ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK
Computer Engineering
and Networks Laboratory

Bio-inspired Optimization and Design

Eckart Zitzler

4. Advanced Design Issues

4.1 Multiobjective Optimization
4.2 Constraint Handling
4.3 Implementation Tools
4.4 Example Application: Network Processor Design

Introductory Example: The Knapsack Problem

weight = 750g profit = 5	weight = 1500g profit = 8	weight = 300g profit = 7	weight = 1000g profit = 3

?

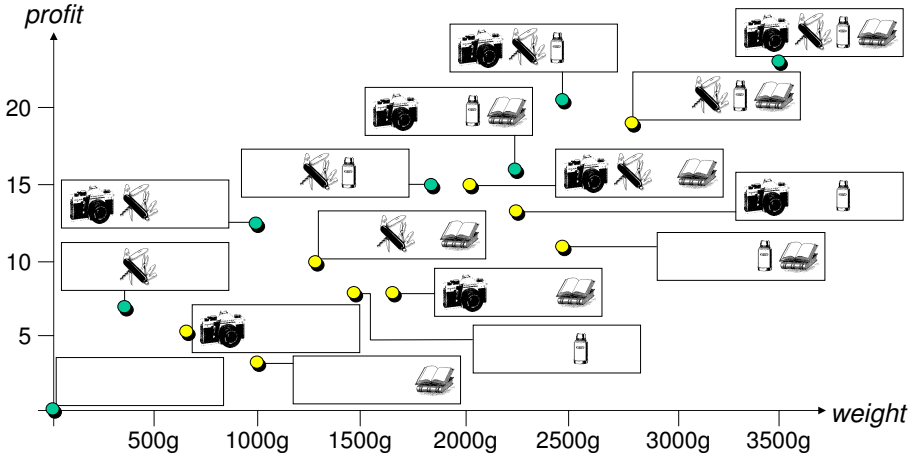
Single objective:

choose subset that
- maximizes overall profit
- w.r.t. a weight limit (constraint)

Multiobjective:

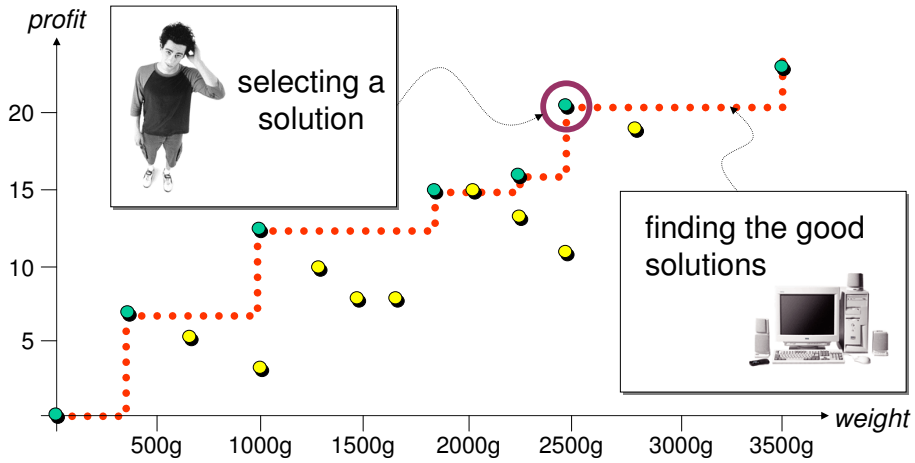
choose subset that
- maximizes overall profit
- minimizes overall weight

The Problem Landscape



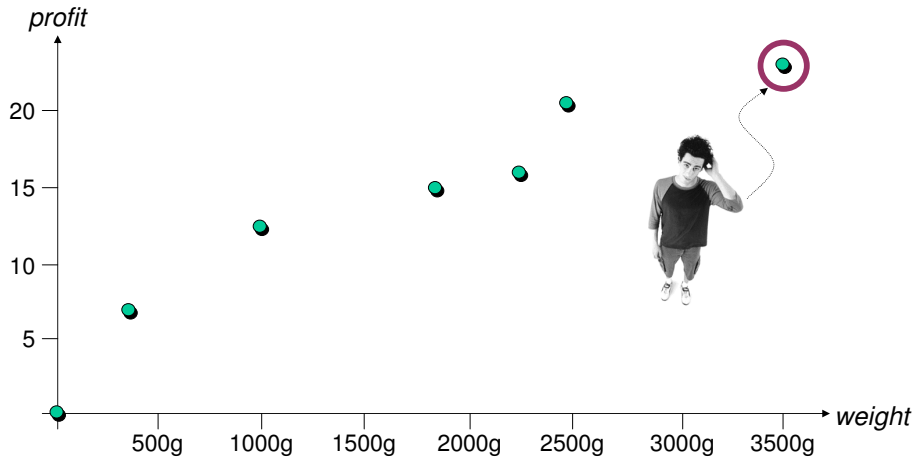
Optimization and Decision Making

Observations:
❶ there is no single optimal solution
❷ no optimum is preferrable to any other



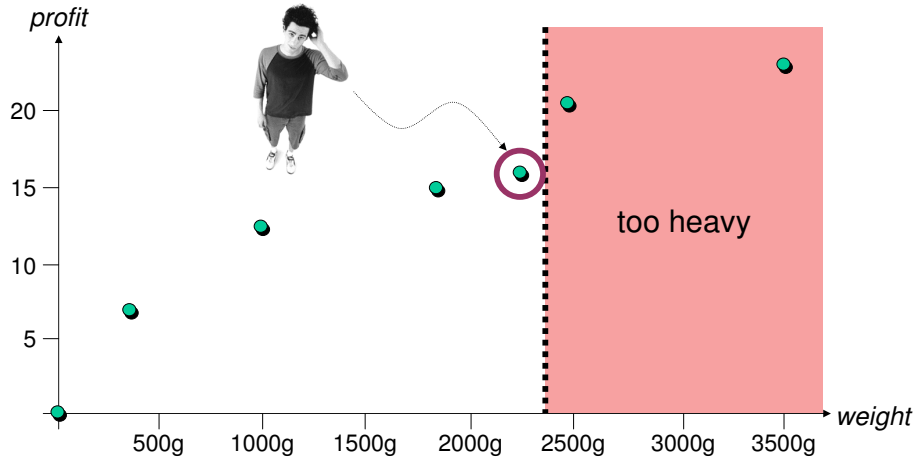
Decision Making: Selecting a Solution

Approaches: • profit more important than cost (ranking)



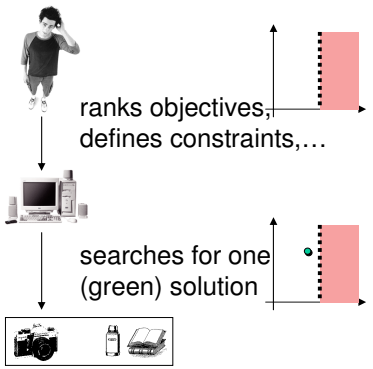
Decision Making: Selecting a Solution

Approaches: • profit more important than cost (ranking)
• weight must not exceed 2400g (constraint)



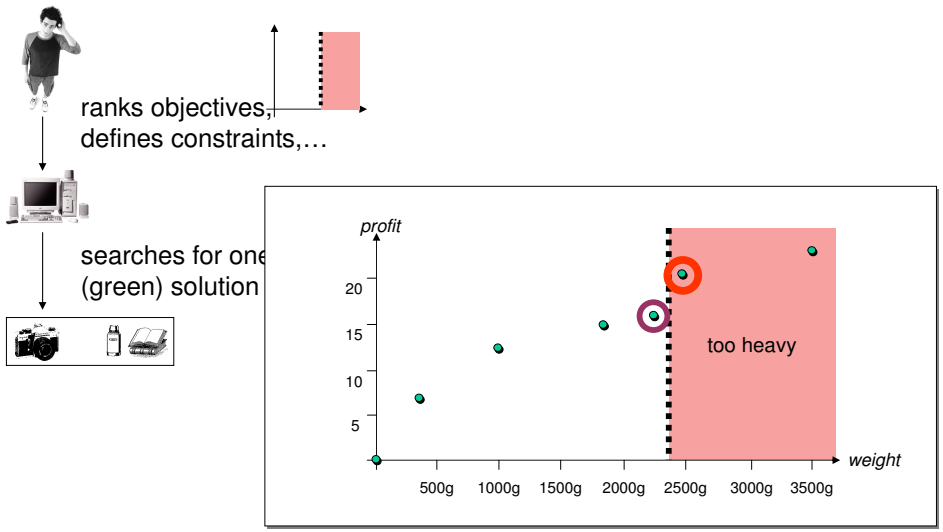
When to Make the Decision

Before Optimization:



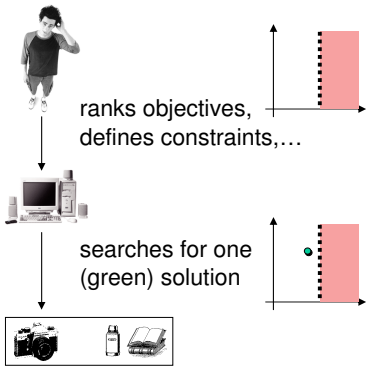
When to Make the Decision

Before Optimization:

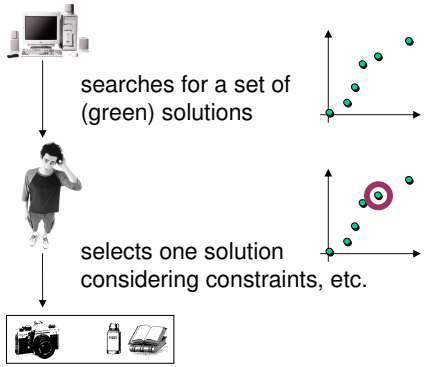


When to Make the Decision

Before Optimization:



After Optimization:



- decision making often easier
- EAs well suited

In the Following...

...you learn:

- how to deal with multiple objectives;
- how to deal with constraints;
- what type of implementation frameworks exist.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK
Computer Engineering
and Networks Laboratory

Bio-inspired Optimization and Design

Eckart Zitzler

4. Advanced Design Issues

→ 4.1 Multiobjective Optimization
4.2 Constraint Handling
4.3 Implementation Tools
4.4 Example Application: Network Processor Design

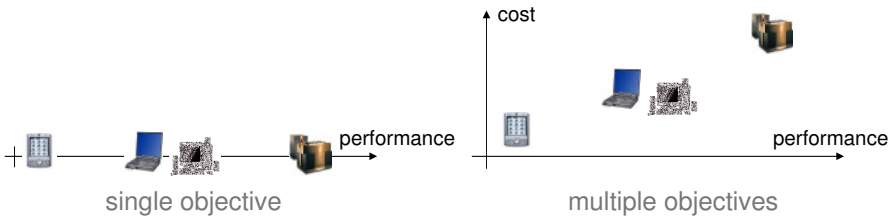
Repetition: Optimization Problem - Definition

A general optimization problem is given by a quadruple *(X, Z, f, rel)* where

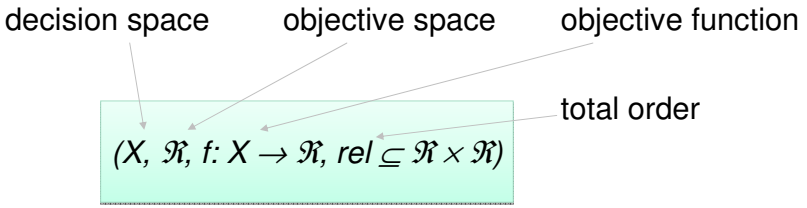
- *X* denotes the decision space containing the elements among which the best is sought; elements of *X* are called decision vectors or simply solutions;
- *Z* denotes the objective space, the space within which the decision vectors are evaluated and compared to each other; elements of *Z* are denoted as objective vectors;
- *f* represents a function $f: X \rightarrow Z$ that assigns each decision vector a corresponding objective vector; *f* is usually neither injective nor surjective;
- *rel* is a binary relation over *Z*, i.e., $rel \subseteq Z \times Z$, which represents a partial order over *Z*.

Repetition: Objective Functions

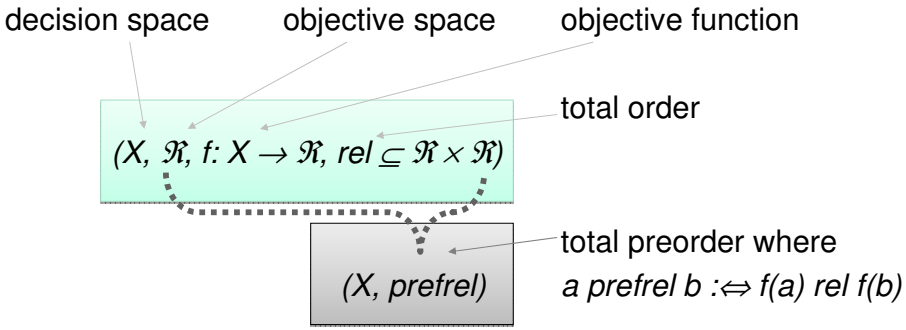
- Usually, f consists of one or several functions $f_1, ..., f_n$ that assign each solution a real number. Such a function $f_i: X \rightarrow \Re$ is called an objective function, and examples are cost, size, execution time, etc.
- In the case of a single objective function (n=1), the problem is denoted as a single-objective optimization problem; a multiobjective optimization problem involves several (n ≥ 2) objective functions:



A Single-Objective Optimization Problem

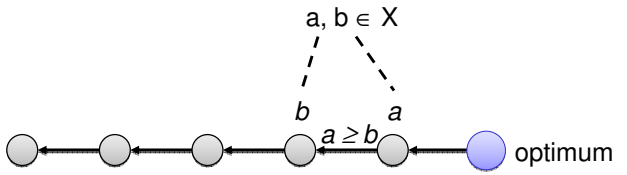


A Single-Objective Optimization Problem



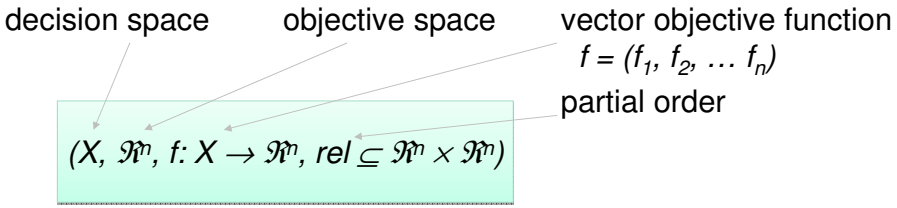
Simple Graphical Representation: Single Objective

Example: ≥ (total order)

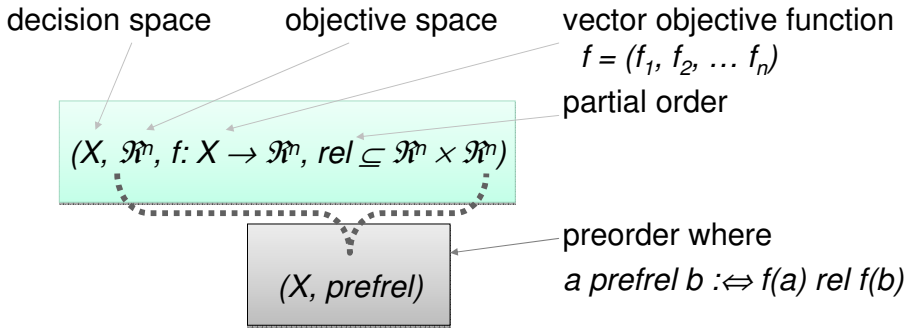


totally ordered

A Multiobjective Optimization Problem



A Multiobjective Optimization Problem



Question: What is an appropriate relation *rel*?

The Concept Of Pareto Dominance

Assumption:

- n objective functions $f_i: X \to \mathbb{R}$ where $Z = \mathbb{R}^n$
- all objectives are to be maximized

Usually considered relation: weak Pareto dominance

- optimization problem: $(X, \mathbb{R}^n, (f_1, ..., f_n), \geq)$ where $\geq$ stands for componentwise greater or equal
- weak Pareto dominance (preference structure on X):

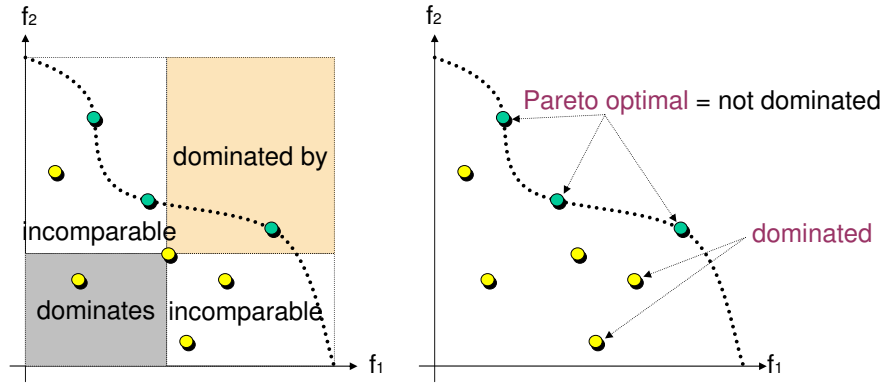
$$x_1 \succeq x_2 :\Leftrightarrow \forall 1 \leq i \leq n : f_i(x_1) \geq f_i(x_2)$$

- Pareto dominance: strict version of weak Pareto dominance

$$x_1 \succ x_2 :\Leftrightarrow x_1 \succeq x_2 \ \wedge \ x_2 \not\succeq x_1$$

Illustration of Pareto Optimality

- Solutions mapped to minimal elements of $(\mathbb{R}^n, \geq)$ are denoted as Pareto optimal.
- The entirety of all Pareto-optimal solutions is denoted as Pareto(-optimal) set.

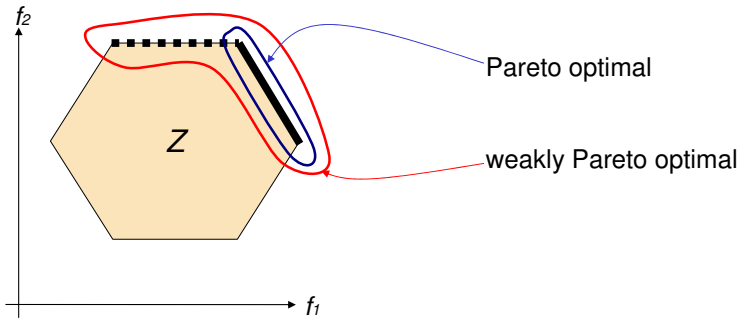


Strict Dominance and Weak Pareto Optimality

Sometimes, one considers a weaker type of optimality:

- A solution is called weakly Pareto optimal iff it is not strictly dominated by any other solution.
- A solution x_1 strictly dominates a solution x_2 if the former is better than the latter in all objectives:

$$x_1 \succ\succ x_2 :\Leftrightarrow \forall 1 \leq i \leq n : f_i(x_1) > f_i(x_2)$$



Vilfredo Pareto (1848 – 1923)

"We will say that the members of a collectivity enjoy maximum ophelimity in a certain position when it is impossible to find a way of moving from that position very slightly in such a manner that the ophelimity enjoyed by each of the individuals of that collectivity increases or decreases. That is to say, **any small displacement in departing from that position necessarily has the effect of increasing the ophelimity which certain individuals enjoy, and decreasing that which others enjoy**, of being agreeable to some and disagreeable to others."

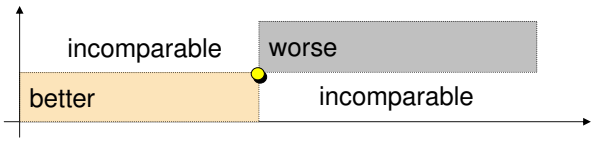
V. Pareto: Manual of Political Economy (in French), 1896

Background: Comparing Objective Vectors

The pair *(Z, rel)* forms a partially ordered set, i.e., for any two objective vectors $a, b \in Z$ there can be four situations:

- *a* and *b* are equal: *a rel b* and *b rel a*
- *a* is better than *b*: *a rel b* and not *(b rel a)*
- *a* is worse than *b*: not *(a rel b)* and *b rel a*
- *a* and *b* are incomparable: neither *a rel b* nor *b rel a*

Example: $Z = \Re^2$, (a_1, a_2) *rel* $(b_1, b_2) :\Leftrightarrow a_1 \leq b_1 \wedge a_2 \leq b_2$



Often, *(Z, rel)* is a totally ordered set, i.e., for all $a, b \in Z$ either *a rel b* or *b rel a* or both holds (no incomparable elements).

Repetition: Preference Structures

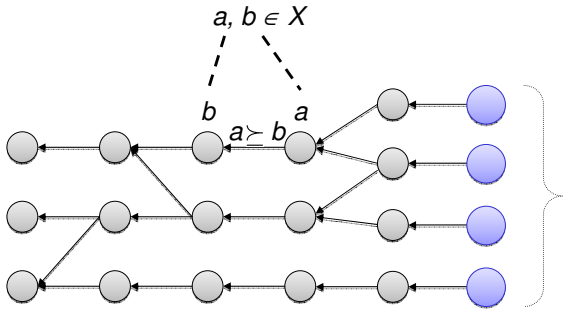
- The function *f* together with the partially ordered set *(Z, rel)* defines a preference structure on the decision space *X* that reflects which solutions the decision maker / user prefers to other solutions.
- The preference structure *prefrel* $\subseteq X \times X$ is a binary relation with

  x_1 *prefrel* $x_2 :\Leftrightarrow f(x_1)$ *rel* $f(x_2)$

  The pair *(X, prefrel)* is an preordered set, but not necessarily a partially ordered set because different solutions may be mapped to the same objective vector and antisymmetry is not fulfilled (indifferent solutions).
- One says:
  - Two solutions x_1, x_2 are equal iff $x_1 = x_2$;
  - A solution x_1 is indifferent to a solution x_2 iff x_1 *prefrel* x_2 and x_2 *prefrel* x_1 and $x_1 \neq x_2$;
  - A solution x_1 is preferred to a solution x_2 iff x_1 *prefrel* x_2;
  - A solution x_1 is strictly preferred to a solution x_2 iff x_1 *prefrel* x_2 and not (x_2 *prefrel* x_1);
  - A solution x_1 is incomparable to a solution x_2 iff neither x_1 *prefrel* x_2 nor x_2 *prefrel* x_1.

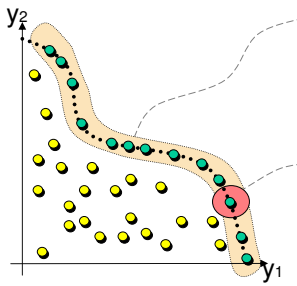
Simple Graphical Representation: Multiple Objectives

Example: $\succeq$ (preorder on the set of solutions X)



minimal elements
(Pareto optima)

Optimization and Decision Making



Pareto optimality:
defines set of optimal trade-offs
(all objectives equally important)

Decision making:
choose best compromise
(based on preference information)

❶ Decision making before search (define single objective)

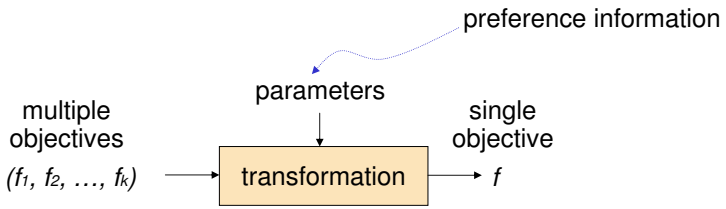
❷ Decision making after search (find/approximate Pareto set first)

❸ Decision making during search (guide search interactively)

❹ Combinations of the above

Decision Making Before Optimization: Aggregation

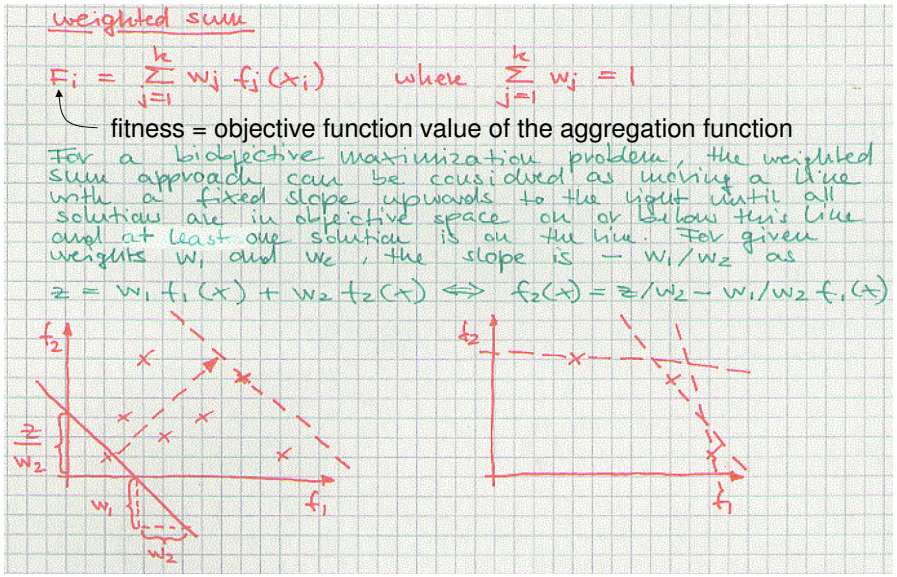
Idea: transform the multiobjective optimization problem into a single-objective optimization problem



Examples:

- ranking the objectives
- transforming *n-1* objectives into constraints
- weighted sum, Tchebycheff

Weighted-Sum Aggregation



Properties of Weighted-Sum Aggregation

1. For every weight combination, a Pareto-optimal solution is optimum:

If $F_i \geq F_j$ for one $x_i \in X$ and all $x_j \in X$, then x_i is weakly Pareto optimal; if in addition $w_\ell > 0$ for all $1 \leq \ell \leq k$, then x_i is Pareto optimal.

Proof: Assume x_i is not weakly Pareto optimal, i.e., $\exists x_j \in X$ with x_j strictly dominates x_i. Then, $f_m(x_j) > f_m(x_i)$ which implies that $F_j > F_i$. ↯

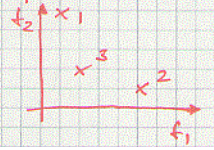
The same arguments can be used for the case $w_\ell \geq 0$. □

2. Not for every Pareto-optimal necessarily a weight-combination exists:

Assume x_i is Pareto optimal. Does a weight combination $w_1, w_2, \ldots, w_k$ exist such that $F_i \geq F_j$ for all $x_j \in X$?

No! Counterexample:

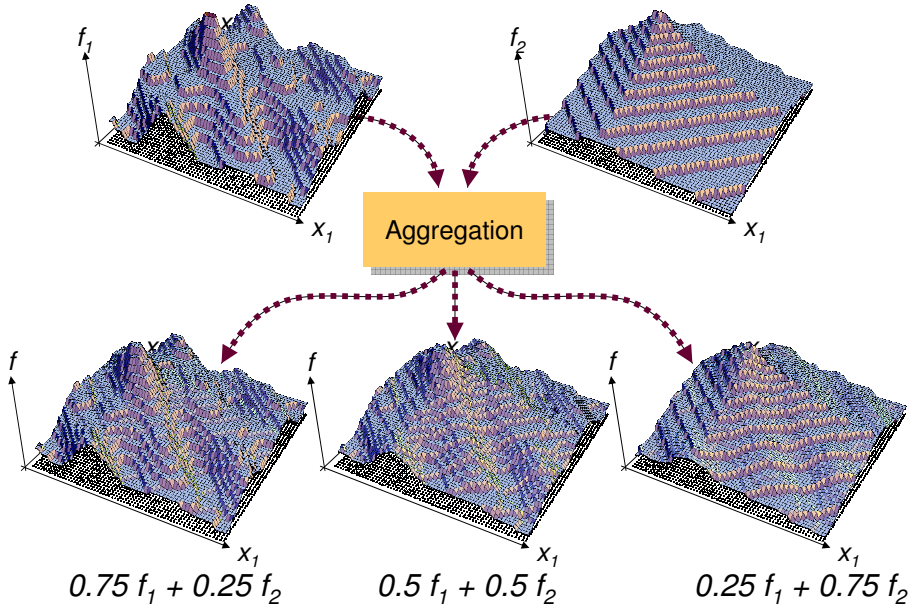
x	1	2	3
f_1	1	5	2
f_2	5	1	2



either $F_1 > F_3$ or $F_2 > F_3$

but 3 is Pareto optimal

Aggregation Example: Weighted Sum



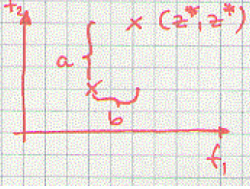
Tchebycheff Aggregation

Weighted Tchebycheff metric

Let $z^* \in \mathbb{R}$ such that $f_i(x) \leq z^*$ for $1 \leq i \leq k$, $x \in X$

$$F_i = \max_{1 \leq j \leq k} \{w_j (z^* - f_j(x_i))\} \quad \text{where} \quad \sum_{1 \leq j \leq k} w_j = 1$$

Note: F_i is to be minimized here!



$$F_i = \max\{w_1 b, w_2 a\}$$

Properties of Tchebycheff Aggregation

1. Every weight combination leads to a weakly Pareto-optimal solution:

If $F_i \leq F_j$ for one $x_i \in X$ and all $x_j \in X$, then x_i is weakly Pareto optimal, provided that $w_\ell > 0$ for all $1 \leq \ell \leq k$.

Proof: Assume x_i is not weakly Pareto optimal, i.e., $\exists x_j \in X$ with x_j strictly dominates x_i. Then, $f_m(x_j) > f_m(x_i)$ which implies $F_j < F_i$. ↯ □

2. For every Pareto-optimal solution, there is a unique weight

Assume x_i is Pareto optimal. Then, there exists a weight combination $w_1, w_2, \ldots, w_k$ such that $F_i \leq F_j$ for all $x_j \in X$.

Proof: Choose $w_1, w_2, \ldots, w_k$ such that

$$w_1(z^* - f_1(x_i)) = w_2(z^* - f_2(x_i)) = \ldots = w_k(z^* - f_k(x_i))$$

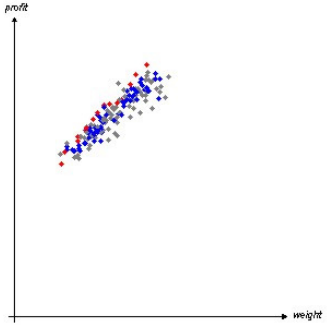
and $\sum w_\ell = 1$. Assume $F_j < F_i$ for one $x_j \in X$. Then

$$w_\ell(z^* - f_\ell(x_j)) < w_\ell(z^* - f_\ell(x_i))$$

for all $1 \leq \ell \leq k$. Accordingly, x_j strictly dominates x_i. ↯ □

Decision Making After Optimization

Idea: Identify or approximate the set Pareto-optimal solutions; the decision making is carried out on the basis of the resulting set of trade-off solutions.

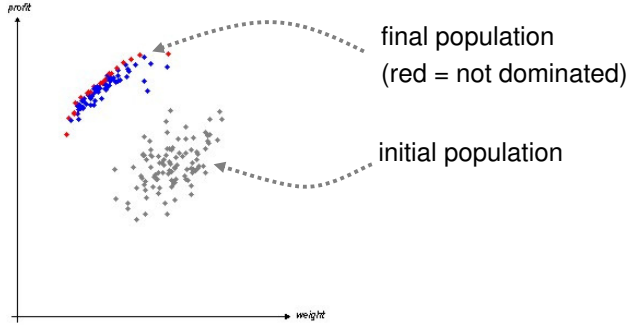


Questions:

- How does this change the underlying optimization problem?
- How can a randomized search algorithm be designed for this type of problem?

Decision Making After Optimization

Idea: Identify or approximate the set Pareto-optimal solutions; the decision making is carried out on the basis of the resulting set of trade-off solutions.

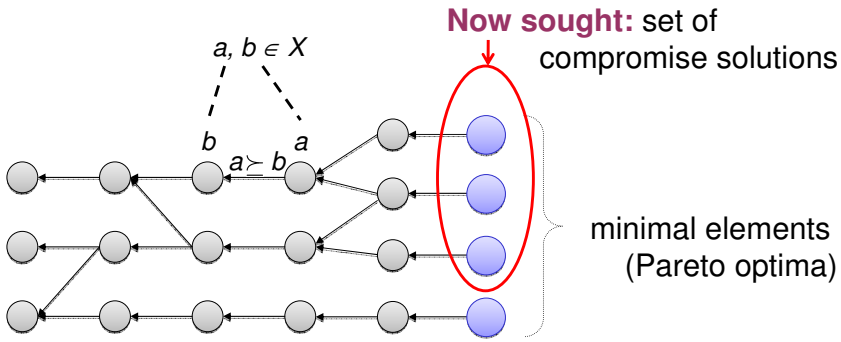


Questions:

- How does this change the underlying optimization problem?
- How can a randomized search algorithm be designed for this type of problem?

Simple Graphical Representation: Multiple Objectives

Example: $\succeq$ (preorder on the set of solutions X)



Now sought: set of compromise solutions

minimal elements (Pareto optima)

The Modified Optimization Problem

- Suppose $(X, \mathcal{F}^n, (f_1, \ldots, f_n), \succeq)$ is the original optimization problem involving n objectives. Then, the transformed optimization problem is

$(\Psi, 2^{\mathbb{R}^n}, f^*, \succeq^*)$ where

- Ψ is the set of Pareto set approximations with $\Psi := \{A \mid A \subseteq X\}$
- $2^{\mathbb{R}^n}$ is the powerset of $\mathbb{R}^n$
- $f^*(A) := S$ with $S = \{(f_1(a), f_2(a), \ldots, f_n(a)) \mid a \in A\}$
- $\succeq^*$ is a dominance relation on sets

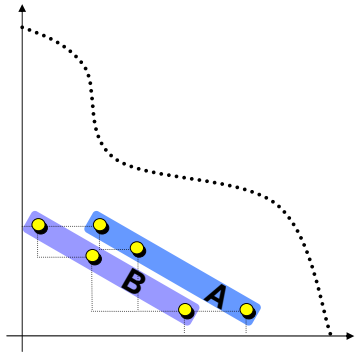
Question: How to define $\succeq^*$ based on $\succeq$?

Dominance Relations on Sets

Extension: a preference relation $\succcurlyeq$ on sets $A, B \in \Psi$

weak Pareto dominance:

$A \succcurlyeq B$ iff
for all $b \in B$ exists $a \in A$
with $a \succeq b$

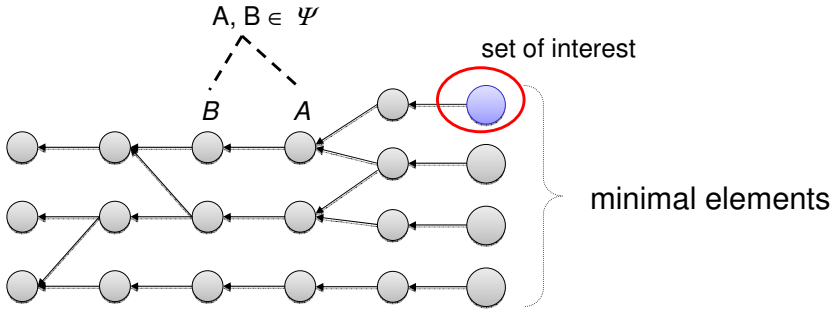


$\Rightarrow$ $(\Psi, \succcurlyeq)$ is reflexive and transitive, but usually not total

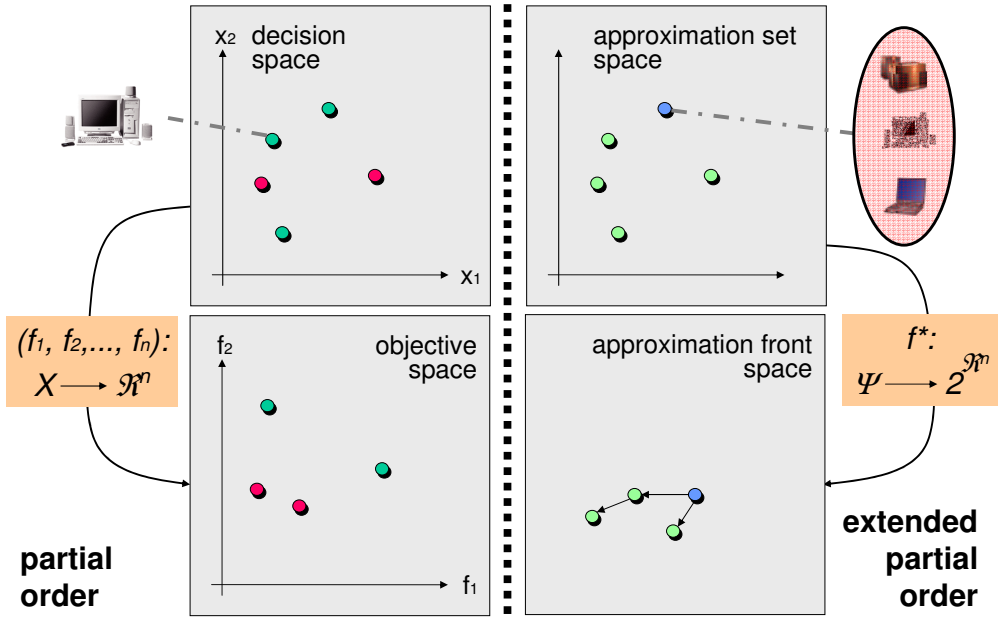
$\Rightarrow$ The mininal elements of $(\Psi, \succcurlyeq)$ represent the Pareto set

Simple Graphical Representation: Set Problem

Example: $\succcurlyeq$ (preorder on the set Ψ of sets of solutions in X)

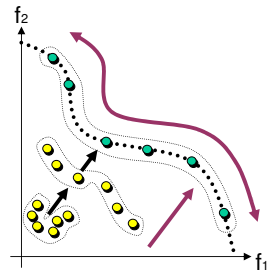


Original and Transformed Problem: Illustration



What Is the Optimization Goal?

- Find all Pareto-optimal solutions?
  - Impossible in continuous search spaces
  - How should the decision maker handle 10000 solutions?
- Find a representative subset of the Pareto set?
  - Many problems are NP-hard
  - What does representative actually mean?
- Find a good approximation of the Pareto set?
  - What is a good approximation?
  - How to formalize intuitive understanding:
    1. close to the Pareto front
    2. well distributed



Crucial: definition of $\succeq$ (different algorithms implement that differently)

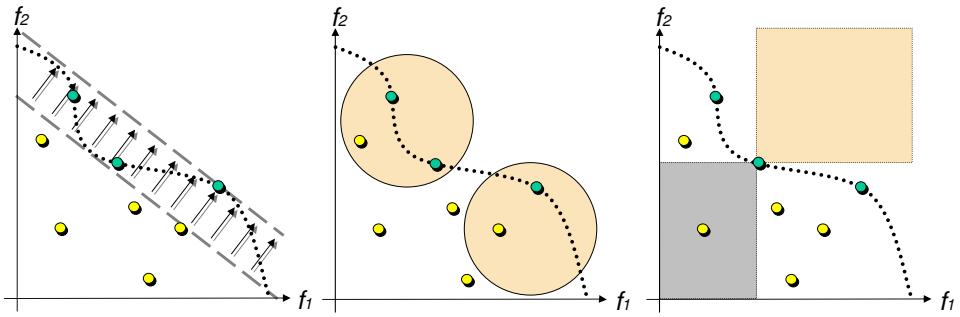
Fitness Assignment Strategies

In general, one can distinguish three classes of fitness assignment strategies for multiobjective search (Pareto set approximation):

aggregation-based *weighted sum*

criterion-based *VEGA*

set-based *SPEA2*



Aggregation-Based Approaches

In principle, there are two strategies:

1. Several separate runs, each with a different parameter combination for the aggregation function (e.g., a weight combination)

   Problem: How to select a representative set of parameters?

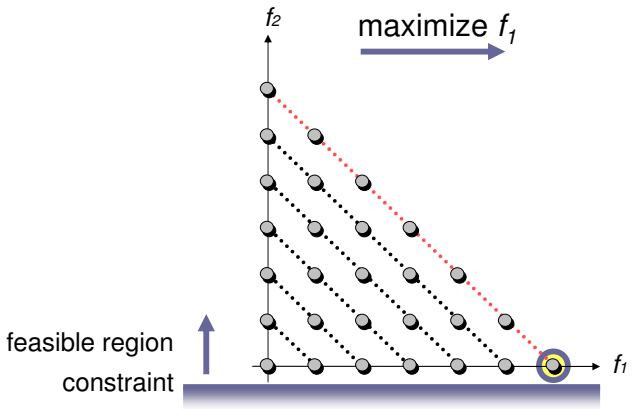
2. Single run within which the parameters are varied, e.g.,

   - by evaluating each individual according to a randomly chosen parameter combination;
   - by performing different selection stages for each given parameter combination;
   - by encoding the parameter combination in each individual separately (undergoes variation)

Example: Multistart Constraint Method

Underlying concept:

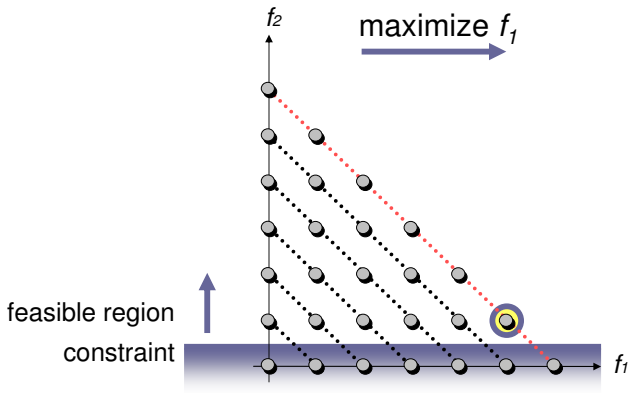
- Convert all objectives except of one into constraints
- Adaptively vary constraints



Example: Multistart Constraint Method

Underlying concept:

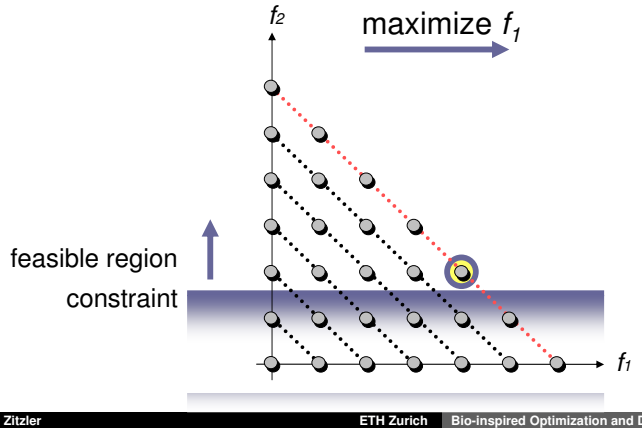
- Convert all objectives except of one into constraints
- Adaptively vary constraints



Example: Multistart Constraint Method

Underlying concept:

- Convert all objectives except of one into constraints
- Adaptively vary constraints

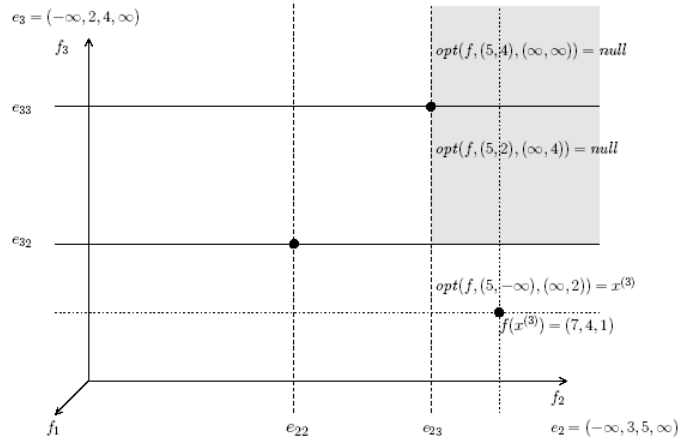


The Constraint Method For More Than Two Objectives

The previous approach based on constraint method can be extended to an arbitrary number of objectives (tricky!) If you are interested, have a look at [Laumanns et al. (2006)].

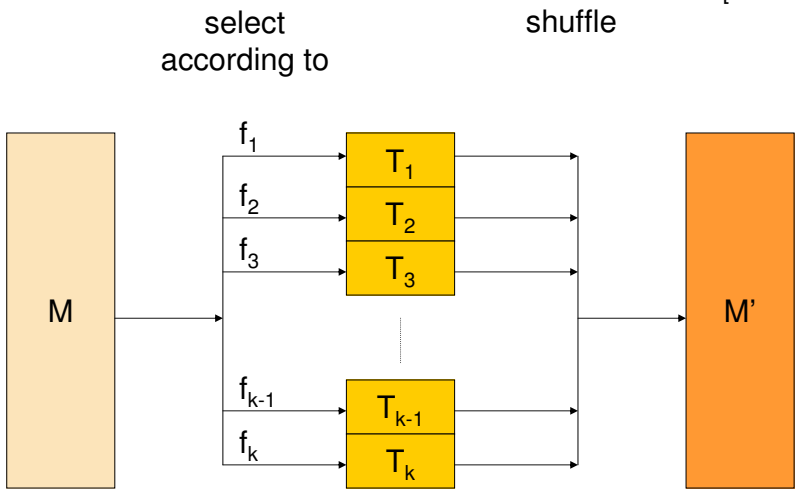
Example for *n* =3:

- f_1 is the objective to optimize
- The boxes are defined by constraints on f_2 and



Switching Between the Objectives: VEGA

[Schaffer (1985)]



VEGA In Detail

example: Vector Evaluated Genetic Algorithm (VEGA) (Schaffer 1985)

mating selection:

```
M' = ∅
for i := 1 to k
    for j := 1 to N/k
        randomly choose xe, xm in M
        if fi(xe) > fi(xm) then
            add xe to M
        else
            add xm to M
        end
    end
end
```

Dominance-Based Ranking

Types of information:

- **dominance rank** — by how many individuals is an individual dominated?
- **dominance count** — how many individuals does an individual dominate?
- **dominance depth** — at which front is an individual located?

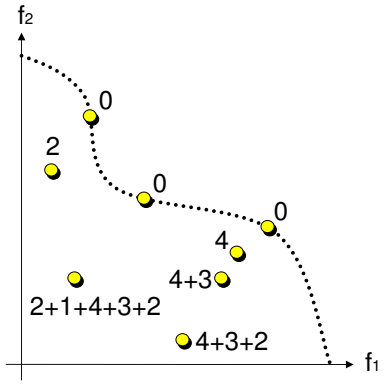
Examples:

- *MOGA, NPGA* — dominance rank
- *NSGA/NSGA-II* — dominance depth
- *SPEA/SPEA2* — dominance count + rank

SPEA2: Fitness Assignment

Basic idea: the less dominated, the fitter...

Principle: first assign each solution a weight (strength), then add up weights of dominating solutions



- S (strength) = #dominated solutions
- R (raw fitness) = ∑ strengths of dominators

Refining Rankings

ranks	pure dominance rank	refined ranking
1	0	❶ density information based on Euclidean distance
2	0	
3	0	0.245
4	1	0.311
5	1	0.329
6	2	
7	2	
8	2	0
9	3	1
		2

no selection pressure within equivalence classes

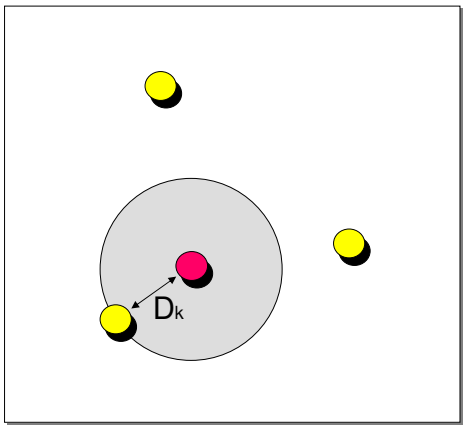
❷ modified dominance relation

SPEA2: Diversity Preservation

Density Estimation

k-th nearest neighbor method:

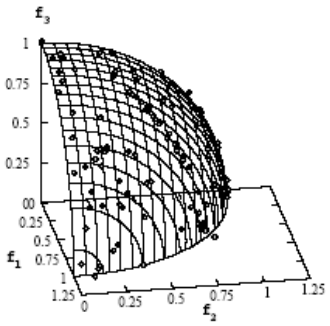
- Fitness = $R + \underbrace{1 / (2 + D_k)}_{< 1}$
- D_k = distance to the k-th nearest individual
- Usually used: k = 2



Influence of the Density Estimation Method

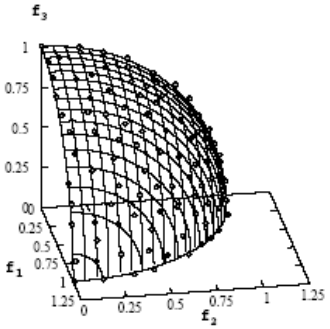
Two Nearest Neighbor Variants

Objective-Wise
NSGA-II



faster
good for 2 objectives

Euclidean Distance
SPEA2

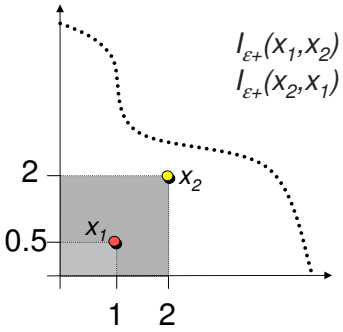


slower
good for 3 objectives and more

Refinement of Dominance Relations

A continouous dominance function (epsilon indicator):

- $I_{\varepsilon+}(x_1,x_2) = min_i\ f_i(x_1) - f_i(x_2)$
- $I_{\varepsilon+}(x_1,x_2) \geq 0$ and $I_{\varepsilon+}(x_2,x_1) < 0 \Leftrightarrow x_1$ dominates x_2



$I_{\varepsilon+}(x_1,x_2)$ gives the degree of how much x_1 dominates x_2: the larger the value, the larger the differences in the objective values

Example: IBEA

Question: How to continuous dominance functions for fitness assignment? [Zitzler, Künzli (2004)]

Given: function I (continous dominance function) with

a dominates b ⇔ I(a, b) < I(b, a)

Idea: measure for "loss in quality" if A is removed

Possible fitness function: $F_i' = \sum_{y_j \in M \setminus \{y_i\}} I(\{y_j\}, \{y_i\})$
(to be maximized)

...corresponds to continuous extension of dominance rank
...blurrs influence of dominating and dominated individuals

Example: IBEA (Cont'd)

Fitness assignment: O(n^2)

Fitness: $F_i = \sum_{y_j \in M \setminus \{y_i\}} -e^{-I(\{y_j\},\{y_i\})/\kappa}$

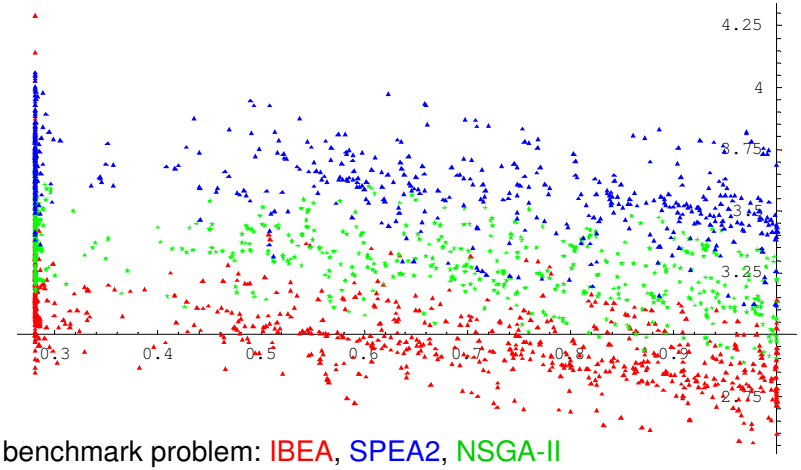
fitness is to be maximized
parameter κ is problem- and indicator-dependent
no additional diversity preservation mechanism

Mating selection: O(n)
binary tournament selection, fitness values constant

Environmental selection: O(n^2)
iteratively remove individual with lowest fitness
update fitness values of remaining individuals after each deletion

Comparison Of IBEA to Other Algorithms

30 runs plotted for three different algorithms on a biobjective minimization problem (ZDT6):



ZDT6 benchmark problem: IBEA, SPEA2, NSGA-II

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK
Computer Engineering and Networks Laboratory

Bio-inspired Optimization and Design

Eckart Zitzler

4. Advanced Design Issues

4.1 Multiobjective Optimization
→ 4.2 Constraint Handling
4.3 Implementation Tools
4.4 Example Application: Network Processor Design

What Are Constraints?

With many applications, constraints restrict the set of possible solutions and divide the search space into feasible and infeasible solutions. In this context, we are interested in finding an optimal solution among the feasible ones; however, often the problem of finding one feasible solution is already hard to solve.

constraints

= set of functions $g_1, g_2, \dots, g_e$ with $g_i : X \to \mathbb{R}$

$X_f = \{ x \in X \mid g_i(x) \geq 0 \text{ for } 1 \leq i \leq e \}$ feasible search space

goal: find $x^* \in X_f$ such that $f(x^*) \geq f(x)$ for $x \in X_f$

examples

$x_1 + x_2 = 2 \iff g_1(x_1, x_2) = 2 - x_1 - x_2$

$g_2(x_1, x_2) = x_1 + x_2 - 2$

$1 \leq x \leq 3 \iff g_1(x) = x - 1, \; g_2(x) = 3 - x$

Example: Welded Beam Design

$$
\begin{aligned}
\text{Minimize} \quad & f_w(\mathbf{x}) = 1.104\,71 h^2 \ell + 0.048\,11 t b (14.0 + \ell), \\
\text{Subject to} \quad & g_1(\mathbf{x}) \equiv 13\,600 - \tau(\mathbf{x}) \geq 0, \\
& g_2(\mathbf{x}) \equiv 30\,000 - \sigma(\mathbf{x}) \geq 0, \\
& g_3(\mathbf{x}) \equiv b - h \geq 0, \\
& g_4(\mathbf{x}) \equiv P_c(\mathbf{x}) - 6000 \geq 0, \\
& g_5(\mathbf{x}) \equiv 0.25 - \delta(\mathbf{x}) \geq 0, \\
& 0.125 \leq h \leq 10, \\
& 0.1 \leq \ell, t, b \leq 10.
\end{aligned}
\tag{4.31}
$$

The terms $\tau(\mathbf{x})$, $\sigma(\mathbf{x})$, $P_c(\mathbf{x})$ and $\delta(\mathbf{x})$ are given below:

$$\tau(\mathbf{x}) = \sqrt{(\tau'(\mathbf{x}))^2 + (\tau''(\mathbf{x}))^2 + \ell \tau'(\mathbf{x}) \tau''(\mathbf{x}) / \sqrt{0.25[\ell^2 + (h + t)^2]}},$$

$$\sigma(\mathbf{x}) = \frac{504\,000}{t^2 b},$$

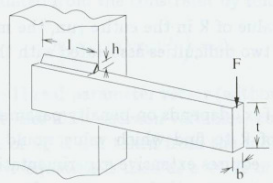


Figure 72 The welded beam design problem.

[Deb (2001)]

Constraint Handling Techniques

❶ Only feasible solution are contained in the population:

a. The initialization of the population and the variation operators are designed such that infeasible solutions cannot be generated.

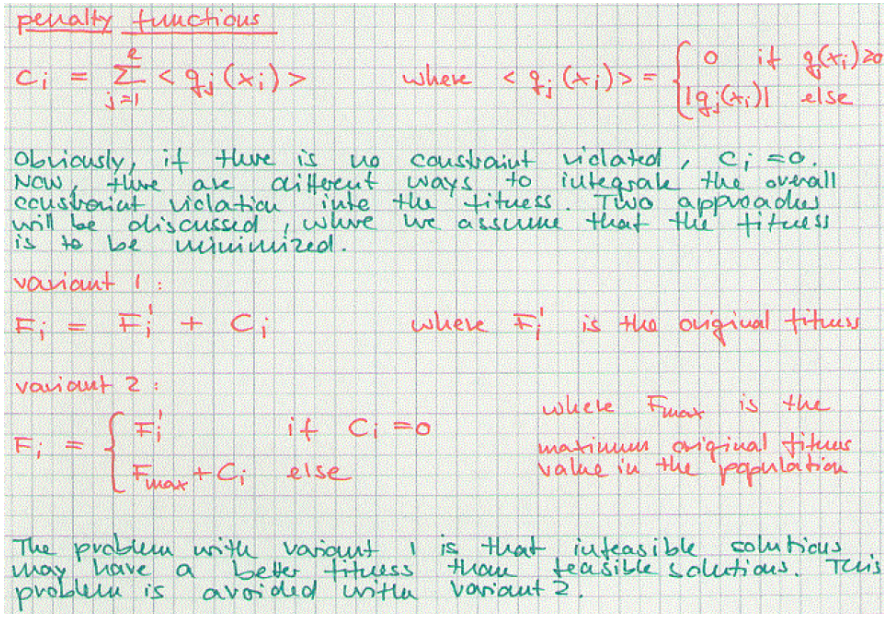
b. Infeasible solutions never enter the population: e.g., whenever an infeasible solutions is generated by mutation, the mutation operator is applied as long as a feasible child emerges.

c. The decoder function is designed such that only feasible solutions are contained in the search space.

❷ Penalize infeasible solutions:

Infeasible solutions will be penalized by diminishing their fitness.

The Penalty Approach



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK
Computer Engineering and Networks Laboratory

Bio-inspired Optimization and Design

Eckart Zitzler

4. Advanced Design Issues

4.1 Multiobjective Optimization
4.2 Constraint Handling
→ 4.3 Implementation Tools
4.4 Example Application: Network Processor Design

Why Are Implementations Tools Useful?

Application engineer

- knowledge in the algorithm domain necessary
- state-of-the-art algorithms get more and more complex
- many algorithms

Algorithm designer

- comparison to competing algorithms mandatory
- tests on various benchmark problems necessary
- algorithms and applications become increasingly complex

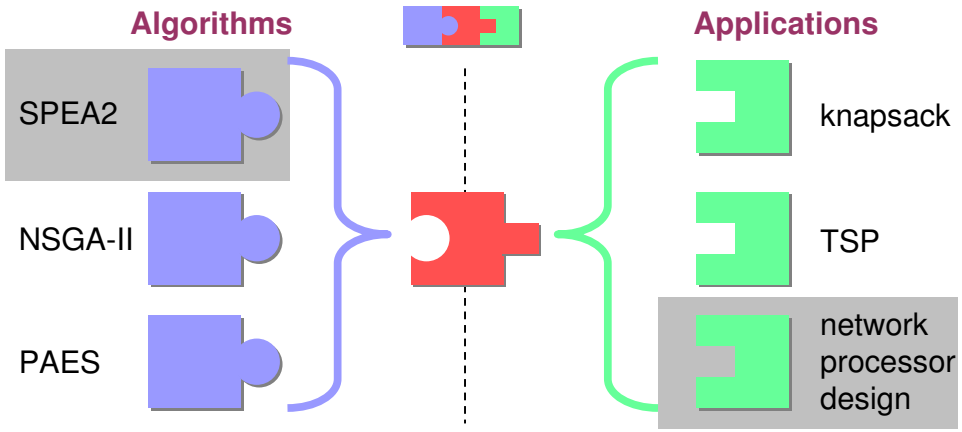
high implementation effort / risk of implementation errors

Programming libraries:

⊕ valuable tools to tailor a particular technique to a specific application

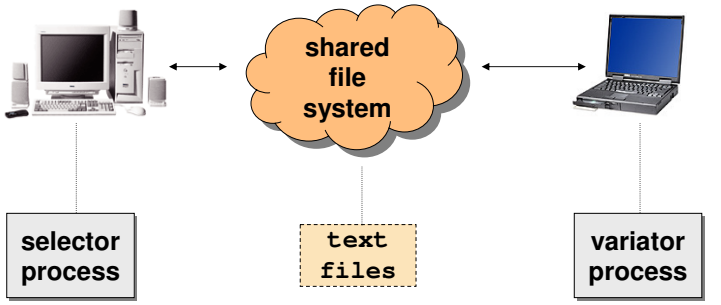
⊖ exchange of optimization algorithm or application still difficult

The Concept of PISA



text-based
Platform and programming language independent **I**nterface
for **S**earch **A**lgorithms [Bleuler et al. (2002)]

PISA: Implementation

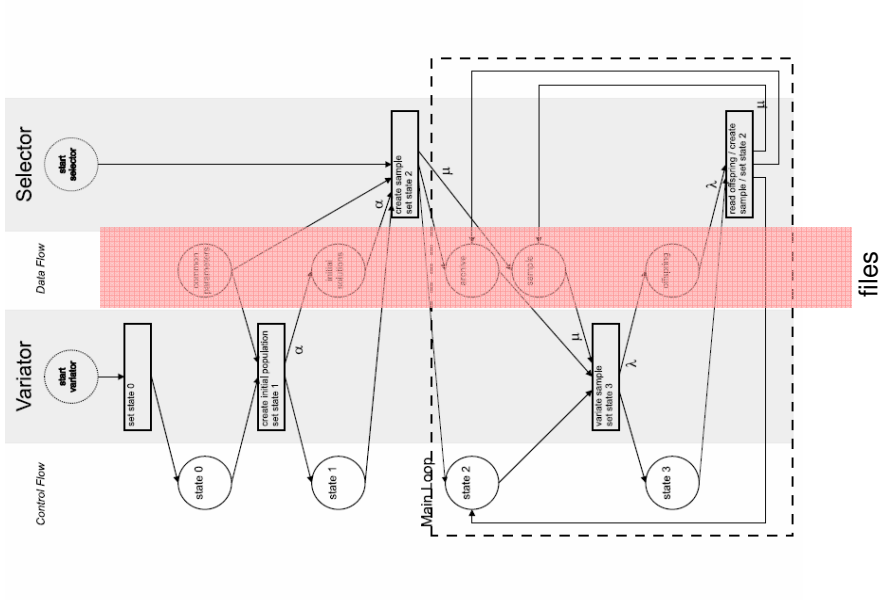


application independent:
- mating / environmental selection
- individuals are described by IDs and objective vectors

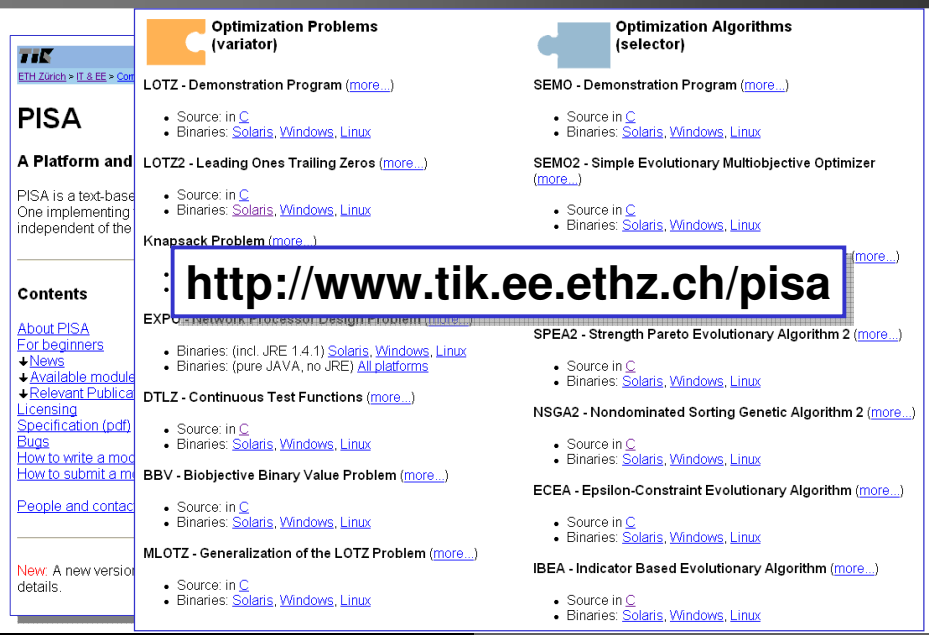
handshake protocol:
- state / action
- individual IDs
- objective vectors
- parameters

application dependent:
- variation operators
- stores and manages individuals

The Handshake Protocol (Petri Net)



Available PISA Modules



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK
Computer Engineering
and Networks Laboratory

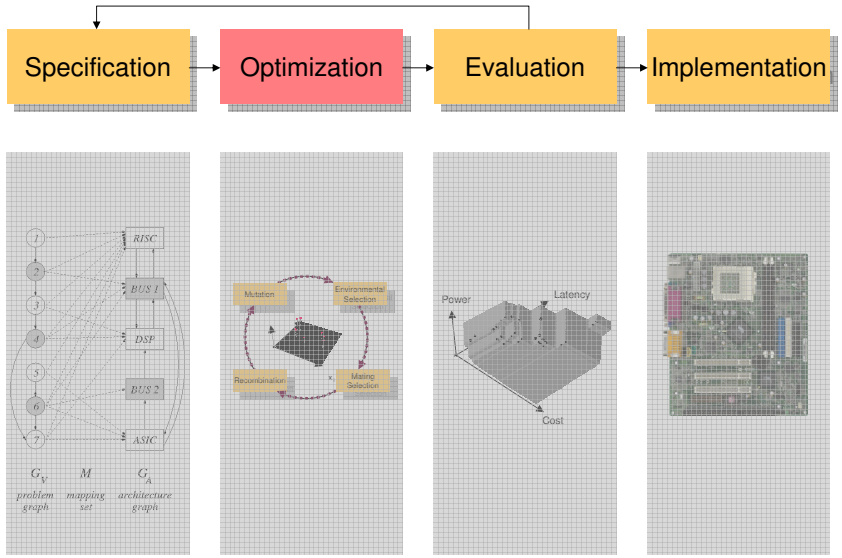
Bio-inspired Optimization and Design

Eckart Zitzler

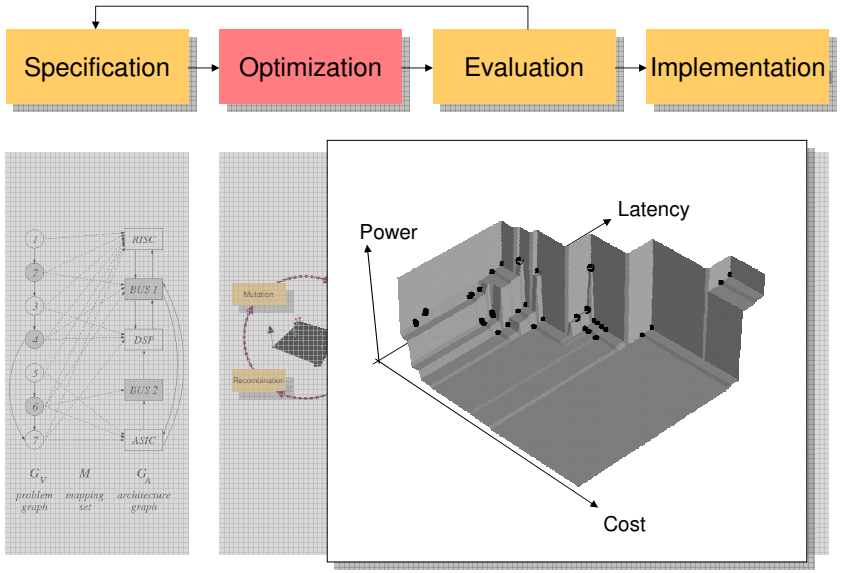
4. Advanced Design Issues

4.1 Multiobjective Optimization
4.2 Constraint Handling
4.3 Implementation Tools
→ 4.4 Example Application: Network Processor Design
(application details are not part of the exam)

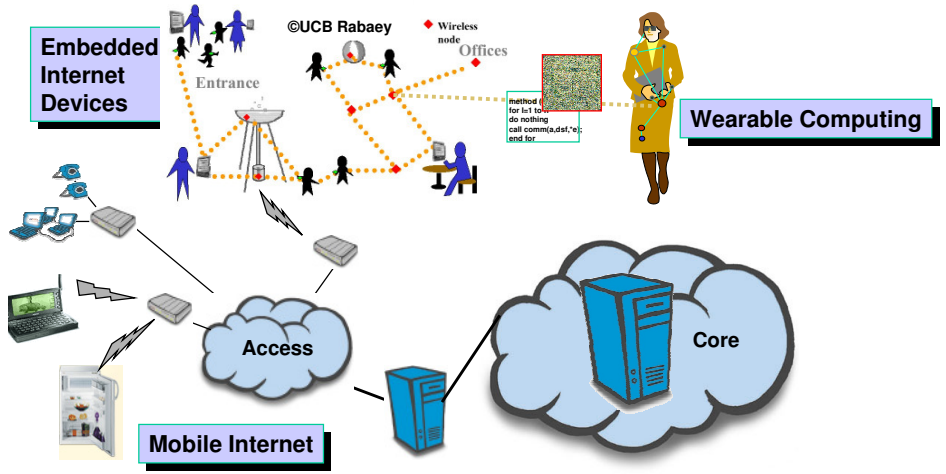
Design Space Exploration: Setting



Design Space Exploration: Setting



Packet Processing in Networks

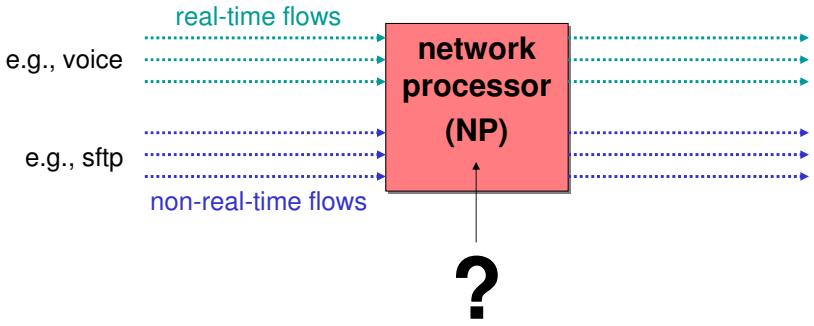


The following presentation describes a network processor application that has been carried out at the Computer Engineering Group at ETH Zurich.

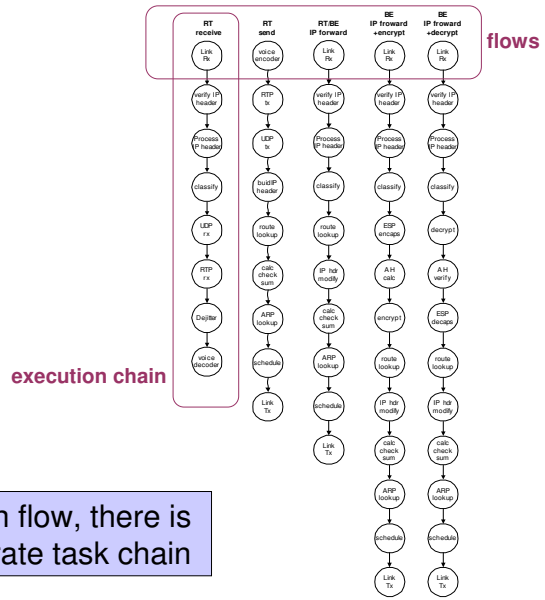
Network Processors: Overview

Network processor = high-performance, programmable device designed to efficiently execute communication workloads

incoming flows (packet streams) | routing / forwarding transcoding encryption / decryption | outgoing flows (processed packets)



Task Model: A Simple Network Processor

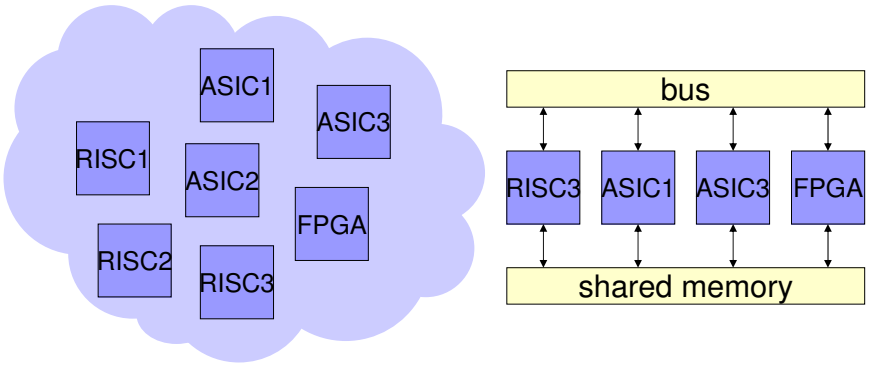


for each flow, there is a separate task chain

Resource Model: Architecture

In the following, we assume a simple architecture model (in general the model allows for more complex architecture):

available resources | selected resources



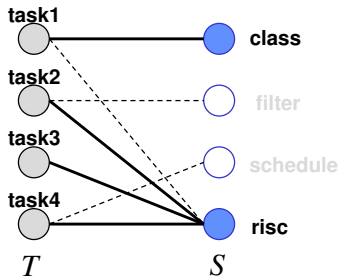
Allocation and Binding

- Allocation can be represented as a function:

$$allocation : S \rightarrow \{true, false\}$$

- Binding is a function:

$$binding : (T \times S) \rightarrow \{true, false\}$$



- Binding restrictions:

$$\forall t \in T \quad \exists s \in S : \; binding(t,s) = true$$
$$\forall s \in S : \; binding(t,s) = true \;\Rightarrow\; allocation(s) = true$$

- **Note:** for each usage scenario, there may be another binding!

Variator Process: Representation

- **Allocation:** integer vector (1 per individual)

1	0	1	0	2

→ for each resource type, number of allocated instances

- **Binding:** integer vector (1 per usage scenario)

5	5	2	4	1	3

→ for each task and for each scenario, selected resource instance

- **Scheduling priorities:** permutation vector (1 per usage scenario)

1	3	2

→ for each flow, its priority

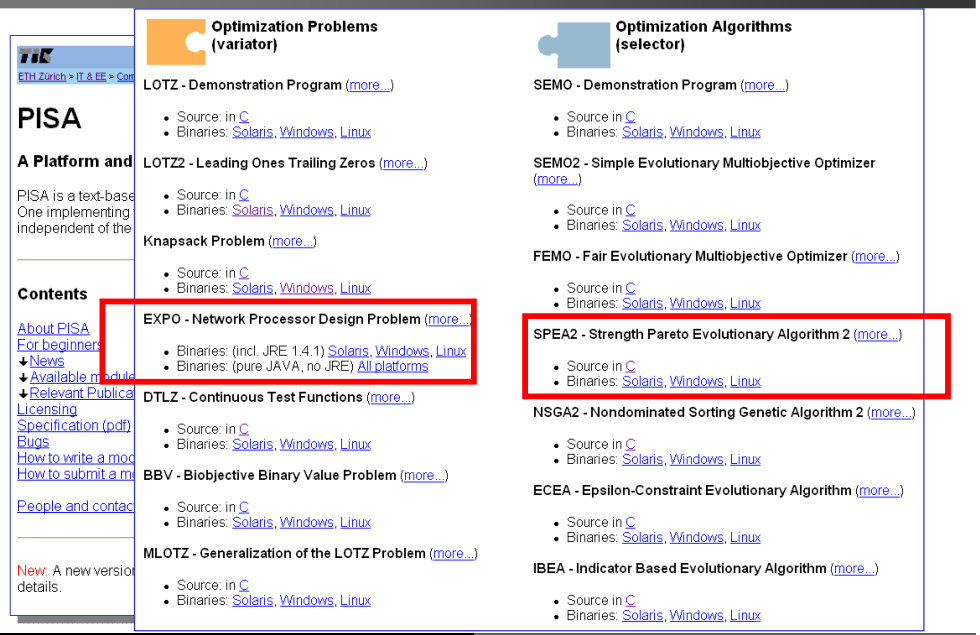
Variator Process: Mutation & Recombination

❶ **Allocation:** integer vector
- mutation: randomly choose an integer between *0* and *max_instances* per resource type
- recombination: one-point crossover
- appropriate decoder function ensures feasible allocations

❷ **Binding:** integer vector
- mutation: randomly choose admissible resource instance
- recombination: one-point crossover
- appropriate decoder function ensures feasible bindings

❸ **Priorities:** permutation vector
- mutation: swap operator
- recombination: scramble-sublist crossover

Demo



References

- S. Bleuler, M. Laumanns, L. Thiele, E. Zitzler (2003): PISA - A Platform and Programming Language Independent Interface for Search Algorithms. EMO 2003 Proceedings, Springer, Berlin, pp. 494 - 508.
- K. Deb (2001): Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Chicherster, UK.
- M. Laumanns, L. Thiele, E. Zitzler (2006): An Efficient Metaheuristic Based on the Epsilon-Constraint Method. European Journal of Operational Research, Volume 169, Issue 3 , pp 932-942. (**http://dx.doi.org/10.1016/j.ejor.2004.08.029**)
- D. Schaffer (1985): Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp. 93-100.
- E. Zitzler, S. Künzli (2004): Indicator-Based Selection in Multiobjective Search. PPSN VIII Proceedings, Springer, Berlin, pp. 832-842.