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

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

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

The Problem Landscape

Optimization and Decision Making

finding the good solutions

selecting a solution
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:**
- ranks objectives, defines constraints,....
- searches for one (green) solution

**After Optimization:**
- too heavy
- weight 3500g
- profit 20
- weight 2500g
- profit 15
- weight 2000g
- profit 10
- weight 1500g
- profit 5
- weight 1000g
- profit 0
4. Advanced Design Issues

4.1 Multiobjective Optimization
4.2 Constraint Handling
4.3 Implementation Tools
4.4 Example Application: Network Processor Design
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 $f_i: X \rightarrow \mathbb{R}$ 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 ($n \geq 2$) objective functions:

![Diagram showing single objective vs. multiple objectives](image)

A Single-Objective Optimization Problem

$$\langle X, \mathbb{R}, f: X \rightarrow \mathbb{R}, \text{rel} \subseteq \mathbb{R} \times \mathbb{R} \rangle$$

Simple Graphical Representation: Single Objective

Example: $\geq$ (total order)

![Graph showing total order](image)
A Multiobjective Optimization Problem

decision space \( X \)

objective space \( \mathbb{R}^n \)

vector objective function \( f = (f_1, f_2, \ldots, f_n) \)

partial order \( \text{rel} \subseteq \mathbb{R}^n \times \mathbb{R}^n \)

\((X, \mathbb{R}^n, f: X \rightarrow \mathbb{R}^n, \text{rel} \subseteq \mathbb{R}^n \times \mathbb{R}^n)\)

The Concept Of Pareto Dominance

Assumption:

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

Usually considered relation: weak Pareto dominance

- optimization problem: \((X, \mathbb{R}^n, (f_1, \ldots, f_n), \geq)\) where \( \geq \) stands for componentwise greater or equal
- weak Pareto dominance (preference structure on \( X \)):
  \[ x_1 \geq x_2 \iff \forall 1 \leq i \leq n : f_i(x_1) \geq f_i(x_2) \]
- Pareto dominance: strict version of weak Pareto dominance
  \[ x_1 > x_2 \iff x_1 \geq x_2 \land x_2 \neq x_1 \]

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:
  $$ x_1 \triangleright x_2 :\iff \forall 1 \leq i \leq n : f_i(x_1) > f_i(x_2) $$

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:** $Z = \mathbb{R}^2$, $(a_1, a_2) \ rel \ (b_1, b_2) :\iff a_1 \leq b_1 \land a_2 \leq b_2$

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 $\text{prefrel} \subseteq X \times X$ is a binary relation with
  $$ x_1, \text{prefrel} x_2 :\iff f(x_1) \ rel f(x_2) $$

The pair $(X, \text{prefrel})$ 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 $x_1 \ \text{prefrel} \ x_2$ and $x_2 \ \text{prefrel} \ x_1$ and $x_1 \neq x_2$;
  - A solution $x_1$ is **preferred** to a solution $x_2$ iff $x_1 \ \text{prefrel} \ x_2$;
  - A solution $x_1$ is **strictly preferred** to a solution $x_2$ iff $x_1 \ \text{prefrel} \ x_2$ and not ($x_2 \ \text{prefrel} \ x_1$);
  - A solution $x_1$ is **incomparable** to a solution $x_2$ iff neither $x_1 \ \text{prefrel} \ x_2$ nor $x_2 \ \text{prefrel} \ x_1$.

Vilfredo Pareto (1848 – 1923)

“We will say that the members of a collectivity enjoy maximum ophehmity 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 ophehmity 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 ophehmity which certain individuals enjoy, and decreasing that which others enjoy, of being agreeable to some and disagreeable to others.”

**Simple Graphical Representation: Multiple Objectives**

**Example:** \( \preceq \) (preorder on the set of solutions \( X \))

\[ a, b \in X \]

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)

1. Decision making before search (define single objective)
2. Decision making after search (find/approximate Pareto set first)
3. Decision making during search (guide search interactively)
4. Combinations of the above

**Decision Making Before Optimization: Aggregation**

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

\[ \text{fitness} = \text{objective function value of the aggregation function} \]

**Examples:**
- ranking the objectives
- transforming \( n-1 \) objectives into constraints
- weighted sum, Tchebycheff

**Weighted-Sum Aggregation**

\[ f_i = \frac{1}{k} w_i f_i(x_i) \text{ where } \sum_{i=1}^{k} w_i = 1 \]

Fitness = objective function value of the aggregation function

For a k-objective maximization problem, the weighted sum approach can be considered as running a line with a fixed slope upwards (i.e., increasing the objective function) on the left until all solutions are in objective space on or below this line. Once the line touches a new solution, \( x_i \), the slope is \( -w_i/w_j \) as

\[ f_i(x_i) + w_j f_j(x_j) \Rightarrow k_i(x) = \frac{w_i}{w_i + w_j} f_i(x) \text{ and } \frac{w_j}{w_i + w_j} f_j(x) \]
### Properties of Weighted-Sum Aggregation

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

   - If \( f_i > f_j \) for all \( i \neq j \) and all \( x \in X \), then \( x \) is weakly Pareto optimal if, for all \( i \leq k \leq e \), then \( x_i \) is Pareto optimal.

   - Proof: Assume \( x_1 \) is not weakly Pareto optimal, i.e., \( f_i(x_1) > f_j(x_0) \) for some \( i \neq j \). Then \( f_i(x_1) > f_j(x_0) \) and \( f_k(x_1) > f_k(x_0) \) for some \( k \). Thus, \( x_1 \) is not weakly Pareto optimal.

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

   - Assume \( x_1 \) is Pareto optimal. Does a weight combination \( w_j \) exist such that \( f_i(x_1) \leq f_j(x_1) \) for all \( i \neq j \)?

   - Note: \( x_1 \) is to be minimized here!

### Aggregation Example: Weighted Sum

- Aggregation:

  - \( 0.75 f_1 + 0.25 f_2 \)
  - \( 0.5 f_1 + 0.5 f_2 \)
  - \( 0.25 f_1 + 0.75 f_2 \)

### Tchebycheff Aggregation

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

   - \( f_i(x) \leq \sum \omega_j f_j(x) \) for all \( i \neq j \) and all \( x \in X \), then \( x \) is weakly Pareto optimal, provided that \( \omega_j > 0 \) for all \( j \).

   - Proof: Assume \( x_1 \) is not weakly Pareto optimal, i.e., \( f_i(x_1) < f_j(x_2) \) for some \( i \) and \( x_2 \). Then \( f_i(x_1) < f_j(x_2) \) implies \( f_i(x_1) < f_j(x_2) \) for all \( i \).

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

   - Assume \( x_1 \) is Pareto optimal. Then, there exists \( \omega_j \) such that \( f_j(x_1) \leq \omega_j f_j(x) \) for all \( x \in X \).

   - Proof: Choose \( \omega_1, \omega_2, \ldots, \omega_k \) such that
     - \( \omega_j f_j(x_1) = \omega_j f_j(x_1) \)
     - \( \omega_j f_j(x_1) = \omega_j f_j(x_1) \)

     for all \( i \leq k \). Therefore, \( f_j(x_1) \) strictly dominates \( x_j \). Accordingly, \( x_j \) strictly dominates \( x_j \).
**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

---

**The Modified Optimization Problem**

- Suppose \( (X, \mathcal{P}, (f_1, \ldots, f_n), \succeq) \) is the original optimization problem involving \( n \) objectives. Then, the transformed optimization problem is \( (\Psi, 2^{\mathbb{R}^n}, f^*, \succeq^*) \) where
  - \( \Psi \) 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 $\succeq$ on sets $A, B \in \Psi$

weak Pareto dominance:

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

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

$\Rightarrow$ The minimal elements of $(\Psi, \succeq)$ represent the Pareto set

**Simple Graphical Representation: Set Problem**

**Example:** $\succeq$ (preorder on the set $\Psi$ 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:
    - close to the Pareto front
    - 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 one into constraints
- Adaptively vary constraints

feasible region
constraint

maximize $f_1$

$\maximize f_1$

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 $f_3$

Switching Between the Objectives: VEGA

select according to

[Schaffer (1985)]

VEGA In Detail

population $k$ separate selections mating pool

$M' = \emptyset$

for $i = 1$ to $k$

randomly choose $x_i$ from $M$

if $f_i(x_i) < f_i(x)$ then

add $x_i$ to $M$

end

end

example: Vector Evaluated Genetic Algorithm (VEGA)

waiting selection:

for $j = 1$ to $N/k$

randomly choose $x_j$ from $M$

if $f_j(x_j) < f_j(x_m)$ then

add $x_j$ to $M$

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

![Graph showing fitness assignment with dominance ranks and counts](image)

**S (strength):** 
- #dominated solutions

**R (raw fitness):**
\[ R = \sum \text{strengths of dominators} \]

### Refining Rankings

<table>
<thead>
<tr>
<th>ranks</th>
<th>pure dominance rank</th>
<th>refined ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.245</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.311</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.329</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

- No selection pressure within equivalence classes
- Density information based on Euclidean distance

### SPEA2: Diversity Preservation

**Density Estimation**

**k-th nearest neighbor method:**
- Fitness = \( R + 1 / (2 + D_k) \)
  - \( D_k \) = distance to the k-th nearest individual
  - Usually used: \( k = 2 \)

![Graph showing diversity preservation](image)
Influence of the Density Estimation Method

Two Nearest Neighbor Variants

Objective-Wise
NSGA-II

Euclidean Distance
SPEA2

faster
fast good for 2 objectives

good for 3 objectives and more

slower

Refinement of Dominance Relations

A continuous dominance function (epsilon indicator):

- $I_{\epsilon}(x_i, x_j) = \min_i f_i(x_i) - f_i(x_j)$
- $I_{\epsilon}(x_i, x_j) \geq 0$ and $I_{\epsilon}(x_j, x_i) < 0 \iff x_i$ dominates $x_j$

$\epsilon$ gives the degree of how much $x_i$ dominates $x_j$; 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 $I$ (continuous dominance function) with

- $a$ dominates $b \iff I(a, b) < I(b, a)$

**Idea:** measure for “loss in quality” if $A$ is removed

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

(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 $\kappa$ 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

---

**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 a feasible solution is already hard to solve.

- Constraints = set of functions $g_1, g_2, \ldots, g_e$ with $g_i : \mathbb{R}^n \rightarrow \{0,1\}$
- Feasible search space $x_f = \{x \in \mathbb{R}^n | g_i(x) \geq 0 \text{ for } 1 \leq i \leq e\}$
- Feasible solutions $\text{Goal: } \text{find } x^* \in x_f$ such that $f(x^*) = f(x)$ for $x \in x_f$

**Example: Welded Beam Design**

Minimize $f_w(x) = 1.1047l^2t + 0.04811ht + 14.01t$, Subject to $g_1(x) = 13.660 - \gamma(x) \geq 0$, $g_2(x) = 30.000 - \alpha(x) \geq 0$, $g_3(x) = b - h \geq 0$, $g_4(x) = P_c(x) - 6000 \geq 0$, $g_5(x) = 0.25 - \delta(x) \geq 0$, $0.125 \leq h \leq 10$, $0.1 \leq c, b \leq 10$.

The terms $\gamma(x), \alpha(x), P_c(x)$ and $\delta(x)$ are given by:

- $\gamma(x) = \frac{1}{12/t^3}$
- $\alpha(x) = \frac{5040000}{t^3b}$

Figure 72: The welded beam design problem.

[Deb (2001)]
Constraint Handling Techniques

1. Only feasible solutions 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 solution 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.

2. Penalize infeasible solutions:
   Infeasible solutions will be penalized by diminishing their fitness.

The Penalty Approach

\[ C_i = \frac{z_i}{\sum_{j=1}^{n} q_j(x_i)} \text{ where } \begin{cases} 0 & \text{if } q_j(x_i) > 0 \\ 1 & \text{otherwise} \end{cases} \]

Obviously, if there is no constraint violated, \( C_i = 0 \).
Now, there are different ways to integrate the overall constraint violation into the fitness. Two approaches will be discussed, where we assume that the fitness is to be minimized.

Variant 1:
\[ F_i = F'_i + C_i \quad \text{where } F'_i \text{ is the original fitness} \]

Variant 2:
\[ F_i = \begin{cases} F'_i & \text{if } C_i = 0 \\ F_{\text{max}} + C_i & \text{else} \end{cases} \quad \text{where } F_{\text{max}} \text{ is the maximum original fitness value in the population} \]

The problem with variant 1 is that infeasible solutions may have a better fitness than feasible solutions. This problem is avoided with variant 2.

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

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

Algorithms

- SPEA2
- NSGA-II
- PAES

Applications

- knapsack
- TSP
- network processor design

The Handshake Protocol (Petri Net)

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

Available PISA Modules

http://www.tik.ee.ethz.ch/pisa
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)
Network Processors: Overview

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

- incoming flows (packet streams)
- routing / forwarding
- transcoding
- encryption / decryption
- real-time flows: e.g., voice
- non-real-time flows: e.g., sftp

**network processor (NP)**

Task Model: A Simple Network Processor

- **Execution chain**

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:
  \[ \forall t \in T \ \exists s \in S : binding(t, s) = true \]
  \[ \forall s \in S : binding(t, s) = true \Rightarrow allocation(s) = true \]

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

- **Allocation**: integer vector (1 per individual)
  
  
  
  
  for each resource type, number of allocated instances

- **Binding**: integer vector (1 per usage scenario)
  
  
  
  
  for each task and for each scenario, selected resource instance

- **Scheduling priorities**: permutation vector (1 per usage scenario)
  
  
  
  
  for each flow, its priority

**Variator Process: Mutation & Recombination**

- **Allocation**: integer vector
  
  - mutation: randomly choose an integer between 0 and \textit{max\_instances} per resource type
  
  - recombination: one-point crossover
  
  - appropriate decoder function ensures feasible allocations

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

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

**References**


