cycle optimality condition to reduce the size of the graph

Using convex relaxation as a lower bound set

Sourd and Spanjaard (2008)

- Problem: bi-objective minimum weight spanning tree problem
- Upper Bound set on Y_N: Incumbent list
- Lower Bound Set on $\bar{Y}_N$: convex relaxation
- Choice of the active node: depth-first search
- Separation procedure:
  - One free edge e is mandatory first and next forbidden
  - Choice of e (static strategy) : e such that $\min\{w_e^1, w_e^2\}$ is minimal
- Construction of feasible points
  - Initialization of the incumbent with the computation of Y_{SN1}, completed with a local search
  - Convex relaxation always generates feasible points

Preprocessing: adaptations of the cut optimality condition and the cycle optimality condition to reduce the size of the graph

Using convex relaxation as a lower bound set

Sourd and Spanjaard (2008)

- Problem: bi-objective minimum weight spanning tree problem
- Upper Bound set on Y_N: Incumbent list
- Lower Bound Set on $\bar{Y}_N$: convex relaxation
- Choice of the active node: depth-first search
- Separation procedure:
  - One free edge e is mandatory first and next forbidden
  - Choice of e (static strategy) : e such that $\min\{w_e^1, w_e^2\}$ is minimal
- Construction of feasible points
  - Initialization of the incumbent with the computation of Y_{SN1}, completed with a local search
  - Convex relaxation always generates feasible points

Preprocessing: adaptations of the cut optimality condition and the cycle optimality condition to reduce the size of the graph

01 Case: Entries found in the literature

- Nakamura and Riley (1981)
  the older reference found (unfortunately not rigorous)
- Kiziltan and Yucaoglu (1983)
  any MO01ILP
- Ulungu and Teghem (1997)
  bi-objective unidimensional 01 knapsack
- Visée et al. (1998)
  bi-objective unidimensional 01 knapsack
- Ramos et al. (1998)
  bi-objective minimum weight spanning tree
- Sourd and Spanjaard (2008)
  bi-objective minimum weight spanning tree
- Florios et al. (2010)
  multi-dimensional multi-objective knapsack
- Jorge (2010) (PhD thesis)
  three-objective uni-dimensional 01 knapsack
- Delort (2011) (PhD thesis)
  bi-objective linear assignment
- Parragh and Tricoire (2014) (Technical report)
  bi-objective ream orienteering problem with time windows

MOB&B: the Mixed 01 Linear Case

Presentation

$$\begin{aligned} \min z(x) &= C(x^T, y^T)^T \\ \text{subject to } A(x^T, y^T)^T &= b \\ x &\in \{0,1\}^{n_1} \\ y &\in \mathbb{R}^{n_2} \end{aligned}$$

$$\begin{aligned} x \in \{0,1\}^{n_1} &\longrightarrow n_1 \text{ binary variables} \\ y \in \mathbb{R}^{n_2} &\longrightarrow n_2 \text{ continuous variables} \\ & \quad\;\; (n_1 + n_2 = n) \\ A \in \mathbb{Z}^{m \times n} &\longrightarrow m \text{ constraints} \\ C \in \mathbb{Z}^{p \times n} &\longrightarrow p \text{ objective vectors} \end{aligned}$$

The pioneer algorithm

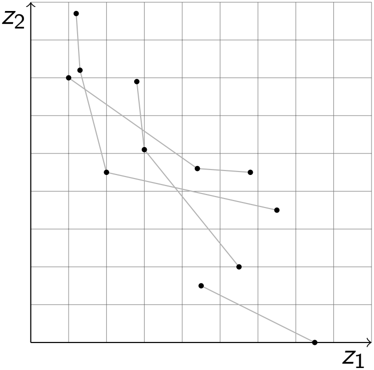
Mavrotas and Diakoulaki (1998, 2005)

- Lower Bound Set for $\bar{Y}_N$:
  Ideal Point of the linear relaxation
- Choice of the active node:
  Depth-first search
- Separation procedure: binary variables fixed to 0 first and next to 1 (in order of index)
- Construction of feasible points:
  When all binary variables are fixed,
  a MOLP is obtained and solved,
  Only extreme points are considered
- Upper Bound Set:
  Restricted incumbent list
- 2005: Final Dominance Test

The pioneer algorithm

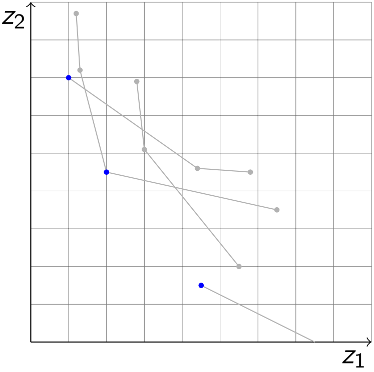
Mavrotas and Diakoulaki (1998, 2005)

- Lower Bound Set for $\bar{Y}_N$:
  Ideal Point of the linear relaxation
- Choice of the active node:
  Depth-first search
- Separation procedure: binary variables fixed to 0 first and next to 1 (in order of index)
- Construction of feasible points:
  When all binary variables are fixed,
  a MOLP is obtained and solved,
  Only extreme points are considered
- Upper Bound Set:
  Restricted incumbent list
- 2005: Final Dominance Test



The pioneer algorithm
Mavrotas and Diakoulaki (1998, 2005)

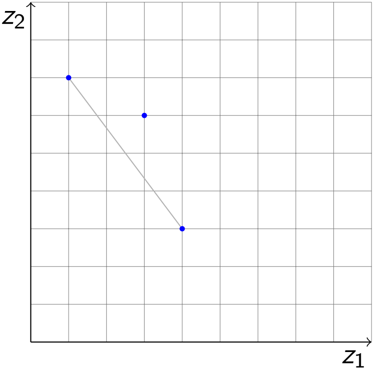
- Lower Bound Set for $\bar{Y}_N$:
  Ideal Point of the linear relaxation
- Choice of the active node:
  Depth-first search
- Separation procedure: binary variables fixed to 0 first and next to 1 (in order of index)
- Construction of feasible points:
  When all binary variables are fixed,
  a MOLP is obtained and solved,
  Only extreme points are considered
- Upper Bound Set:
  Restricted incumbent list
- 2005: Final Dominance Test



The pioneer algorithm

Mavrotas and Diakoulaki (1998, 2005)

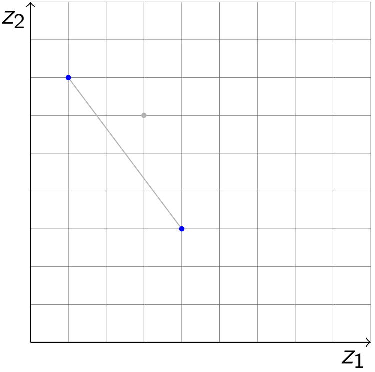
- Lower Bound Set for $\bar{Y}_N$:
  Ideal Point of the linear relaxation
- Choice of the active node:
  Depth-first search
- Separation procedure: binary variables fixed to 0 first and next to 1 (in order of index)
- Construction of feasible points:
  When all binary variables are fixed,
  a MOLP is obtained and solved,
  Only extreme points are considered
- Upper Bound Set:
  Restricted incumbent list
- 2005: Final Dominance Test



The pioneer algorithm

Mavrotas and Diakoulaki (1998, 2005)

- Lower Bound Set for $\bar{Y}_N$:
  Ideal Point of the linear relaxation
- Choice of the active node:
  Depth-first search
- Separation procedure: binary variables fixed to 0 first and next to 1 (in order of index)
- Construction of feasible points:
  When all binary variables are fixed,
  a MOLP is obtained and solved,
  Only extreme points are considered
- Upper Bound Set:
  Restricted incumbent list
- 2005: Final Dominance Test



The first full and correct algorithm with two objectives

Vincent et al. (2013)

Theorem

The nondominated set of a BOMIP is composed of edges (that can be closed, half-open, open or reduced to a point).

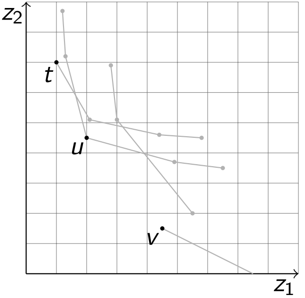


Figure: Representation of D_{ex} with extreme nondominated

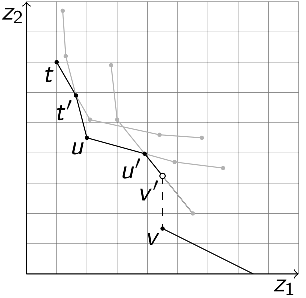


Figure: Representation of Z with extreme and non-extreme

The first full and correct algorithm
Vincent et al. (2013)

- Lower Bound Set for $\bar{Y}_N$: Ideal point of linear relaxation, linear relaxation, Ideal point of convex relaxation, convex relaxation
- Upper Bound Set: Extended Incumbent list
- Construction of feasible points:
  - Initialization of the incumbent list with the computation of Y_{SN1}. Extended with the solutions of MOLPs with obtained set of fixed variables
  - Feasible solutions obtained when all binary variables are fixed
- Choice of the active node: (some kind of) depth-first search
- Separation procedure
  - Consideration of variables x_k by decreasing order of the absolute value of

$$e(k) := (c_k^1 - \mu_1) + (c_k^2 - \mu_2)$$

  - If $e(k) > 0$: Variable fixed to 1 first and next to 0
  - else: Variable fixed to 0 first and next to 1

The first full and correct algorithm

Vincent et al. (2013)

- Lower Bound Set for $\bar{Y}_N$: Ideal point of linear relaxation, linear relaxation, Ideal point of convex relaxation, convex relaxation
- Upper Bound Set: Extended Incumbent list
- Construction of feasible points:
  - Initialization of the incumbent list with the computation of Y_{SN1}, Extended with the solutions of MOLPs with obtained set of fixed variables
  - Feasible solutions obtained when all binary variables are fixed
- Choice of the active node: (some kind of) depth-first search
- Separation procedure
  - Consideration of variables x_k by decreasing order of the absolute value of

$$e(k) := (c_k^1 - \mu_1) + (c_k^2 - \mu_2)$$

  - If $e(k) > 0$: Variable fixed to 1 first and next to 0
  - else: Variable fixed to 0 first and next to 1

The first full and correct algorithm

Vincent et al. (2013)

- Lower Bound Set for $\bar{Y}_N$: Ideal point of linear relaxation, linear relaxation, Ideal point of convex relaxation, convex relaxation
- Upper Bound Set: Extended Incumbent list
- Construction of feasible points:
  - Initialization of the incumbent list with the computation of Y_{SN1}, Extended with the solutions of MOLPs with obtained set of fixed variables
  - Feasible solutions obtained when all binary variables are fixed
- Choice of the active node: (some kind of) depth-first search
- Separation procedure
  - Consideration of variables x_k by decreasing order of the absolute value of

$$e(k) := (c_k^1 - \mu_1) + (c_k^2 - \mu_2)$$

  - If $e(k) > 0$: Variable fixed to 1 first and next to 0
  - else: Variable fixed to 0 first and next to 1

Mixed 01 Linear Case: Entries found in the literature

Overview

- Mavrotas and Diakoulaki (1998, 2005)
- Vincent et al. (2013)
- Stidsen et al. (2014)
  - Bi-objective case
  - One objective function with only binary variables
  - Local Branch and Bound: use of slicing
- Belotti et al. (2013) (Technical report)
  - Bi-objective case: consideration of integer variables rather than binary variables
- Vincent et al. (2013) (PhD thesis)
  - Different strategies applied in a two phase method
  - Three-objective case

Conclusion for the multi-objective branch and bound

- Natural extension of single-objective algorithms but...
- ... initially inefficient using the ideal and the nadir points as bounds
- Promising methods using bound sets
- The strategies for choosing the active node and for the separation procedure remain basic in the published methods
- Other relaxations than the convex or linear relaxations are scarcely studied
- Linear relaxation not tight enough $\Rightarrow$ Development of multi-objective Branch and Cut algorithms

Conclusion for the multi-objective branch and bound

- Natural extension of single-objective algorithms but...
- ... initially inefficient using the ideal and the nadir points as bounds
- Promising methods using bound sets
- The strategies for choosing the active node and for the separation procedure remain basic in the published methods
- Other relaxations than the convex or linear relaxations are scarcely studied
- Linear relaxation not tight enough $\Rightarrow$ Development of multi-objective Branch and Cut algorithms

Conclusion for the multi-objective branch and bound

- Natural extension of single-objective algorithms but...
- ... initially inefficient using the ideal and the nadir points as bounds
- Promising methods using bound sets
- The strategies for choosing the active node and for the separation procedure remain basic in the published methods
- Other relaxations than the convex or linear relaxations are scarcely studied
- Linear relaxation not tight enough $\Rightarrow$ Development of multi-objective Branch and Cut algorithms

Conclusion for the multi-objective branch and bound

- Natural extension of single-objective algorithms but...
- ... initially inefficient using the ideal and the nadir points as bounds
- Promising methods using bound sets
- The strategies for choosing the active node and for the separation procedure remain basic in the published methods
- Other relaxations than the convex or linear relaxations are scarcely studied
- Linear relaxation not tight enough $\Rightarrow$ Development of multi-objective Branch and Cut algorithms

References

- Multicriteria Optimization
  Matthias Ehrgott
  Second edition
  Chapter 9-10

Acknowledgements

- Matthias Ehrgott

  Slides and figures from
  Lecture 4: Multiobjective Combinatorial Optimization