ROAR NET

Multiobjective Optimisation

1st ROAR-NET Training School

Andreia P. Guerreiro
INESC-ID, Portugal

cost
EUROPEAN COOPERATION IN SCIENCE & TECHNOLOGY

Funded by the European Union

Outline

- Multiobjective Optimization
- Preference Articulation
- Solving Multiobjective Optimisation Problems
- Performance Assessment
- API extension to Multiobjective Optimisation
- Concluding Remarks

Examples

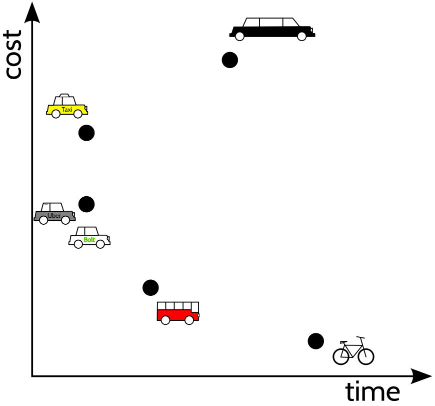
- Book an hotel room
  - Cheapest, most comfortable, closest to the city center

Examples

- Book an hotel room
  - Cheapest, most comfortable, closest to the city center
- Transport for travelling from home to work
  - Cheapest, fastest

Examples

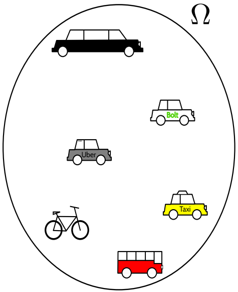
- Book an hotel room
  - Cheapest, most comfortable, closest to the city center
- Transport for travelling from home to work
  - Cheapest, fastest



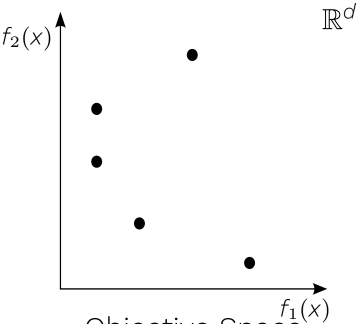
Multiobjective Optimisation

- Considering minimisation:

$$\min_{x \in \Omega} \quad f(x) = (f_1(x), f_2(x), \ldots, f_d(x))$$



Decision Space

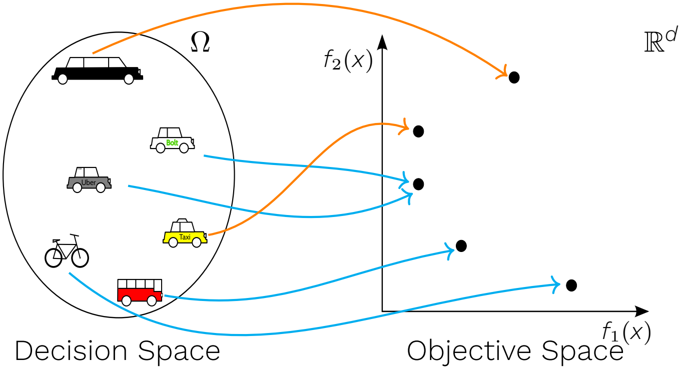


Objective Space

Multiobjective Optimisation

- Considering minimisation:

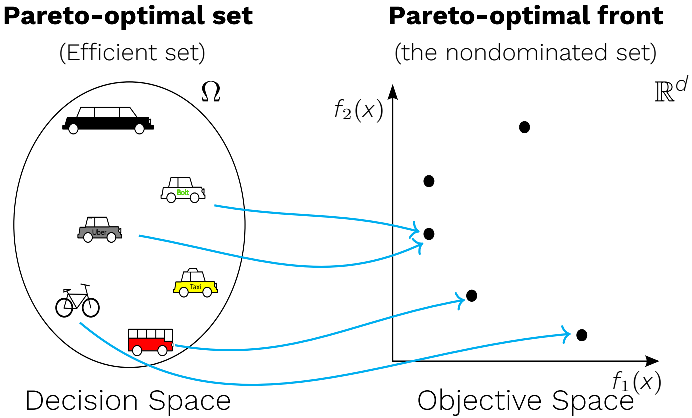
$$\min_{x \in \Omega} \quad f(x) = (f_1(x), f_2(x), \ldots, f_d(x))$$



Multiobjective Optimisation

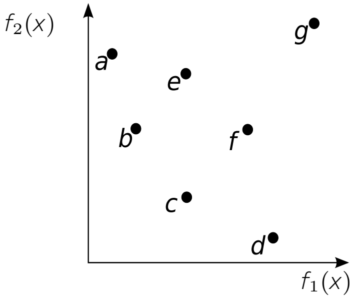
- Considering minimisation:

$$\min_{x \in \Omega} \quad f(x) = (f_1(x), f_2(x), \ldots, f_d(x))$$



Dominance Relations

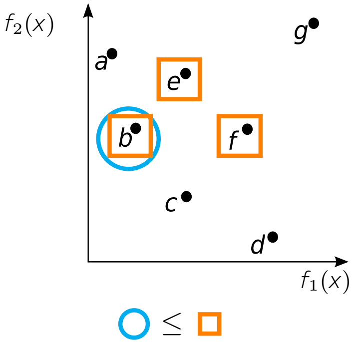
Given $u, v \in \mathbb{R}^d$:



Dominance Relations

Given $u, v \in \mathbb{R}^d$:

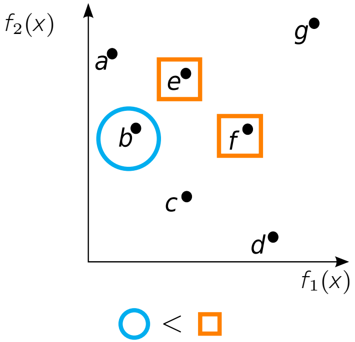
- *u weakly dominates v* ($u \leq v$)
  - If $u_i \leq v_i$ for all $1 \leq i \leq d$



Dominance Relations

Given $u, v \in \mathbb{R}^d$:

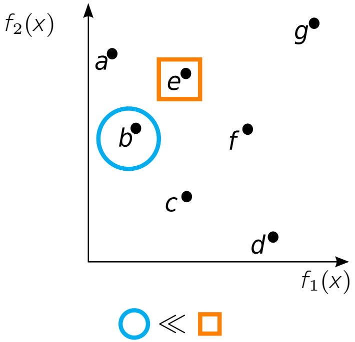
- *u weakly dominates v* ($u \leq v$)
  - If $u_i \leq v_i$ for all $1 \leq i \leq d$
- *u (strictly) dominates v* ($u < v$)
  - If $u \leq v$ and $v \nleq u$



Dominance Relations

Given $u, v \in \mathbb{R}^d$:

- *u weakly dominates v* ($u \leq v$)
  - If $u_i \leq v_i$ for all $1 \leq i \leq d$
- *u (strictly) dominates v* ($u < v$)
  - If $u \leq v$ and $v \nleq u$
- *u stongly dominates v* ($u \ll v$)
  - If $u_i < v_i$ for all $1 \leq i \leq d$



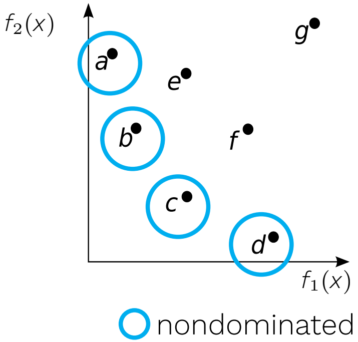
Dominance Relations

Given $u, v \in \mathbb{R}^d$:

- *u weakly dominates v* ($u \leq v$)
  - If $u_i \leq v_i$ for all $1 \leq i \leq d$
- *u (strictly) dominates v* ($u < v$)
  - If $u \leq v$ and $v \nleq u$
- *u stongly dominates v* ($u \ll v$)
  - If $u_i < v_i$ for all $1 \leq i \leq d$

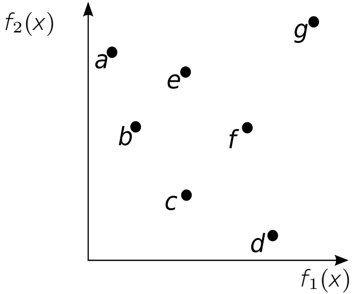
Given $\mathrm{A} \subset \mathbb{R}^d$ and $u \in A$:

- *u is a nondominated point*
  - If there is no $v \in \mathrm{A}$ such that $v < u$



Set-Dominance Relations

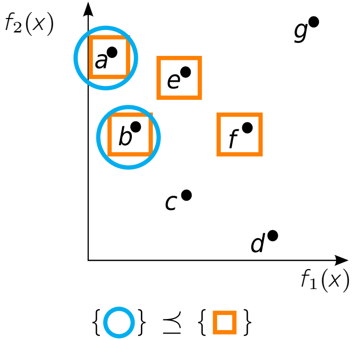
Given $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$:



Set-Dominance Relations

Given $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$:

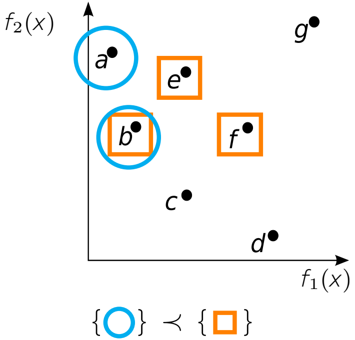
- A *weakly dominates* B $(\mathrm{A} \preceq \mathrm{B})$
  - If $\forall_{b \in \mathrm{B}} \exists_{a \in \mathrm{A}} : a \leq b$



Set-Dominance Relations

Given $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$:

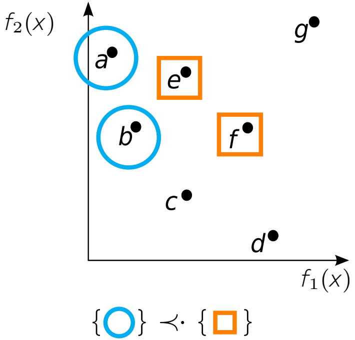
- A *weakly dominates* B ($\mathrm{A} \preceq \mathrm{B}$)
  - If $\forall_{b \in \mathrm{B}} \exists_{a \in \mathrm{A}} : a \leq b$
- A *(strictly) dominates* B ($\mathrm{A} \prec \mathrm{B}$)
  - If $\mathrm{A} \preceq \mathrm{B}$ and $\mathrm{B} \not\preceq \mathrm{A}$



Set-Dominance Relations

Given $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$:

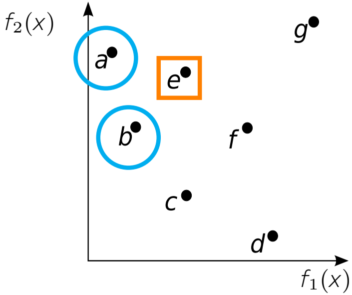
- A *weakly dominates* B $(\mathrm{A} \preceq \mathrm{B})$
  - If $\forall_{b \in \mathrm{B}} \exists_{a \in \mathrm{A}} : a \leq b$
- A *(strictly) dominates* B $(\mathrm{A} \prec \mathrm{B})$
  - If $\mathrm{A} \preceq \mathrm{B}$ and $\mathrm{B} \npreceq \mathrm{A}$
- A *(strictly) dominates* B elementwise $(\mathrm{A} \prec\cdot\, \mathrm{B})$
  - If $\forall_{b \in \mathrm{B}} \exists_{a \in \mathrm{A}} : a < b$



Set-Dominance Relations

Given $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$:

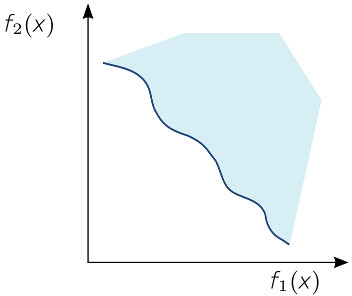
- A *weakly dominates* B ($\mathrm{A} \preceq \mathrm{B}$)
  - If $\forall_{b \in \mathrm{B}} \exists_{a \in \mathrm{A}} : a \leq b$
- A *(strictly) dominates* B ($\mathrm{A} \prec \mathrm{B}$)
  - If $\mathrm{A} \preceq \mathrm{B}$ and $\mathrm{B} \not\preceq \mathrm{A}$
- A *(strictly) dominates* B elementwise ($\mathrm{A} \prec\cdot\, \mathrm{B}$)
  - If $\forall_{b \in \mathrm{B}} \exists_{a \in \mathrm{A}} : a < b$
- A *strongly dominates* B ($\mathrm{A} \prec\!\!\prec \mathrm{B}$)
  - If $\forall_{b \in \mathrm{B}} \exists_{a \in \mathrm{A}} : a \ll b$



$\{\bigcirc\} \prec\!\!\prec \{\square\}$

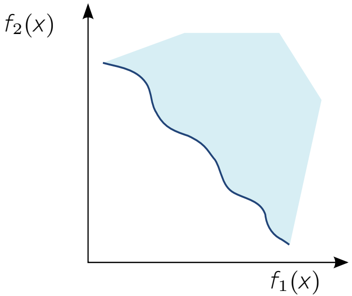
Decision Making

- Conflicting objectives
  - Pareto-optimal front
    - Usually not known
    - Possible infinite



Decision Making

- Conflicting objectives
  - Pareto-optimal front
    - Usually not known
    - Possible infinite
- The best solution depends on the Decision Maker (DM) preferences
  - Subjective information



Preference Information

- Helps discriminate between incomparable solutions

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
    1. **No-preference** methods

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
  1. **No-preference** methods
     - DM has no expectations

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
    1. **No-preference** methods
        - DM has no expectations
        - Return a (any) Pareto-optimal solution

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
  1. **No-preference** methods
     - DM has no expectations
     - Return a (any) Pareto-optimal solution
  2. **A posteriori** methods

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
  1. **No-preference** methods
     - DM has no expectations
     - Return a (any) Pareto-optimal solution
  2. **A posteriori** methods
     - Return the full or an approximation of the Pareto front

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
    1. **No-preference** methods
        - DM has no expectations
        - Return a (any) Pareto-optimal solution
    2. **A posteriori** methods
        - Return the full or an approximation of the Pareto front
        - DM then selects the most preferable solution

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
  1. **No-preference** methods
     - DM has no expectations
     - Return a (any) Pareto-optimal solution
  2. **A posteriori** methods
     - Return the full or an approximation of the Pareto front
     - DM then selects the most preferable solution
  3. **Interactive/progressive** methods

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
  1. **No-preference** methods
     - DM has no expectations
     - Return a (any) Pareto-optimal solution
  2. **A posteriori** methods
     - Return the full or an approximation of the Pareto front
     - DM then selects the most preferable solution
  3. **Interactive/progressive** methods
     - DM periodically expresses preferences during the optimisation process

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
  1. **No-preference** methods
     - DM has no expectations
     - Return a (any) Pareto-optimal solution
  2. **A posteriori** methods
     - Return the full or an approximation of the Pareto front
     - DM then selects the most preferable solution
  3. **Interactive/progressive** methods
     - DM periodically expresses preferences during the optimisation process
     - the search progresses towards the most preferable solution

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
    1. **No-preference** methods
        - DM has no expectations
        - Return a (any) Pareto-optimal solution
    2. **A posteriori** methods
        - Return the full or an approximation of the Pareto front
        - DM then selects the most preferable solution
    3. **Interactive/progressive** methods
        - DM periodically expresses preferences during the optimisation process
        - the search progresses towards the most preferable solution
    4. **A priori** methods

Preference Information

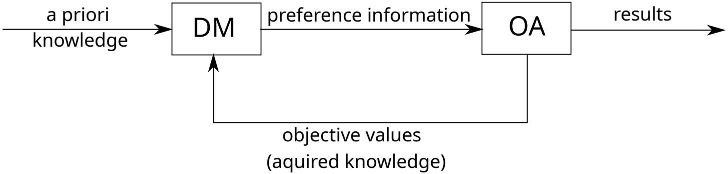
- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
  1. **No-preference** methods
     - DM has no expectations
     - Return a (any) Pareto-optimal solution
  2. **A posteriori** methods
     - Return the full or an approximation of the Pareto front
     - DM then selects the most preferable solution
  3. **Interactive/progressive** methods
     - DM periodically expresses preferences during the optimisation process
     - the search progresses towards the most preferable solution
  4. **A priori** methods
     - DM provides preference information pior to the optimisation

Preference Information

- Helps discriminate between incomparable solutions
- May be provided at different stages of the optimisation process
- Optimisation methods are classified according to when it is articulated
  1. **No-preference** methods
     - DM has no expectations
     - Return a (any) Pareto-optimal solution
  2. **A posteriori** methods
     - Return the full or an approximation of the Pareto front
     - DM then selects the most preferable solution
  3. **Interactive/progressive** methods
     - DM periodically expresses preferences during the optimisation process
     - the search progresses towards the most preferable solution
  4. **A priori** methods
     - DM provides preference information pior to the optimisation
     - Focus on solutions on the preferable regions

Preference Articulation as a Process

- The optimisation process does not have to be static
- Preference articulation can be viewed as a process



Preference Modelling

Useful for

- Favouring one of two incomparable solutions

Preference Modelling

Useful for

- Favouring one of two incomparable solutions
- Handling constraints

Preference Modelling

Useful for

- Favouring one of two incomparable solutions
- Handling constraints

Methods

- Utility/scalarisation functions
    - Function that maps each solution to a value
    - Examples

Preference Modelling

Useful for

- Favouring one of two incomparable solutions
- Handling constraints

Methods

- Utility/scalarisation functions
  - Function that maps each solution to a value
  - Examples
    - Weight-based (such as weighted sum)

Preference Modelling

Useful for

- Favouring one of two incomparable solutions
- Handling constraints

Methods

- Utility/scalarisation functions
  - Function that maps each solution to a value
  - Examples
    - Weight-based (such as weighted sum)
    - reference-point based

Preference Modelling

Useful for

- Favouring one of two incomparable solutions
- Handling constraints

Methods

- Utility/scalarisation functions
  - Function that maps each solution to a value
  - Examples
    - Weight-based (such as weighted sum)
    - reference-point based
- Binary relation

Preference Modelling

Useful for

- Favouring one of two incomparable solutions
- Handling constraints

Methods

- Utility/scalarisation functions
  - Function that maps each solution to a value
  - Examples
    - Weight-based (such as weighted sum)
    - reference-point based
- Binary relation
  - Pairwise comparison between solution

Preference Modelling

Useful for

- Favouring one of two incomparable solutions
- Handling constraints

Methods

- Utility/scalarisation functions
  - Function that maps each solution to a value
  - Examples
    - Weight-based (such as weighted sum)
    - reference-point based
- Binary relation
  - Pairwise comparison between solution
  - Some solutions may remain incomparable

Preference Modelling

Useful for

- Favouring one of two incomparable solutions
- Handling constraints

Methods

- Utility/scalarisation functions
  - Function that maps each solution to a value
  - Examples
    - Weight-based (such as weighted sum)
    - reference-point based
- Binary relation
  - Pairwise comparison between solution
  - Some solutions may remain incomparable
  - Should respect Pareto dominance

Preference Modelling

Useful for

- Favouring one of two incomparable solutions
- Handling constraints

Methods

- Utility/scalarisation functions
  - Function that maps each solution to a value
  - Examples
    - Weight-based (such as weighted sum)
    - reference-point based
- Binary relation
  - Pairwise comparison between solution
  - Some solutions may remain incomparable
  - Should respect Pareto dominance
  - Examples

Preference Modelling

Useful for

- Favouring one of two incomparable solutions
- Handling constraints

Methods

- Utility/scalarisation functions
  - Function that maps each solution to a value
  - Examples
    - Weight-based (such as weighted sum)
    - reference-point based
- Binary relation
  - Pairwise comparison between solution
  - Some solutions may remain incomparable
  - Should respect Pareto dominance
  - Examples
    - lexicographic order

Preference Modelling

Useful for

- Favouring one of two incomparable solutions
- Handling constraints

Methods

- Utility/scalarisation functions
  - Function that maps each solution to a value
  - Examples
    - Weight-based (such as weighted sum)
    - reference-point based
- Binary relation
  - Pairwise comparison between solution
  - Some solutions may remain incomparable
  - Should respect Pareto dominance
  - Examples
    - lexicographic order
    - preferability relation

Approaches

- Aim of (a posteriori) multiobjective optimisation algorithms

Approaches

- Aim of (a posteriori) multiobjective optimisation algorithms
    1. Find the whole Pareto front

Approaches

- Aim of (a posteriori) multiobjective optimisation algorithms
  1. Find the whole Pareto front
  2. Find a ”good” subset of the Pareto front (a representation)

Approaches

ROAR NET

- Aim of (a posteriori) multiobjective optimisation algorithms
    1. Find the whole Pareto front
    2. Find a ”good” subset of the Pareto front (a representation)
    3. Find a ”good” approximation of the Pareto front

Approaches

- Aim of (a posteriori) multiobjective optimisation algorithms
  1. Find the whole Pareto front
  2. Find a ”good” subset of the Pareto front (a representation)
  3. Find a ”good” approximation of the Pareto front
- Guarantees regarding the outcome set

Approaches

- Aim of (a posteriori) multiobjective optimisation algorithms
  1. Find the whole Pareto front
  2. Find a ”good” subset of the Pareto front (a representation)
  3. Find a ”good” approximation of the Pareto front
- Guarantees regarding the outcome set
  - The quality of each solution

Approaches

- Aim of (a posteriori) multiobjective optimisation algorithms
    1. Find the whole Pareto front
    2. Find a ”good” subset of the Pareto front (a representation)
    3. Find a ”good” approximation of the Pareto front
- Guarantees regarding the outcome set
    - The quality of each solution
        - is it efficient or not?

Approaches

- Aim of (a posteriori) multiobjective optimisation algorithms
  1. Find the whole Pareto front
  2. Find a ”good” subset of the Pareto front (a representation)
  3. Find a ”good” approximation of the Pareto front
- Guarantees regarding the outcome set
  - The quality of each solution
    - is it efficient or not?
  - The quality of the set of solutions

Approaches

- Aim of (a posteriori) multiobjective optimisation algorithms
  1. Find the whole Pareto front
  2. Find a ”good” subset of the Pareto front (a representation)
  3. Find a ”good” approximation of the Pareto front
- Guarantees regarding the outcome set
  - The quality of each solution
    - is it efficient or not?
  - The quality of the set of solutions
    - How well does it cover the Pareto front?

Approaches

- Aim of (a posteriori) multiobjective optimisation algorithms
  1. Find the whole Pareto front
  2. Find a ”good” subset of the Pareto front (a representation)
  3. Find a ”good” approximation of the Pareto front
- Guarantees regarding the outcome set
  - The quality of each solution
    - is it efficient or not?
  - The quality of the set of solutions
    - How well does it cover the Pareto front?
    - How well are the points distributed along the Pareto front?

Approaches

- Aim of (a posteriori) multiobjective optimisation algorithms
  1. Find the whole Pareto front
  2. Find a ”good” subset of the Pareto front (a representation)
  3. Find a ”good” approximation of the Pareto front
- Guarantees regarding the outcome set
  - The quality of each solution
    - is it efficient or not?
  - The quality of the set of solutions
    - How well does it cover the Pareto front?
    - How well are the points distributed along the Pareto front?
    - Does it provide an approximation guarantee?

Approaches

- Aim of (a posteriori) multiobjective optimisation algorithms
  1. Find the whole Pareto front
  2. Find a ”good” subset of the Pareto front (a representation)
  3. Find a ”good” approximation of the Pareto front
- Guarantees regarding the outcome set
  - The quality of each solution
    - is it efficient or not?
  - The quality of the set of solutions
    - How well does it cover the Pareto front?
    - How well are the points distributed along the Pareto front?
    - Does it provide an approximation guarantee?
  - Provide bounds on the location of the Pareto front

Methods

- Solve a sequence of single-objective problems

Methods

- Solve a sequence of single-objective problems
  - e.g., Scalarisations

Methods

- Solve a sequence of single-objective problems
  - e.g., Scalarisations
- Constructive search approaches

Methods

- Solve a sequence of single-objective problems
  - e.g., Scalarisations
- Constructive search approaches
  - e.g., Multiobjective branch-and-bound

Methods

- Solve a sequence of single-objective problems
  - e.g., Scalarisations
- Constructive search approaches
  - e.g., Multiobjective branch-and-bound
- Population-based approaches

Methods

- Solve a sequence of single-objective problems
  - e.g., Scalarisations
- Constructive search approaches
  - e.g., Multiobjective branch-and-bound
- Population-based approaches
  - e.g., Evolutionary multiobjective optimisation algorithms

Scalarisations

- Transform the multiobjective problem into a (parameterised) single-objective problem

Scalarisations

- Transform the multiobjective problem into a (parameterised) single-objective problem
- Solve multiple times with different scalarisation parameters

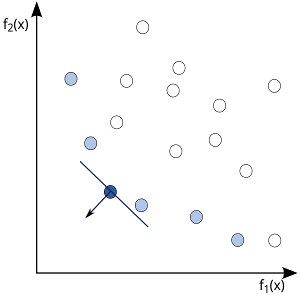
Scalarisations

- Transform the multiobjective problem into a (parameterised) single-objective problem
- Solve multiple times with different scalarisation parameters
- Examples of scalarisations

Scalarisations

- Transform the multiobjective problem into a (parameterised) single-objective problem
- Solve multiple times with different scalarisation parameters
- Examples of scalarisations
    - Weighted sum scalarisation

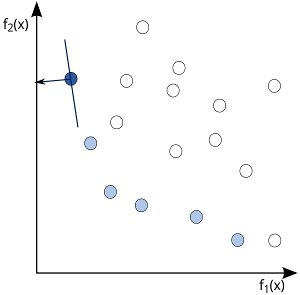
$$\min_{x \in \Omega} \sum_{i=1}^{d} w_i f_i(x)$$



Scalarisations

- Transform the multiobjective problem into a (parameterised) single-objective problem
- Solve multiple times with different scalarisation parameters
- Examples of scalarisations
  - Weighted sum scalarisation

$$\min_{x \in \Omega} \sum_{i=1}^{d} w_i f_i(x)$$



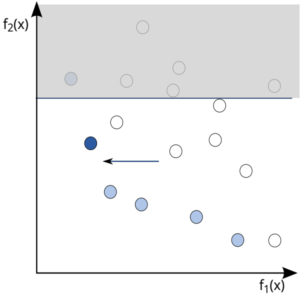
Scalarisations

- Transform the multiobjective problem into a (parameterised) single-objective problem
- Solve multiple times with different scalarisation parameters
- Examples of scalarisations
  - Weighted sum scalarisation

$$\min_{x \in \Omega} \sum_{i=1}^{d} w_i f_i(x)$$

  - ε-constraint scalarisation

$$\min_{x \in \Omega} f_1(x)$$
$$\text{s.t.} \quad f_i(x) \leqslant \varepsilon_i, \; i = 2, \ldots, d$$



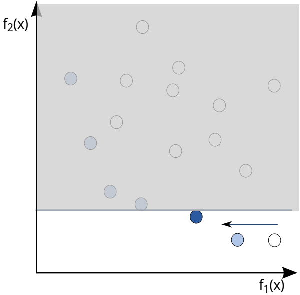
Scalarisations

- Transform the multiobjective problem into a (parameterised) single-objective problem
- Solve multiple times with different scalarisation parameters
- Examples of scalarisations
  - Weighted sum scalarisation

$$\min_{x \in \Omega} \sum_{i=1}^{d} w_i f_i(x)$$

  - ε-constraint scalarisation

$$\min_{x \in \Omega} f_1(x)$$
$$\text{s.t.} \quad f_i(x) \leqslant \varepsilon_i, \; i = 2, \ldots, d$$



Scalarisations

- Transform the multiobjective problem into a (parameterised) single-objective problem
- Solve multiple times with different scalarisation parameters
- Examples of scalarisations
- Desirable properties

Scalarisations

- Transform the multiobjective problem into a (parameterised) single-objective problem
- Solve multiple times with different scalarisation parameters
- Examples of scalarisations
- Desirable properties
  - Correctness: Optimal solutions of a scalarisation are always (weakly) efficient solutions

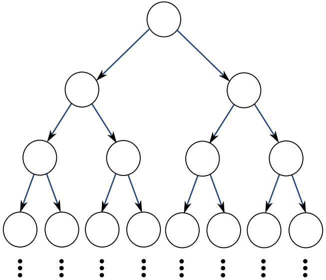
Scalarisations

- Transform the multiobjective problem into a (parameterised) single-objective problem
- Solve multiple times with different scalarisation parameters
- Examples of scalarisations
- Desirable properties
  - Correctness: Optimal solutions of a scalarisation are always (weakly) efficient solutions
  - Completeness: Every point in the Pareto front can be determined by an appropriate choice of the parameters

Multiobjective Branch-and-Bound

- Branching

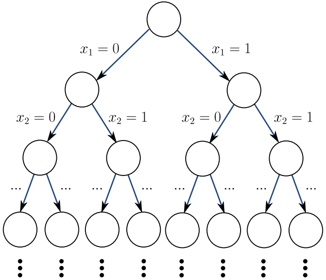
- Bounding



Multiobjective Branch-and-Bound

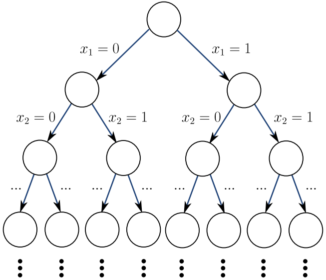
- Branching
  - Branching strategy (e.g., choose next branching variable x_i)

- Bounding



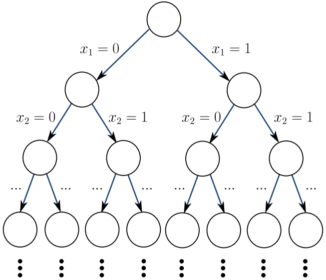
Multiobjective Branch-and-Bound

- Branching
  - Branching strategy (e.g., choose next branching variable x_i)
  - Partitioning procedure (e.g., set branching variable to 0 or 1)
- Bounding



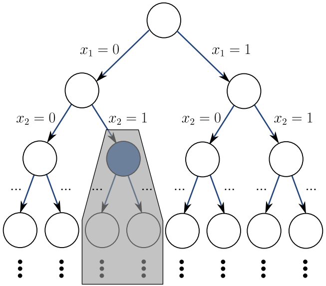
Multiobjective Branch-and-Bound

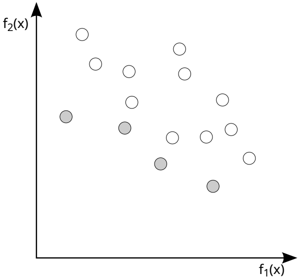
- Branching
  - Branching strategy (e.g., choose next branching variable x_i)
  - Partitioning procedure (e.g., set branching variable to 0 or 1)
  - Static vs dynamic
- Bounding



Multiobjective Branch-and-Bound

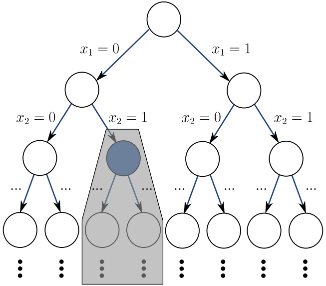
- Branching
  - Branching strategy (e.g., choose next branching variable x_i)
  - Partitioning procedure (e.g., set branching variable to 0 or 1)
  - Static vs dynamic
- Bounding

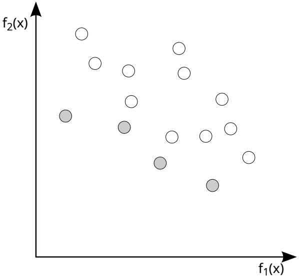




Multiobjective Branch-and-Bound

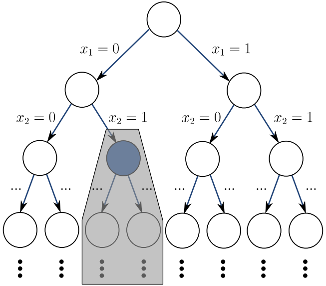
- Branching
  - Branching strategy (e.g., choose next branching variable x_i)
  - Partitioning procedure (e.g., set branching variable to 0 or 1)
  - Static vs dynamic
- Bounding (in the single-objective case)
  - Lower bound on the objective **values** of the solutions in the subtree

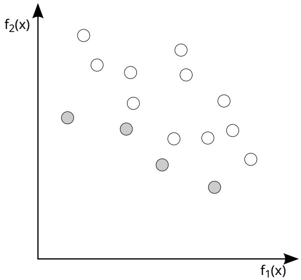




Multiobjective Branch-and-Bound

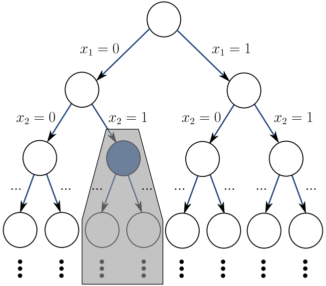
- Branching
  - Branching strategy (e.g., choose next branching variable x_i)
  - Partitioning procedure (e.g., set branching variable to 0 or 1)
  - Static vs dynamic
- Bounding (in the single-objective case)
  - Lower bound on the objective **values** of the solutions in the subtree
  - Upper bound on the objective **value** of the optimal solution

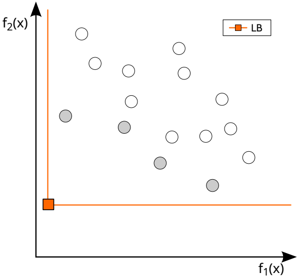




Multiobjective Branch-and-Bound

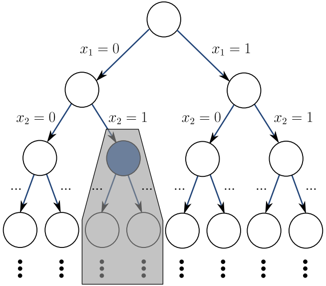
- Branching
  - Branching strategy (e.g., choose next branching variable x_i)
  - Partitioning procedure (e.g., set branching variable to 0 or 1)
  - Static vs dynamic
- Bounding (in the multiobjective case)
  - Lower bound **set** on the objective **points** of the solutions in the subtree
  - Upper bound **set** on the **Pareto front**

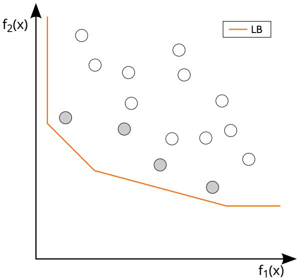




Multiobjective Branch-and-Bound

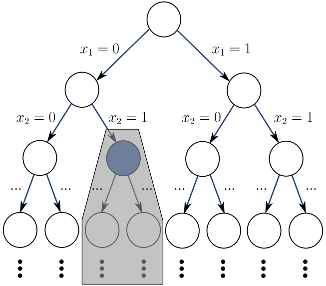
- Branching
  - Branching strategy (e.g., choose next branching variable x_i)
  - Partitioning procedure (e.g., set branching variable to 0 or 1)
  - Static vs dynamic
- Bounding (in the multiobjective case)
  - Lower bound **set** on the objective **points** of the solutions in the subtree
  - Upper bound **set** on the **Pareto front**

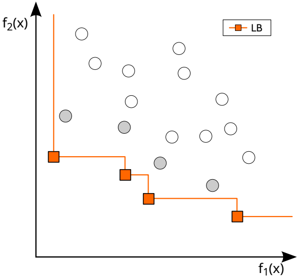




Multiobjective Branch-and-Bound

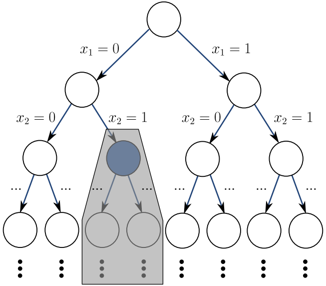
- Branching
  - Branching strategy (e.g., choose next branching variable x_i)
  - Partitioning procedure (e.g., set branching variable to 0 or 1)
  - Static vs dynamic
- Bounding (in the multiobjective case)
  - Lower bound **set** on the objective **points** of the solutions in the subtree
  - Upper bound **set** on the **Pareto front**

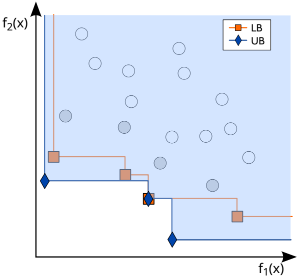




Multiobjective Branch-and-Bound

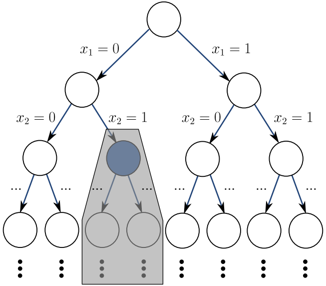
- Branching
  - Branching strategy (e.g., choose next branching variable x_i)
  - Partitioning procedure (e.g., set branching variable to 0 or 1)
  - Static vs dynamic
- Bounding (in the multiobjective case)
  - Lower bound **set** on the objective **points** of the solutions in the subtree
  - Upper bound **set** on the **Pareto front**

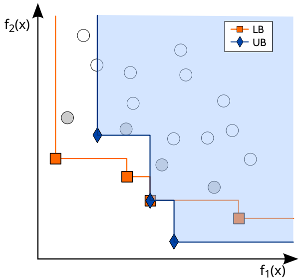




Multiobjective Branch-and-Bound

- Branching
  - Branching strategy (e.g., choose next branching variable x_i)
  - Partitioning procedure (e.g., set branching variable to 0 or 1)
  - Static vs dynamic
- Bounding (in the multiobjective case)
  - Lower bound **set** on the objective **points** of the solutions in the subtree
  - Upper bound **set** on the **Pareto front**



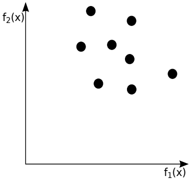


Evolutionary Multiobjective Optimisation (EMO)

- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection

Evolutionary Multiobjective Optimisation (EMO)

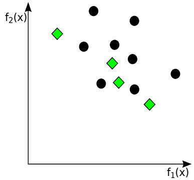
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



Initial population (generation 0)

Evolutionary Multiobjective Optimisation (EMO)

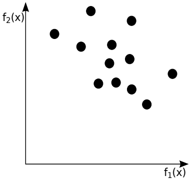
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



Generate offspring

Evolutionary Multiobjective Optimisation (EMO)

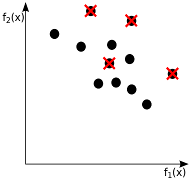
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



(Environmental) selection

Evolutionary Multiobjective Optimisation (EMO)

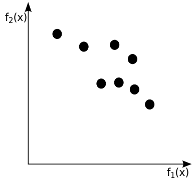
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



(Environmental) selection:
Discard solutions

Evolutionary Multiobjective Optimisation (EMO)

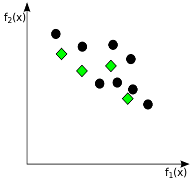
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



Population (generation 1)

Evolutionary Multiobjective Optimisation (EMO)

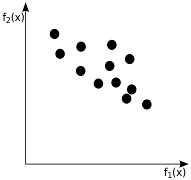
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



Generate offspring

Evolutionary Multiobjective Optimisation (EMO)

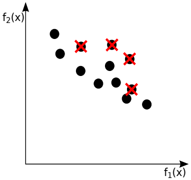
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



(Environmental) selection

Evolutionary Multiobjective Optimisation (EMO)

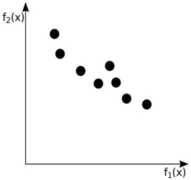
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



(Environmental) selection:
Discard solutions

Evolutionary Multiobjective Optimisation (EMO)

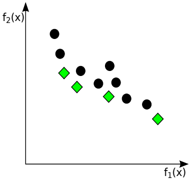
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



Population (generation 2)

Evolutionary Multiobjective Optimisation (EMO)

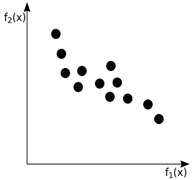
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



Generate offspring

Evolutionary Multiobjective Optimisation (EMO)

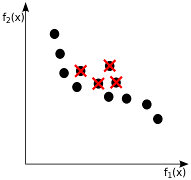
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



(Environmental) selection

Evolutionary Multiobjective Optimisation (EMO)

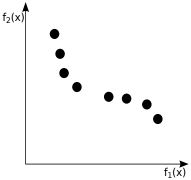
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



(Environmental) selection:
Discard solutions

Evolutionary Multiobjective Optimisation (EMO)

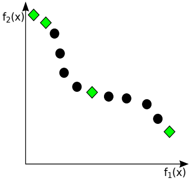
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



Population (generation 3)

Evolutionary Multiobjective Optimisation (EMO)

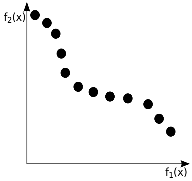
- Evolutionary Algorithms (EAs)
    - Inspired on natural selection
    - Well-suited for multiobjective optimisation
    - Search for multiple nondominated solutions simultaneously
        - Generation of new solutions
        - Selection



Generate offspring

Evolutionary Multiobjective Optimisation (EMO)

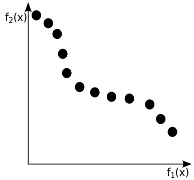
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection



(Environmental) selection:
Which solutions to discard?

Evolutionary Multiobjective Optimisation (EMO)

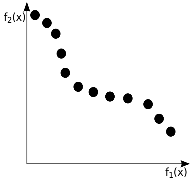
- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection
- Preference information
  - Discriminate incomparable solutions



(Environmental) selection:
Which solutions to discard?

Evolutionary Multiobjective Optimisation (EMO)

- Evolutionary Algorithms (EAs)
  - Inspired on natural selection
  - Well-suited for multiobjective optimisation
  - Search for multiple nondominated solutions simultaneously
    - Generation of new solutions
    - Selection
- Preference information
  - Discriminate incomparable solutions
  - When not available
    - Keep a diverse set of solutions



(Environmental) selection:
Which solutions to discard?

Selection in EMO

- Crucial to the EA’s ability to approximate the Pareto front as a whole

Selection in EMO

- Crucial to the EA's ability to approximate the Pareto front as a whole
- Main concerns:

Selection in EMO

- Crucial to the EA's ability to approximate the Pareto front as a whole
- Main concerns:
  - Select the best solutions

Selection in EMO

- Crucial to the EA's ability to approximate the Pareto front as a whole
- Main concerns:
    - Select the best solutions
    - Keep solutions spread along the Pareto Front

Selection in EMO

- Crucial to the EA's ability to approximate the Pareto front as a whole
- Main concerns:
  - Select the best solutions
  - Keep solutions spread along the Pareto Front
- Two different approaches:

Selection in EMO

- Crucial to the EA's ability to approximate the Pareto front as a whole
- Main concerns:
  - Select the best solutions
  - Keep solutions spread along the Pareto Front
- Two different approaches:
  - Focused on individual quality

Selection in EMO

- Crucial to the EA's ability to approximate the Pareto front as a whole
- Main concerns:
  - Select the best solutions
  - Keep solutions spread along the Pareto Front
- Two different approaches:
  - Focused on individual quality
    - Select the best individuals

Selection in EMO

- Crucial to the EA's ability to approximate the Pareto front as a whole
- Main concerns:
  - Select the best solutions
  - Keep solutions spread along the Pareto Front
- Two different approaches:
  - Focused on individual quality
    - Select the best individuals
    - Then, use diversity preservation techniques

Selection in EMO

- Crucial to the EA's ability to approximate the Pareto front as a whole
- Main concerns:
  - Select the best solutions
  - Keep solutions spread along the Pareto Front
- Two different approaches:
  - Focused on individual quality
    - Select the best individuals
    - Then, use diversity preservation techniques
  - Focused on set quality

Selection in EMO

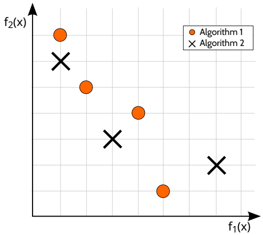
- Crucial to the EA's ability to approximate the Pareto front as a whole
- Main concerns:
  - Select the best solutions
  - Keep solutions spread along the Pareto Front
- Two different approaches:
  - Focused on individual quality
    - Select the best individuals
    - Then, use diversity preservation techniques
  - Focused on set quality
    - Based on (set-)quality indicators

Selection in EMO

- Crucial to the EA's ability to approximate the Pareto front as a whole
- Main concerns:
  - Select the best solutions
  - Keep solutions spread along the Pareto Front
- Two different approaches:
  - Focused on individual quality
    - Select the best individuals
    - Then, use diversity preservation techniques
  - Focused on set quality
    - Based on (set-)quality indicators
    - Given a set of n solutions find the subset of k solutions that maximizes a given quality indicator

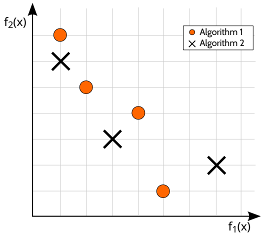
Performance Assessment

- Different optimisation algorithms usually produce different results



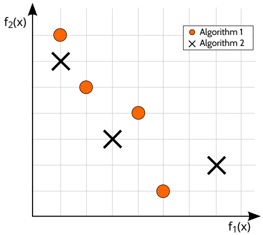
Performance Assessment

- Different optimisation algorithms usually produce different results
  - Different runs of the same algorithm may produce different results (e.g., stochastic algorithms)



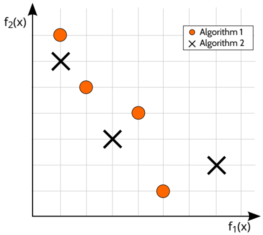
Performance Assessment

- Different optimisation algorithms usually produce different results
  - Different runs of the same algorithm may produce different results (e.g., stochastic algorithms)
- The solutions produced in one optimisation run may be incomparable to those produced in another run



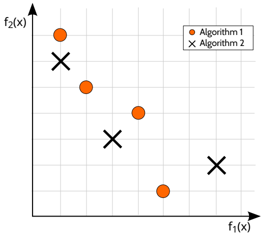
Performance Assessment

- Different optimisation algorithms usually produce different results
  - Different runs of the same algorithm may produce different results (e.g., stochastic algorithms)
- The solutions produced in one optimisation run may be incomparable to those produced in another run
- Two types of methods to assess the performance of multiobjective optimisation algorithms



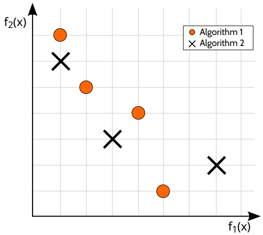
Performance Assessment

- Different optimisation algorithms usually produce different results
  - Different runs of the same algorithm may produce different results (e.g., stochastic algorithms)
- The solutions produced in one optimisation run may be incomparable to those produced in another run
- Two types of methods to assess the performance of multiobjective optimisation algorithms
  - (Set-)Quality Indicators (e.g., the Hypervolume Indicator)



Performance Assessment

- Different optimisation algorithms usually produce different results
  - Different runs of the same algorithm may produce different results (e.g., stochastic algorithms)
- The solutions produced in one optimisation run may be incomparable to those produced in another run
- Two types of methods to assess the performance of multiobjective optimisation algorithms
  - (Set-)Quality Indicators (e.g., the Hypervolume Indicator)
  - Empirical Attainment Function

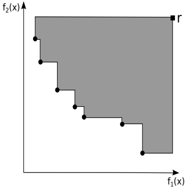


(Set-)Quality indicators

- A (set-)quality indicator $\mathcal{I}$ is a function that maps a set of points in the objective space to a real value

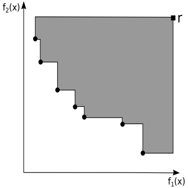
(Set-)Quality indicators

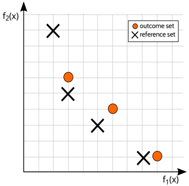
- A (set-)quality indicator $\mathcal{I}$ is a function that maps a set of points in the objective space to a real value
- Examples:
  - The hypervolume indicator



(Set-)Quality indicators

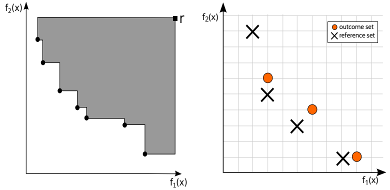
- A (set-)quality indicator $\mathcal{I}$ is a function that maps a set of points in the objective space to a real value
- Examples:
  - The hypervolume indicator
  - (additive) ε-indicator





(Set-)Quality indicators

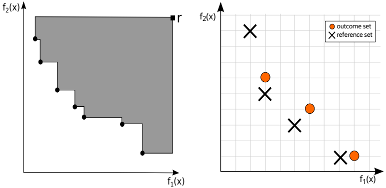
- A (set-)quality indicator $\mathcal{I}$ is a function that maps a set of points in the objective space to a real value
- Examples:
  - The hypervolume indicator
  - (additive) ε-indicator



- Performance assessment

(Set-)Quality indicators

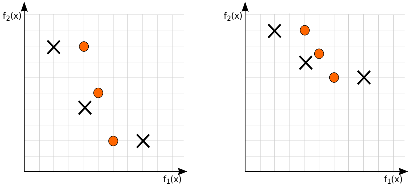
- A (set-)quality indicator $\mathcal{I}$ is a function that maps a set of points in the objective space to a real value
- Examples:
  - The hypervolume indicator
  - (additive) ε-indicator



- Performance assessment
- Guide the search process (subset selection view)
  - Find a subset of k solutions that maximizes a given quality indicator

Properties of Quality Indicators

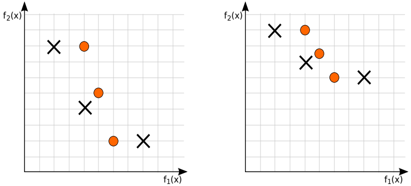
Scaling invariance/independence



$A_1 = \{\times\},\ B_1 = \{\bullet\}$ $\qquad$ $A_2 = \{\times\},\ B_2 = \{\bullet\}$

Properties of Quality Indicators

Scaling invariance/independence

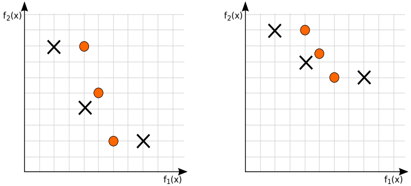


$A_1 = \{\times\}, B_1 = \{\bullet\}$ $\quad$ $A_2 = \{\times\}, B_2 = \{\bullet\}$

Scaling invariance: $\mathcal{I}(A_1) = \mathcal{I}(A_2)$ and $\mathcal{I}(B_1) = \mathcal{I}(B_2)$

Properties of Quality Indicators

Scaling invariance/independence



$A_1 = \{\times\}, B_1 = \{\bullet\}$ $A_2 = \{\times\}, B_2 = \{\bullet\}$

Scaling invariance: $\mathcal{I}(A_1) = \mathcal{I}(A_2)$ and $\mathcal{I}(B_1) = \mathcal{I}(B_2)$

Scaling independence: $\mathcal{I}(A_1) \geq \mathcal{I}(B_1) \Rightarrow \mathcal{I}(A_2) \geq \mathcal{I}(B_2)$

Properties of Quality Indicators

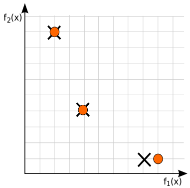
- Monotonicity
    - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)

Properties of Quality Indicators

- Monotonicity
  - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)
  - A set-indicator $\mathcal{I}$ is weakly $\mathcal{R}$-monotonic if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A} \; \mathcal{R} \; \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) \geq \mathcal{I}(\mathrm{B})$.

Properties of Quality Indicators

- Monotonicity
  - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)
  - A set-indicator $\mathcal{I}$ is weakly $\mathcal{R}$-monotonic if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A}\ \mathcal{R}\ \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) \geq \mathcal{I}(\mathrm{B})$.
    - $\mathcal{I}$ is weakly $\preceq$-monotonic



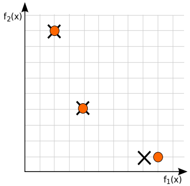
$$\{\times\} \preceq \{\bullet\} \Rightarrow \mathcal{I}(\{\times\}) \geqslant \mathcal{I}(\{\bullet\})$$

Properties of Quality Indicators

- Monotonicity
  - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)
  - A set-indicator $\mathcal{I}$ is weakly $\mathcal{R}$-monotonic if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A} \;\mathcal{R}\; \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) \geq \mathcal{I}(\mathrm{B})$.
    - $\mathcal{I}$ is weakly $\preceq$-monotonic
  - If $\mathrm{A} \;\mathcal{R}\; \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) > \mathcal{I}(\mathrm{B})$, the indicator is strictly $\mathcal{R}$-monotonic

Properties of Quality Indicators

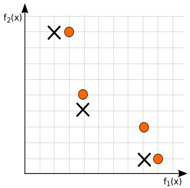
- Monotonicity
  - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)
  - A set-indicator $\mathcal{I}$ is weakly $\mathcal{R}$-monotonic if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A} \; \mathcal{R} \; \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) \geq \mathcal{I}(\mathrm{B})$.
    - $\mathcal{I}$ is weakly $\preceq$-monotonic
  - If $\mathrm{A} \; \mathcal{R} \; \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) > \mathcal{I}(\mathrm{B})$, the indicator is strictly $\mathcal{R}$-monotonic
    - $\mathcal{I}$ is strictly $\prec$-monotonic



$$\{\times\} \prec \{\bullet\} \;\Rightarrow\; \mathcal{I}(\{\times\}) > \mathcal{I}(\{\bullet\})$$

Properties of Quality Indicators

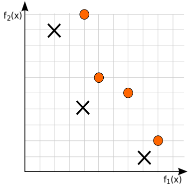
- Monotonicity
  - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)
  - A set-indicator $\mathcal{I}$ is weakly $\mathcal{R}$-monotonic if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A}\ \mathcal{R}\ \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) \geq \mathcal{I}(\mathrm{B})$.
    - $\mathcal{I}$ is weakly $\preceq$-monotonic
  - If $\mathrm{A}\ \mathcal{R}\ \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) > \mathcal{I}(\mathrm{B})$, the indicator is strictly $\mathcal{R}$-monotonic
    - $\mathcal{I}$ is strictly $\prec$-monotonic
    - $\mathcal{I}$ is strictly $\prec\!\cdot$-monotonic



$$\{\times\} \prec\!\cdot\ \{\bullet\} \Rightarrow \mathcal{I}(\{\times\}) > \mathcal{I}(\{\bullet\})$$

Properties of Quality Indicators

- Monotonicity
  - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)
  - A set-indicator $\mathcal{I}$ is weakly $\mathcal{R}$-monotonic if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A}\ \mathcal{R}\ \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) \geq \mathcal{I}(\mathrm{B})$.
    - $\mathcal{I}$ is weakly $\preceq$-monotonic
  - If $\mathrm{A}\ \mathcal{R}\ \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) > \mathcal{I}(\mathrm{B})$, the indicator is strictly $\mathcal{R}$-monotonic
    - $\mathcal{I}$ is strictly $\prec$-monotonic
    - $\mathcal{I}$ is strictly $\prec\!\cdot$-monotonic
    - $\mathcal{I}$ is strictly $\prec\!\!\prec$-monotonic



$$\{\times\} \prec\!\!\prec \{\bullet\} \Rightarrow \mathcal{I}(\{\times\}) > \mathcal{I}(\{\bullet\})$$

Properties of Quality Indicators

- Monotonicity
  - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)
  - A set-indicator $\mathcal{I}$ is weakly $\mathcal{R}$-monotonic if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A} \; \mathcal{R} \; \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) \geq \mathcal{I}(\mathrm{B})$.
    - $\mathcal{I}$ is weakly $\preceq$-monotonic
  - If $\mathrm{A} \; \mathcal{R} \; \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) > \mathcal{I}(\mathrm{B})$, the indicator is strictly $\mathcal{R}$-monotonic
    - $\mathcal{I}$ is strictly $\prec$-monotonic
    - $\mathcal{I}$ is strictly $\prec\!\cdot$-monotonic
    - $\mathcal{I}$ is strictly $\prec\!\!\prec$-monotonic
  - Allow to infer a lower bound on the number of Pareto-optimal solution in any subset maximizing the indicator $\mathcal{I}$

Properties of Quality Indicators

- Monotonicity
  - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)
  - A set-indicator $\mathcal{I}$ is weakly $\mathcal{R}$-monotonic if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A} \; \mathcal{R} \; \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) \geq \mathcal{I}(\mathrm{B})$.
    - $\mathcal{I}$ is weakly $\preceq$-monotonic $\Rightarrow \exists_{S \in \mathcal{A}} \; |S \cap P| = \min(k, |P|)$
  - If $\mathrm{A} \; \mathcal{R} \; \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) > \mathcal{I}(\mathrm{B})$, the indicator is strictly $\mathcal{R}$-monotonic
    - $\mathcal{I}$ is strictly $\prec$-monotonic
    - $\mathcal{I}$ is strictly $\prec\cdot$-monotonic
    - $\mathcal{I}$ is strictly $\prec\!\!\prec$-monotonic
  - Allow to infer a lower bound on the number of Pareto-optimal solution in any subset maximizing the indicator $\mathcal{I}$

Properties of Quality Indicators

- Monotonicity
  - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)
  - A set-indicator $\mathcal{I}$ is weakly $\mathcal{R}$-monotonic if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A} \; \mathcal{R} \; \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) \geq \mathcal{I}(\mathrm{B})$.
    - $\mathcal{I}$ is weakly $\preceq$-monotonic $\Rightarrow \exists_{S \in \mathcal{A}} \; |S \cap P| = \min(k, |P|)$
  - If $\mathrm{A} \; \mathcal{R} \; \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) > \mathcal{I}(\mathrm{B})$, the indicator is strictly $\mathcal{R}$-monotonic
    - $\mathcal{I}$ is strictly $\prec$-monotonic $\Rightarrow \forall_{S \in \mathcal{A}} \; |S \cap P| = \min(k, |P|)$
    - $\mathcal{I}$ is strictly $\prec\!\cdot$-monotonic
    - $\mathcal{I}$ is strictly $\prec\!\!\prec$-monotonic
  - Allow to infer a lower bound on the number of Pareto-optimal solution in any subset maximizing the indicator $\mathcal{I}$

Properties of Quality Indicators

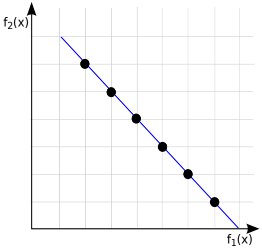
- Monotonicity
  - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)
  - A set-indicator $\mathcal{I}$ is weakly $\mathcal{R}$-monotonic if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A}\ \mathcal{R}\ \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) \geq \mathcal{I}(\mathrm{B})$.
    - $\mathcal{I}$ is weakly $\preceq$-monotonic $\Rightarrow \exists_{S \in \mathcal{A}}\ |S \cap P| = \min(k, |P|)$
  - If $\mathrm{A}\ \mathcal{R}\ \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) > \mathcal{I}(\mathrm{B})$, the indicator is strictly $\mathcal{R}$-monotonic
    - $\mathcal{I}$ is strictly $\prec$-monotonic $\Rightarrow \forall_{S \in \mathcal{A}}\ |S \cap P| = \min(k, |P|)$
    - $\mathcal{I}$ is strictly $\prec\cdot$-monotonic $\Rightarrow \forall_{S \in \mathcal{A}}\ |S \cap P| \geqslant 1$
    - $\mathcal{I}$ is strictly $\prec\!\!\prec$-monotonic
  - Allow to infer a lower bound on the number of Pareto-optimal solution in any subset maximizing the indicator $\mathcal{I}$

Properties of Quality Indicators

- Monotonicity
  - Given a set-dominance relation $\mathcal{R}$ (e.g.: weak dominance, $\preceq$)
  - A set-indicator $\mathcal{I}$ is weakly $\mathcal{R}$-monotonic if, given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, $\mathrm{A}\ \mathcal{R}\ \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) \geq \mathcal{I}(\mathrm{B})$.
    - $\mathcal{I}$ is weakly $\preceq$-monotonic $\Rightarrow \exists_{S \in \mathcal{A}}\ |S \cap P| = \min(k, |P|)$
  - If $\mathrm{A}\ \mathcal{R}\ \mathrm{B}$ implies $\mathcal{I}(\mathrm{A}) > \mathcal{I}(\mathrm{B})$, the indicator is strictly $\mathcal{R}$-monotonic
    - $\mathcal{I}$ is strictly $\prec$-monotonic $\Rightarrow \forall_{S \in \mathcal{A}}\ |S \cap P| = \min(k, |P|)$
    - $\mathcal{I}$ is strictly $\prec\cdot$-monotonic $\Rightarrow \forall_{S \in \mathcal{A}}\ |S \cap P| \geqslant 1$
    - $\mathcal{I}$ is strictly $\prec\!\!\prec$-monotonic $\Rightarrow \forall_{S \in \mathcal{A}}\ |S \cap P| \geqslant 0$
  - Allow to infer a lower bound on the number of Pareto-optimal solution in any subset maximizing the indicator $\mathcal{I}$

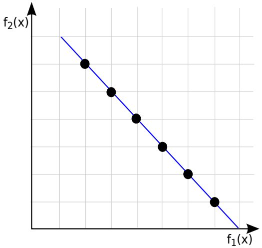
Properties of Quality Indicators

- Optimal μ-distributions
  - Characterize the indicator-optimal subsets of size up to μ



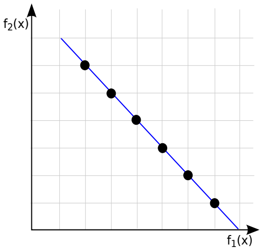
Properties of Quality Indicators

- Optimal μ-distributions
  - Characterize the indicator-optimal subsets of size up to μ
- Characteristics



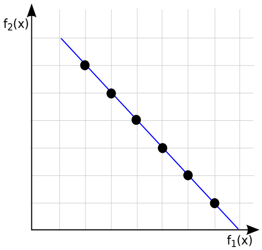
Properties of Quality Indicators

- Optimal μ-distributions
  - Characterize the indicator-optimal subsets of size up to μ
- Characteristics
  - Existence



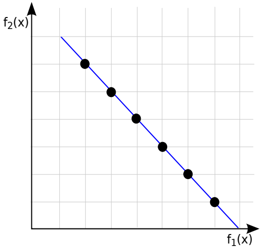
Properties of Quality Indicators

- Optimal μ-distributions
  - Characterize the indicator-optimal subsets of size up to μ
- Characteristics
  - Existence
  - Uniqueness



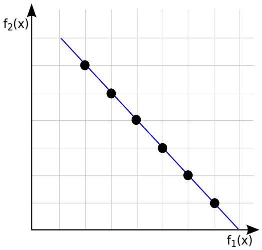
Properties of Quality Indicators

- Optimal μ-distributions
  - Characterize the indicator-optimal subsets of size up to μ
- Characteristics
  - Existence
  - Uniqueness
  - Connection to the Pareto front



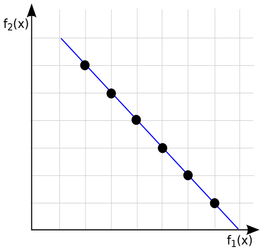
Properties of Quality Indicators

- Optimal μ-distributions
  - Characterize the indicator-optimal subsets of size up to μ
- Characteristics
  - Existence
  - Uniqueness
  - Connection to the Pareto front
  - Monotonicity with respect to μ



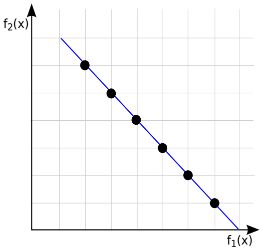
Properties of Quality Indicators

- Optimal μ-distributions
  - Characterize the indicator-optimal subsets of size up to μ
- Characteristics
  - Existence
  - Uniqueness
  - Connection to the Pareto front
  - Monotonicity with respect to μ
  - Influence of parameters



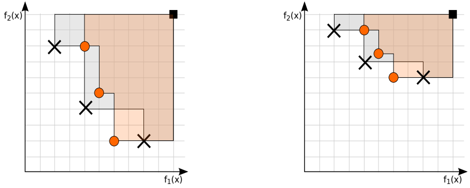
Properties of Quality Indicators

- Optimal μ-distributions
  - Characterize the indicator-optimal subsets of size up to μ
- Characteristics
  - Existence
  - Uniqueness
  - Connection to the Pareto front
  - Monotonicity with respect to μ
  - Influence of parameters
  - Distribution



Properties of the Hypervolume Indicator

- Scaling independent
  - The order defined by the hypervolume indicator is preserved under linear scaling transformations of the objective space

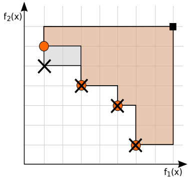


$A_1 = \{\times\}, B_1 = \{●\}$ $A_2 = \{\times\}, B_2 = \{●\}$

$$H(A_1) \geq H(B_1) \Rightarrow H(A_2) \geq H(B_2)$$

Properties of the Hypervolume Indicator

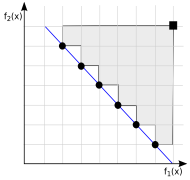
- Scaling independent
  - The order defined by the hypervolume indicator is preserved under linear scaling transformations of the objective space
- Strictly monotonic with respect to (strict) *set-dominance* ($\prec$)
  - Given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, if $\mathrm{A} \prec \mathrm{B}$ then $\mathcal{I}(\mathrm{A}) > \mathcal{I}(\mathrm{B})$
  - All solutions in an optimal subset are Pareto-optimal solutions



$$\{\times\} \prec \{\bullet\} \;\Rightarrow\; H(\{\times\}) > H(\{\bullet\})$$

Properties of the Hypervolume Indicator

- Scaling independent
  - The order defined by the hypervolume indicator is preserved under linear scaling transformations of the objective space
- Strictly monotonic with respect to (strict) *set-dominance* ($\prec$)
  - Given two point sets $\mathrm{A}, \mathrm{B} \subset \mathbb{R}^d$, if $\mathrm{A} \prec \mathrm{B}$ then $\mathcal{I}(\mathrm{A}) > \mathcal{I}(\mathrm{B})$
  - All solutions in an optimal subset are Pareto-optimal solutions
- Optimal μ-distributions
  - Ex.: The points in an optimal subset of a two-dimensional continuous linear Pareto front are evenly spaced

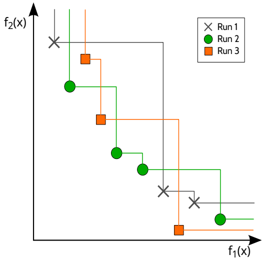


Empirical Attainment Function

- Allows the distribution of the outcomes of different runs of an optimisation algorithm to be studied in terms of location
- Example with 3 runs

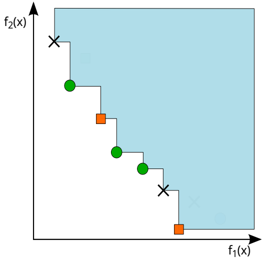
Empirical Attainment Function

- Allows the distribution of the outcomes of different runs of an optimisation algorithm to be studied in terms of location
- Example with 3 runs
    -



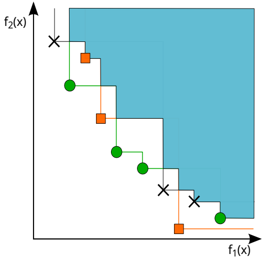
Empirical Attainment Function

- Allows the distribution of the outcomes of different runs of an optimisation algorithm to be studied in terms of location
- Example with 3 runs
  - Region of the space attained by at least $\frac{1}{3}$ of the runs



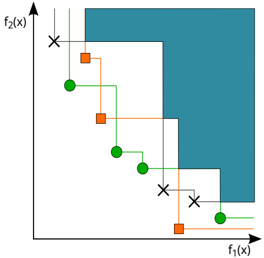
Empirical Attainment Function

- Allows the distribution of the outcomes of different runs of an optimisation algorithm to be studied in terms of location
- Example with 3 runs
  - Region of the space attained by at least $\frac{2}{3}$ of the runs



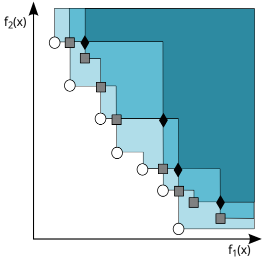
Empirical Attainment Function

- Allows the distribution of the outcomes of different runs of an optimisation algorithm to be studied in terms of location
- Example with 3 runs
    - Region of the space attained by at least $\frac{3}{3}$ of the runs



Empirical Attainment Function

- Allows the distribution of the outcomes of different runs of an optimisation algorithm to be studied in terms of location
- Example with 3 runs
  - All attainment regions



API extension to Multiobjetive Optimisation

Only affects solution evaluation:

```
objective_value(Solution) : double[0..1]
objective_value_increment(Move, Solution) : double[0..1]
```

Considers new abstract types `Value` and `Increment`:

```
objective_value(Solution) : Value[0..1]
objective_value_increment(Move, Solution) : Increment[0..1]
```

API extension to Multiobjective Optimisation

New type: `PreferenceModel`

Scalarisations:

```
scalarisation(PreferenceModel, Value): real
```

Preference relations:

```
better_or_indifferent(PreferenceModel, Value, Value): boolean
better(PreferenceModel, Value, Value): boolean
indifferent(PreferenceModel, Value, Value): boolean
incomparable(PreferenceModel, Value, Value): boolean

compare(PreferenceModel, Value, Value):
                {Better | Worse | Indifferent}
compare_partial(PreferenceModel, Value, Value):
                {Better | Worse | Indifferent | Incomparable}
```

API extension to Multiobjetive Optimisation

Other non-scalarising approaches

```
selection(PreferenceModel, Value[0..*], n=None) : int[0..*]
ranking(PreferenceModel, Value[0..*]) : int[0..*]
```

Concluding Remarks

- Solving multiobjective optimisation problem poses challenges

Concluding Remarks

- Solving multiobjective optimisation problem poses challenges
  - Size of the Pareto front

Concluding Remarks

- Solving multiobjective optimisation problem poses challenges
  - Size of the Pareto front
  - Unknown preferences

Concluding Remarks

- Solving multiobjective optimisation problem poses challenges
  - Size of the Pareto front
  - Unknown preferences
  - Preference modelling

Concluding Remarks

- Solving multiobjective optimisation problem poses challenges
  - Size of the Pareto front
  - Unknown preferences
  - Preference modelling
- It is important to understand

Concluding Remarks

- Solving multiobjective optimisation problem poses challenges
  - Size of the Pareto front
  - Unknown preferences
  - Preference modelling
- It is important to understand
  - The implication of each algorithmic approach

Concluding Remarks

- Solving multiobjective optimisation problem poses challenges
  - Size of the Pareto front
  - Unknown preferences
  - Preference modelling
- It is important to understand
  - The implication of each algorithmic approach
    - Biases towards (sets of) solutions

Concluding Remarks

- Solving multiobjective optimisation problem poses challenges
  - Size of the Pareto front
  - Unknown preferences
  - Preference modelling
- It is important to understand
  - The implication of each algorithmic approach
    - Biases towards (sets of) solutions
    - Guarantees regarding the quality of the outcome set

Concluding Remarks

- Solving multiobjective optimisation problem poses challenges
  - Size of the Pareto front
  - Unknown preferences
  - Preference modelling
- It is important to understand
  - The implication of each algorithmic approach
    - Biases towards (sets of) solutions
    - Guarantees regarding the quality of the outcome set
  - The theoretical properties of quality indicators and their implications

Concluding Remarks

- Solving multiobjective optimisation problem poses challenges
  - Size of the Pareto front
  - Unknown preferences
  - Preference modelling
- It is important to understand
  - The implication of each algorithmic approach
    - Biases towards (sets of) solutions
    - Guarantees regarding the quality of the outcome set
  - The theoretical properties of quality indicators and their implications
- Leads to better usage of optimisation algorithms and performance assessment tools

Concluding Remarks

- Solving multiobjective optimisation problem poses challenges
  - Size of the Pareto front
  - Unknown preferences
  - Preference modelling
- It is important to understand
  - The implication of each algorithmic approach
    - Biases towards (sets of) solutions
    - Guarantees regarding the quality of the outcome set
  - The theoretical properties of quality indicators and their implications
- Leads to better usage of optimisation algorithms and performance assessment tools
- The API allows to incorporate different models of preferences

Thanks!

Acknowledgments

This presentation is based upon work from COST Action Randomised Optimisation Algorithms Research Network (ROAR-NET), CA22137, supported by COST (European Cooperation in Science and Technology).

This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project: 2022.08367.CEECIND/CP1717/CT0001

EUROPEAN COOPERATION IN SCIENCE & TECHNOLOGY

Funded by the European Union

Fundação para a Ciência e a Tecnologia