Interactive Multiobjective Optimization: Applications and Tools to Support Decision Making

Basic concepts of multiobjective optimization and problem formulation

University of Jyväskylä
Faculty of Information Technology

11.8.2025

Introductions

What is this course?

Course Title:
Interactive Multiobjective Optimization:
Applications and Tools to Support Decision Making

Course Code: JSS34

Lecturers:

- Juho Roponen
- Giovanni Misitano
- Bhupinder Singh Saini
- Giomara Lárraga
- Juuso Pajasmaa

Coordinator:

- Giovanni Misitano

Course Duration:
The course will span five days (one week).

What is expected?

Participants are expected to have prior knowledge of:

- Python (basic programming skills)
- Fundamentals of calculus
- Optimization and mathematical programming
- Basics of single-objective optimization

Learning outcomes

After completing the course, participants will have a solid understanding of multiobjective optimization and interactive methods to support decision-making. They will be familiar with both scalarization-based and evolutionary interactive methods, as well as various visualizations and graphical user interfaces to assist decision makers. Additionally, students will gain the necessary skills to apply the DESDEO framework in modeling and solving their own data-driven optimization problems.

What is taught?

Monday	Basic concepts of multiobjective optimization and problem formulation
Tuesday	Scalarization-based methods
Wednesday	Evolutionary multiobjective optimization
Thursday	Visualizations, user interfaces, other supporting tools
Friday	Hybridization of methods and more advanced methods

Each day will consist of lectures and practical sessions. Course attendees will also have a chance to complete an optional final project to earn additional credits.

- Lectures (3h/day) will be given before noon in the morning focusing on theory.
- Practical sessions (3h/day) are given in the afternoon where ideas discussed during the morning lectures are applied in practice.

Who am I?

- Doctor of Science in Technology from Aalto University.
- Major subject: Systems Analysis and Operations Research.
- Postdoctoral researcher in the Faculty of Information Technology in the Multiobjective Optimization research group since autumn 2023.
- Research topics: mathematical modeling, optimization, and simulation — especially decision analysis, game theory, and uncertainty modeling.
- Application areas of research: combat modeling, route optimization, nuclear safety, technology foresight, and most recently forest management.
- Teacher on the linear and discrete optimization course.

Who are you?

Let's do a round of introductions

- Who are you and what are you studying?
- What is the phase of your studies (MSc/PhD)?
- Previous experiences of (multiobjective) optimization?
- How is this course related to your studies/research?
- What are your expectations about this course?

Optimization

Optimization Problem

The goal of optimization (lat. optimum – the best) is to find the best possible solution within given constraints.

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{x}) \\ \text{Subject to} \quad & g(\mathbf{x}) \leq 0, \\ & h(\mathbf{x}) = 0. \end{aligned}$$

- The decision variable $\mathbf{x}$ (optimization variable) is typically a vector.
- The function $f(\mathbf{x})$ is the problem's objective function and measures the quality of a solution.
- Depending on the problem, the objective function may be minimized or maximized.

Applications of Optimization

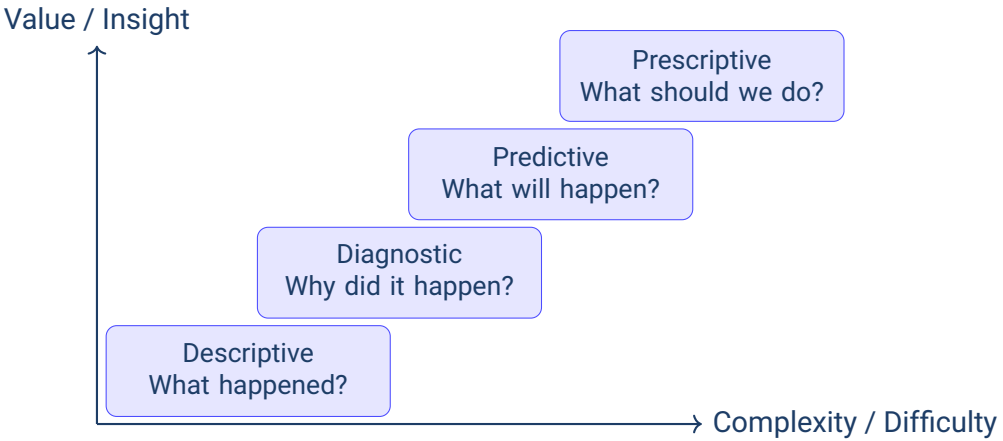
Machine Learning

- Finding the neural network that best fits the training data
- Finding the decision tree that best fits the training data
- Finding the surface that best separates the data into two classes (Classification)
- Finding the curve that best fits the data (Regression Analysis)

Operations Research

- Finding the fastest route from Jyväskylä to Forssa using the Finnish road network
- Creating a work schedule that minimizes overtime costs
- Selecting stocks for an investment portfolio
- Finding a cost-effective location for a logistics center

The Four Levels of Data Analysis



Constraints

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{x}) \\ \text{Subject to} \quad & g(\mathbf{x}) \leq 0, \\ & h(\mathbf{x}) = 0. \end{aligned}$$

- The function $g(\mathbf{x})$ is an inequality constraint.
- The function $h(\mathbf{x})$ is an equality constraint.
- There can be one, many, or no inequality or equality constraints.
- If the optimization problem has no constraints, it is called unconstrained optimization.

Solving an Optimization Problem

$$
\begin{aligned}
&\text{Minimize} && f(\mathbf{x}) && \\
&\text{Subject to} && g_i(\mathbf{x}) \leq 0, && i = 1, 2, \ldots, m \\
& && h_j(\mathbf{x}) = 0, && j = 1, 2, \ldots, l
\end{aligned}
$$

- When there are many constraints g_i and h_j, it can be difficult to find even a point $\mathbf{x}$ that satisfies all of them. (Feasible solution)
- Real-world optimization problems often involve numerous constraints and are always solved with a computer.
- Solving an optimization problem in practice can be very easy, because you just need to tell a computer what to minimize or maximize, and under which constraints.
- Formulating the problem is the difficult part, because the best formulation for a problem depends on the type of solver or algorithm used, and a proper formulation can enable the use of a more efficient solver.

Classes of Optimization Problems

- **Linear, continuous**
  Objective and constraints are linear; all variables are continuous.
- **(Mixed-)Integer Linear Programming (ILP/MILP)**
  Linear models with integer and possibly continuous variables; combinatorially hard.
- **Convex (e.g., Quadratic Programming)**
  Objective and feasible region are convex; globally solvable with efficient methods.
- **Mixed-Integer Convex Programming (MICP)**
  Convex functions combined with integer variables; significantly more complex.
- **Non-convex**
  Non-convex objective or constraints; global optimum is difficult to guarantee.
- **Black-box / Simulation-based Optimization**
  Objective and/or constraints cannot be algebraically expressed or evaluated directly; evaluated through simulations, heuristics, or experiments.

Example

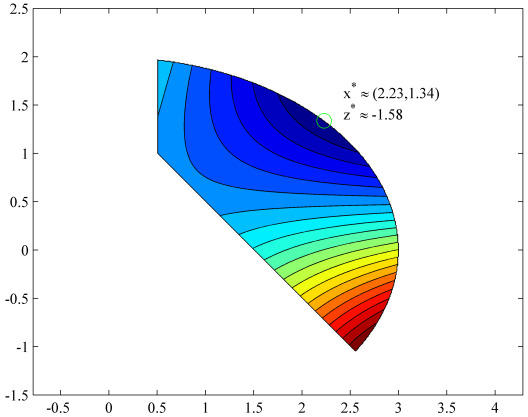
Feasible set of points:

$$S = \left\{ x \in \mathbb{R}^2 \mid x_1 \geq \frac{1}{2},\ x_1 + x_2 \geq \frac{3}{2},\ \left(\frac{x_1}{3}\right)^2 + \left(\frac{x_2}{2}\right)^2 \leq 1 \right\}$$

and the objective function:

$$f(x) = (1 - x_1)x_2 + x_1(1 - \sin x_2)$$

- This optimization problem has exactly one solution.
- How many solutions can an optimization problem generally have?

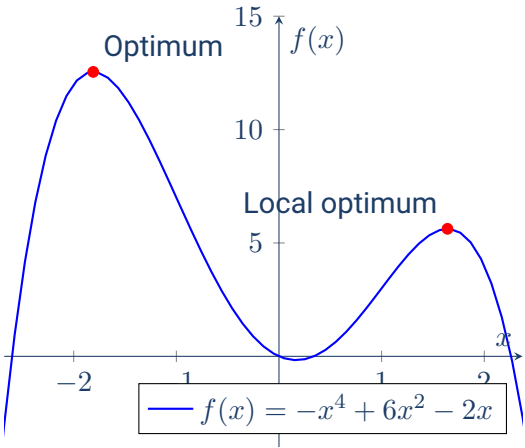


Number of Solutions

- An optimization problem has zero solutions if:
  - The set of feasible solutions is empty, i.e., $S = \emptyset$.
  - The objective function can decrease without bound, i.e., $\nexists z^* = \inf_{\mathbf{x} \in S} f(\mathbf{x})$.
  - The function has an infimum (greatest lower bound) in S, but it is never reached. (In this case, the infimum z^* may still be considered a solution.)
  - The same applies to maximization and supremum (least upper bound).
- One solution is most common, but it is often hard to verify.
- Some other integer number of solutions, e.g., if the function is periodic or otherwise non-monotonic.
- Infinitely many solutions, if the optimum is attained at infinitely many points. This can occur, for example, if a component x_i of the decision vector $\mathbf{x}$ does not affect $f(\mathbf{x})$.
- If a solution exists, the objective function $f(\mathbf{x})$ has only one (globally) optimal value.

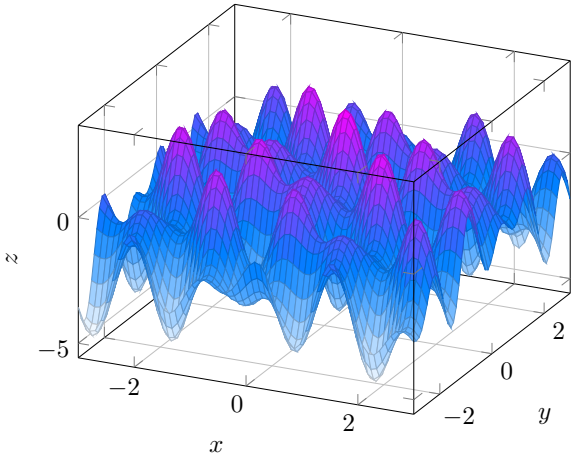
Local and Global Optima

- The objective function $f(\mathbf{x})$ often has only one global optimum value.
- However, it may also have one or more local optimum values.
- Depending on the solver, method, or algorithm used, a computer may return a local instead of a global optimum.



Multidimensional Problems

- In single-variable problems, local optima rarely cause major issues.
- The situation changes when the number of decision variables increases and objective functions become more complex.
- In real-world problems, the decision vector $\mathbf{x}$ may easily have thousands of elements.



Writing Optimization Problems

$$
\begin{aligned}
\text{Minimize} \quad & f(\mathbf{x}) \\
\text{Subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, 2, ..., m \\
& h_j(\mathbf{x}) = 0 \quad j = 1, 2, ..., l \\
& \mathbf{x} \geq 0 \\
& \mathbf{x} \in \mathbb{R}^n
\end{aligned}
$$

$$
\begin{aligned}
\min_{\mathbf{x} \in \mathbb{R}^n_+} \quad & f(\mathbf{x}) \\
\text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \leq 0 \\
& \mathbf{h}(\mathbf{x}) = 0
\end{aligned}
$$

$$
\min_{\mathbf{x} \in S} \quad f(\mathbf{x}),
$$

where $S = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \geq 0,\ \mathbf{g}(\mathbf{x}) \leq 0,\ \mathbf{h}(\mathbf{x}) = 0\}$.

All forms shown on this slide are equivalent. Different formulations appear in both literature and optimization software.

Optimization or Programming?

- Mathematical optimization and mathematical programming are synonyms.
  - Also true in other languages like Finnish: optimointi and ohjelmointi
- The same applies to linear optimization/programming as well as to discrete, dynamic, convex, etc.
  - Also in other languages
- Talking about programming in the context of purely mathematical problems can be confusing nowadays, but the term has historical roots™.
- Mathematical programming in general—and linear programming in particular—are older than programmable digital computers.

Multiobjective optimization

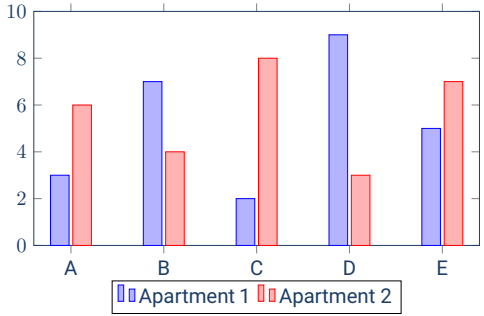
Multi-Objective Optimization Problem

$$
\begin{aligned}
\min \quad & f_1(\mathbf{x}) \\
\min \quad & f_2(\mathbf{x}) \\
\max \quad & f_3(\mathbf{x}) \\
\text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \leq 0, \\
& \mathbf{h}(\mathbf{x}) = 0.
\end{aligned}
$$

- A multi-objective optimization problem differs from other optimization problems in that it includes multiple objectives.
- Each objective is represented by a separate objective function.
- Like other optimization problems, multi-objective problems can also be linear, nonlinear, convex, smooth, robust, etc.

Example of a Multi-Objective Problem

- Suppose you're looking for an apartment in Jyväskylä.
- You want the apartment to be as A) cheap as possible, B) close to the university, C) close to the city center, D) quiet, and E) spacious.
- There will never be a single best solution, because the cheapest apartment in the city is certainly not also the largest.



This is not strictly speaking a multi-objective optimization problem but rather a multi-criteria decision problem, since the goal is simply to choose the best option among a set of known alternatives.

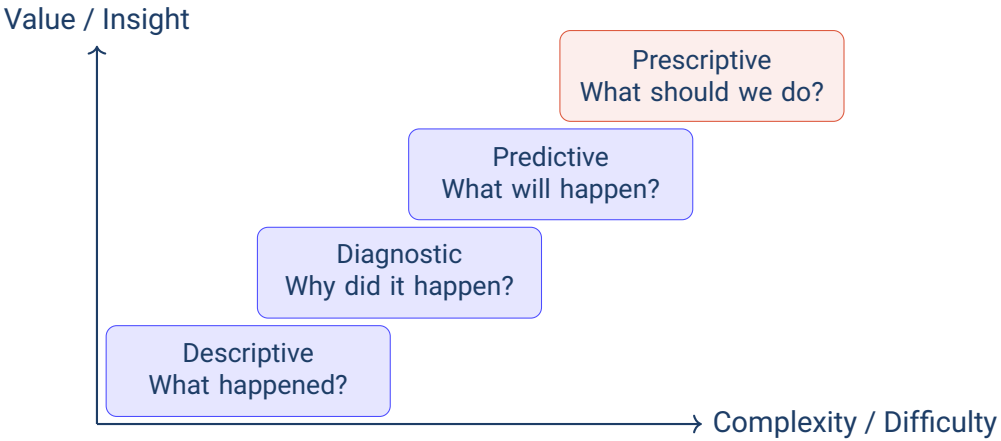
Multi-Objective Problems in the Real World

- Multi-objective optimization problems are very common in real-world decision-making.
- Most practical problems involve trade-offs between conflicting objectives.
- There is no single best solution, the goal is to find a compromise that the decision maker likes.

Examples of real-world multi-objective problems:

- Designing a product: minimize cost, maximize performance, minimize environmental impact
- Transportation planning: minimize travel time, minimize cost, minimize emissions
- Investment portfolio selection: maximize return, minimize risk, ensure liquidity
- Energy grid management: maximize reliability, minimize cost, minimize carbon footprint
- Machine learning model design: minimize prediction error, minimize complexity

The Four Levels of Data Analysis



Vector-Valued Objective Function

$$\begin{aligned} \min \quad & f_1(\mathbf{x}) \\ \min \quad & f_2(\mathbf{x}) \\ \max \quad & f_3(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \leq 0, \\ & \mathbf{h}(\mathbf{x}) = 0. \end{aligned}$$

$$\begin{aligned} \min \quad & \mathbf{f}(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \leq 0, \\ & \mathbf{h}(\mathbf{x}) = 0, \end{aligned}$$

where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^3$ and $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ -f_3(\mathbf{x}) \end{bmatrix}$

- Often, instead of listing several objective functions, a single vector-valued objective function is used in multi-objective optimization problems.
- This requires that all objective functions be either minimized or maximized.

Linear Multi-Objective Problem

$$
\begin{aligned}
\min \quad & \mathbf{c}_1^\top \mathbf{x} \\
\min \quad & \mathbf{c}_2^\top \mathbf{x} \\
\min \quad & \mathbf{c}_3^\top \mathbf{x} \\
\text{s.t.} \quad & A_{\text{ub}}\mathbf{x} \leq \mathbf{b}_{\text{ub}}, \\
& A_{\text{eq}}\mathbf{x} = \mathbf{b}_{\text{eq}}.
\end{aligned}
$$

$$
\begin{aligned}
\min \quad & C\mathbf{x} \\
\text{s.t.} \quad & A_{\text{ub}}\mathbf{x} \leq \mathbf{b}_{\text{ub}}, \\
& A_{\text{eq}}\mathbf{x} = \mathbf{b}_{\text{eq}},
\end{aligned}
$$

where $C = \begin{bmatrix} \mathbf{c}_1^\top \\ \mathbf{c}_2^\top \\ \mathbf{c}_3^\top \end{bmatrix}$

- A linear multi-objective problem can also be written so that the decision vector $\mathbf{x}$ is multiplied by a matrix instead of a vector in the objective function, resulting in a vector.

Comparing Vectors

- What does $\min \mathbf{f}(\mathbf{x})$ mean when $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$?
- The minimum of a real-valued function f is typically defined so that

$$f(x^*) = \min f(\mathbf{x}),$$

  if $f(\mathbf{x}^*) \leq f(\mathbf{x}),\ \forall \mathbf{x}$.
- With this definition, almost no vector-valued function would have a minimum, making their optimization very difficult.
- If we compare the vectors $\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, it is not true that $\mathbf{a} \leq \mathbf{b}$ or $\mathbf{a} \geq \mathbf{b}$.
- For example, the function $\mathbf{f}(x) = \begin{bmatrix} -x \\ x \end{bmatrix}$ would have no minimum over $\mathbb{R}$ or any nontrivial subset of $\mathbb{R}$.

$$\begin{aligned} \min \quad & \mathbf{f}(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \leq 0, \\ & \mathbf{h}(\mathbf{x}) = 0 \end{aligned}$$

Solution to a Multi-Objective Problem

- A multi-objective problem has no unique solution if all objectives are treated equally.
- However, not all solutions are equally good, because some solutions can be worse than others with respect to all objective functions.
- A general solution to the problem can be considered a set of solutions that includes all solutions that are not clearly worse than any other.
- For the function $\mathbf{f} : X \to Z$, the solution set would be

$$\begin{aligned} \min \quad & \mathbf{f}(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \leq 0, \\ & \mathbf{h}(\mathbf{x}) = 0 \end{aligned}$$

$$P(Z) = \left\{ \mathbf{z}^* = \mathbf{f}(\mathbf{x}^*) \,|\, \mathbf{x}^* \in X : \nexists \mathbf{x} \in X \text{ s.t. } \mathbf{f}(\mathbf{x}) \leq \mathbf{f}(\mathbf{x}^*) \text{ and } \mathbf{f}(\mathbf{x}) \neq \mathbf{f}(\mathbf{x}^*) \right\}$$

Pareto Dominance

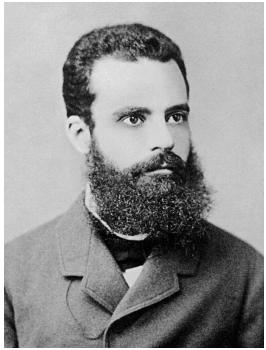
- Comparing vectors using only $\leq$ and $=$ relations is difficult.
- So we define a new relation $\prec$ for comparing vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, such that

$$\mathbf{a} \prec \mathbf{b} \Leftrightarrow a_i \leq b_i, \ \forall i \in \{1, ..., n\}$$

  and $\leq$ is $<$ for some i.
- This partial ordering between vectors is called Pareto dominance.
- $\mathbf{a} \prec \mathbf{b}$ is read as "a dominates b."
- The solution set of $\mathbf{f} : X \to Z$ can now be written as:

$$P(Z) = \left\{\mathbf{z}^* = \mathbf{f}(\mathbf{x}^*) \mid \mathbf{x}^* \in X : \nexists \mathbf{x} \in X \text{ s.t. } \mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}^*)\right\}$$

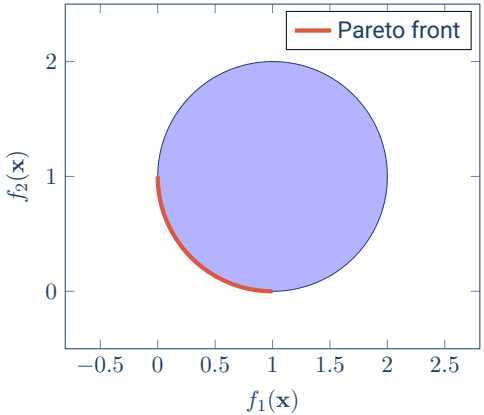
- The solution set of a vector-valued function is thus the set of Pareto non-dominated solutions.



Vilfredo Pareto
1848 – 1923

Pareto Front

- Solutions that are not Pareto dominated are also called Pareto efficient.
- The set of Pareto efficient solutions is called the Pareto front (or Pareto surface).
- If the multi-objective optimization problem has no more than 3 objectives, plotting the Pareto front can be a good way to visualize the trade-offs between efficient solutions.



Pareto front of two minimized functions.

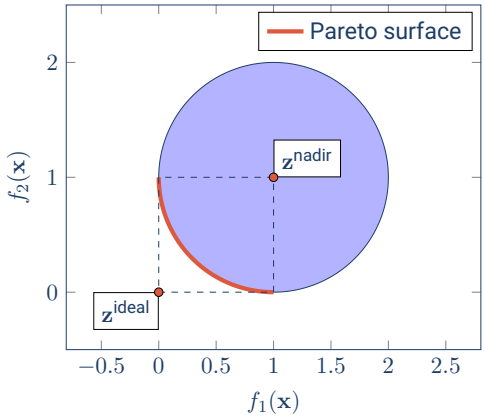
Ideal and Nadir

- The combination of the best objective function values found on the Pareto surface is called the ideal point:

$$\mathbf{z}^{\text{ideal}} = \begin{bmatrix} \min_{\mathbf{z} \in P(Z)} z_1 \\ \vdots \\ \min_{\mathbf{z} \in P(Z)} z_n \end{bmatrix}$$

- The combination of the worst objective function values found on the Pareto surface is called the nadir point:

$$\mathbf{z}^{\text{nadir}} = \begin{bmatrix} \max_{\mathbf{z} \in P(Z)} z_1 \\ \vdots \\ \max_{\mathbf{z} \in P(Z)} z_n \end{bmatrix}$$



Pareto surface of two minimized objective functions.

Calculating the Ideal Point

- The ideal point represents the best possible objective values, assuming each could be optimized independently.
- For an n-objective problem:

$$\mathbf{z}^{\text{ideal}} = \begin{bmatrix} \min_{\mathbf{x} \in X} f_1(\mathbf{x}) \\ \vdots \\ \min_{\mathbf{x} \in X} f_n(\mathbf{x}) \end{bmatrix}$$

- Each objective function f_i is minimized individually over the feasible set X.
- This typically requires solving n separate single-objective optimization problems.
- As such, computing the ideal point is relatively easy, especially compared to the nadir.

Calculating the Nadir Point

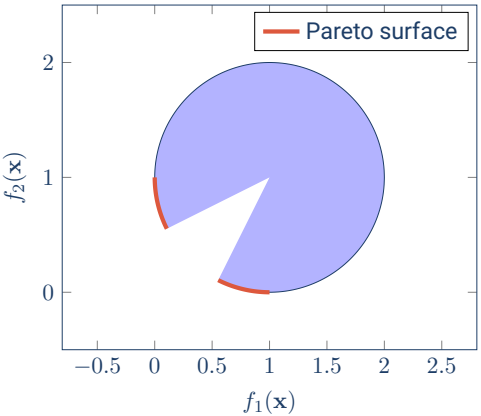
- The nadir point represents the worst (i.e., largest) values of each objective among Pareto-optimal solutions.
- For an n-objective problem:

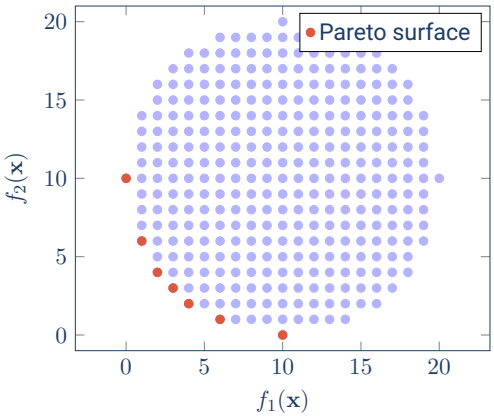
$$\mathbf{z}^{\text{nadir}} = \begin{bmatrix} \max_{\mathbf{z} \in P(Z)} z_1 \\ \vdots \\ \max_{\mathbf{z} \in P(Z)} z_n \end{bmatrix}$$

- Unlike the ideal point, the nadir requires knowledge of the entire Pareto front $P(Z)$.
- This makes it much harder to compute exactly — especially for nonlinear or discrete problems.
- In practice, the nadir point is often approximated using a sample of known Pareto-optimal solutions.

Different Pareto Surfaces

The Pareto surface of convex multiobjective optimization problems is continuous, but that of non-convex problems usually is not.





Calculating Pareto Surfaces

- Calculating the Pareto surface for a multiobjective optimization problem is usually not easy.
- For certain types of problems, it is possible to construct a mathematical expression that represents all Pareto points, but typically one must settle for approximations of the Pareto surface.
- For example, all Pareto-optimal solutions of a constrained continuous linear multiobjective problem can be represented as suitable convex combinations of the Pareto-optimal extreme points of the solution set.
- Optimization solvers and algorithms typically return only a single solution.
- There are, for example, evolutionary multiobjective optimization algorithms designed to approximate the entire Pareto surface at once.

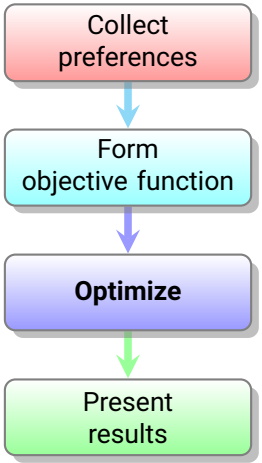
Solving multiobjective problems

Unique Solution

- A multiobjective problem has no mathematically unique solution, but in reality, we usually want to find a single solution.
- Multiobjective optimization methods typically follow one of four approaches to find a solution:
  - Preference-free methods search for a compromise solution when it's not possible to obtain information about the relative importance of the objectives from the decision-maker.
  - In a priori methods, preference information is collected before optimization, and an attempt is made to find a solution that matches those preferences.
  - In a posteriori methods, a representative set of Pareto-optimal solutions is first computed, and then, after optimization, the best one is selected based on the decision-maker's preferences.
  - In interactive methods, the decision-maker searches for their best solution iteratively.

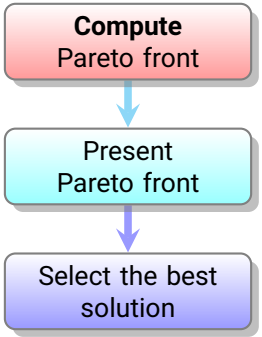
A Priori Methods

- A priori methods transform a multiobjective problem into a standard single objective optimization problem using information about the decision-maker's preferences.
- This sounds like a perfect way to solve all the challenges of multiobjective optimization.
- The optimization itself is usually easy.
- However, modeling the preferences of the decision-maker as a function reliably and correctly is extremely difficult.
- There are systematic methods for eliciting preferences, but they are either cumbersome to use or just not very good.



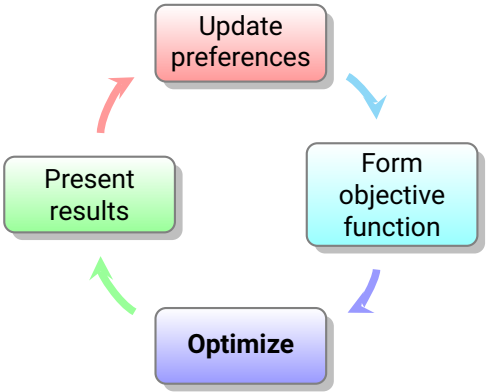
A Posteriori Methods

- A posteriori methods first compute the Pareto front of the multiobjective problem and then select the best result based on the decision-maker's preferences.
- In practice, computing the full Pareto front is usually impossible, so the goal is to compute a representative approximation of the Pareto-optimal solutions.
- Estimating the Pareto front is computationally difficult for complex optimization problems.
- Visually presenting the approximation is also challenging when there are many objectives.
- Many "a posteriori" methods focus solely on approximating the Pareto front and ignore how to actually make the final decision.



Interactive Methods

- Interactive methods work iteratively.
- Based on the decision-maker's expressed preferences, a Pareto-optimal solution is computed and shown to the decision-maker, who either accepts the result or updates their preferences.
- A strength of this approach is that the decision-maker can update their preferences as they learn what is realistically achievable.
- If optimization takes a long time, the whole iterative process becomes time-consuming.



The Best Multiobjective Optimization Method

- Different approaches to multiobjective optimization have different strengths and weaknesses.
- Choosing the best multiobjective optimization method is itself a multiobjective optimization problem.
- No one has yet succeeded in deciding what the best method for choosing the best method is, so the question remains open.
- The multiobjective optimization research group here at the University of Jyväskylä mainly focuses on interactive methods.

  https://optgroup.it.jyu.fi/

Scalarization

- Optimization algorithms and solvers are generally not designed to solve multiobjective problems directly.
- Therefore, multiobjective problems are often solved by defining a real-valued function that somehow evaluates how good a solution is with respect to the original problem.
- These scalar-valued functions are called scalarizing functions.
- Using a scalarization function $g : \mathbb{R}^m \to \mathbb{R}$, a multiobjective problem with m objectives can be scalarized into a single-objective problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & g(\mathbf{f}(\mathbf{x})) \\ \text{s.t.} \quad & \mathbf{x} \in S \end{aligned}$$

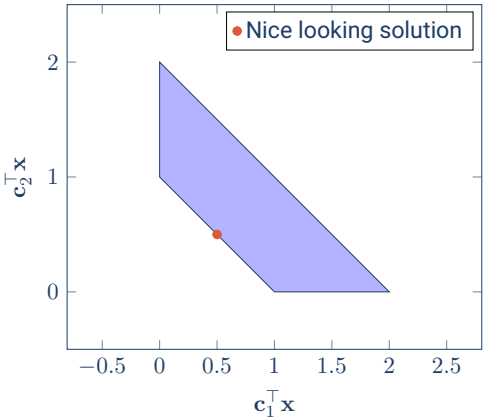
- Scalarization often also involves additional constraints.

Linear Example

- The figure shows the feasible set for a linear multiobjective problem:

$$\min_{\mathbf{x}} \quad C\mathbf{x}$$
$$A\mathbf{x} \leq \mathbf{b}$$

- We want to find a solution that balances the objectives, ideally from the center of the Pareto front.
- What kind of scalarizing function could help us find such a point, if we don't know the Pareto front in advance?



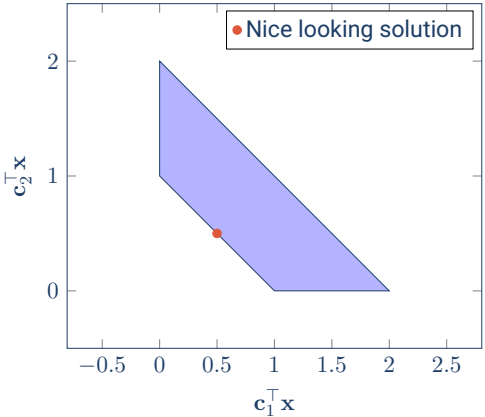
Two linear objective functions to minimize.

Weighted Sum

- A common first idea is to optimize a weighted sum of the objectives:

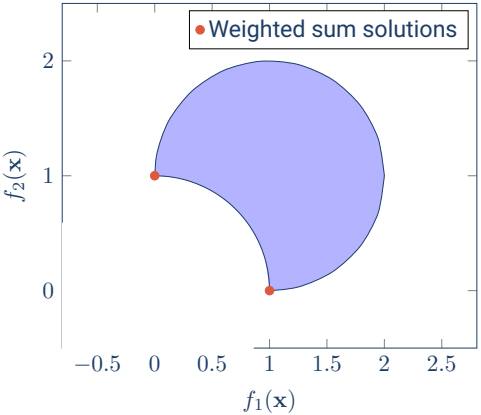
$$\min_{\mathbf{x}\in S} \sum_{i=1}^{m} w_i f_i(\mathbf{x}),$$

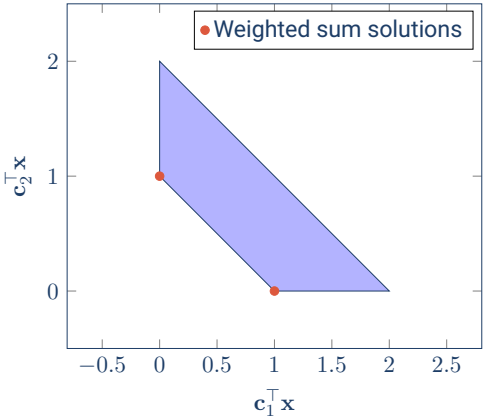
  where the weights $w_i > 0$ are parameters of the scalarization.
- Also known as linear scalarization.
- The downside is, that this idea is often pretty terrible.



Two linear objective functions to minimize.

Problems with the Weighted Sum





Weighted sum can never find solutions on the non-convex parts of the Pareto front.
In linear problems, it typically only finds corner points, because of how solvers work.

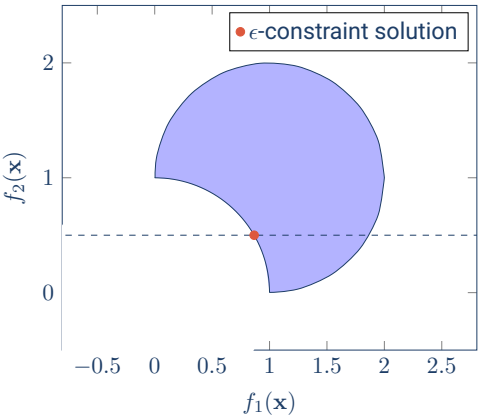
ϵ-constraint Method

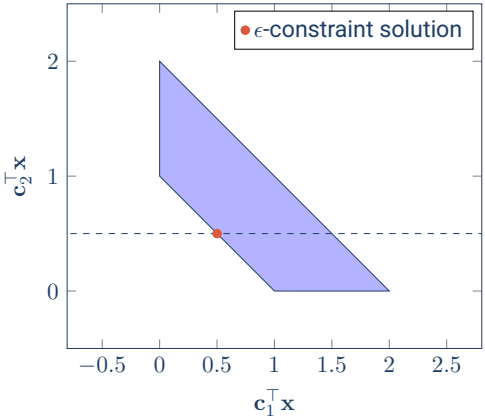
- If you want a simple method for computing Pareto points, use the ϵ-constraint method.
- One of the objectives f_j is selected for optimization, and constraints are added for all others:

$$\begin{aligned} \min_{\mathbf{x} \in S} \quad & f_j(\mathbf{x}) \\ \text{s.t.} \quad & f_i(\mathbf{x}) \leq \epsilon_i, \qquad \forall i \in \{1, ..., m\} \setminus \{j\} \end{aligned}$$

- By varying the ϵ-constraints between the ideal and nadir values, a good approximation of the entire Pareto front can be computed.
- When there are a lot of objectives, approximating the Pareto front becomes difficult, because the number of required problem instances increases exponentially.
- The method can also struggle to find solutions that match decision maker's preferences.

Solutions from the ϵ-Constraint Method





Both plots include the constraint $f_2(\mathbf{x}) \leq 0.5$. In the linear case, it's easy to find a point from the center of the Pareto front. In the nonlinear case, it may require multiple attempts.

p-norm

- Many scalarization methods are based on minimizing the distanec to some reference point.
- Distances between vectors are usually measured with a weighted p-norm:

$$||\mathbf{z} - \mathbf{z}'||_{\mathbf{w},p} = \left(\sum_{i=1}^{m} \left(w_i \, |z_i - z_i'| \right)^p \right)^{\frac{1}{p}}$$

- Common choices: $p = 1$, 2, and ∞
  - $p = 1$ is poor. Why?
  - $p = 2$ is the Euclidean norm – the "usual" distance.
  - $p = \infty$ is:

$$||\mathbf{z} - \mathbf{z}'||_{\mathbf{w},\infty} = \max \left[w_i |z_i - z_i'| \right]$$

Method of Global Criterion

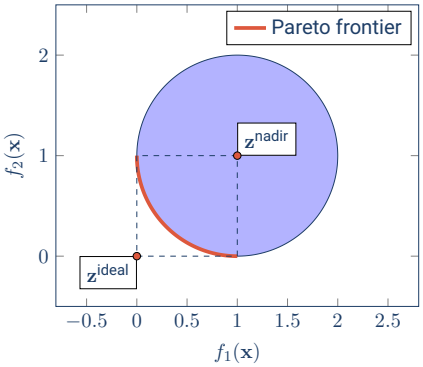
- The method of global criterion is an example of a preference-independent method.
- It minimizes the distance to the ideal point:

$$\min_{\mathbf{x} \in S} ||\mathbf{f}(\mathbf{x}) - \mathbf{z}^{\text{ideal}}||,$$

where $|| \cdot ||$ is a weighted p-norm.

- Typically, the weighting vector $\mathbf{w}$ is used to normalize the objective values to be in $[0, 1]$ range to make them comparable:

$$w_i = \frac{1}{|z_i^{\text{ideal}} - z_i^{\text{nadir}}|}$$



Minimizing Distance to a Target

- Suppose the decision-maker wants a solution that is similar to a target point $\bar{\mathbf{z}}$.
- In that case, we can minimize the distance to the target:

$$\min_{\mathbf{x} \in S} ||\mathbf{f}(\mathbf{x}) - \bar{\mathbf{z}}||,$$

  where $|| \cdot ||$ is some weighted p-norm.
- This is exactly the same optimization problem as in the global criterion method, but the reference point is different.
- The weights w_i can be used to normalize the objective function values between the ideal and nadir, or to emphasize the importance of certain objectives.
- By varying the target $\bar{\mathbf{z}}$, the method can also be used to approximate the Pareto front.

Other Scalarizing Functions

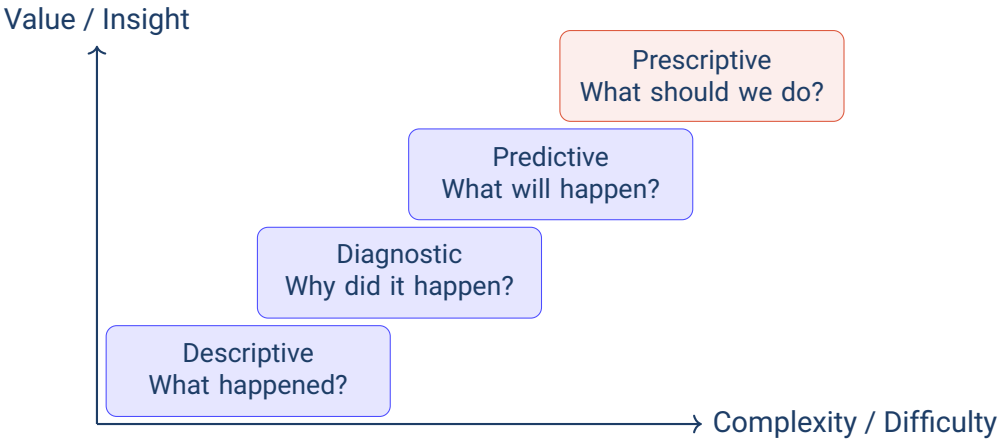
- Perhaps the most commonly used scalarization method not yet mentioned is replacing the objective functions with a utility function $u\big(\mathbf{f}(\mathbf{x})\big)$.
- The utility function is intended to accurately represent how the possible values of the objective functions relate to each other in terms of the decision-maker's preferences.
- A priori methods are typically based on defining utility functions.
- In stochastic optimization, one usually maximizes the expected value of the utility function.
- In practice, defining a utility function is often difficult, especially if the Pareto front of the problem is not well known.
- You are going to learn more about scalarization methods tomorrow.

Formulating optimization problems

The Two Hardest Problems in Optimization

- In principle, optimization is simple: tell the computer what you want and let the solver handle the rest.
- In practice, the hardest parts are:
  1. **Deciding what to optimize**
     What are the actual goals and trade-offs the decision maker is interested in? What outcomes are desirable?
  2. **Formulating the problem**
     How can the real-world situation be modeled with mathematical variables, objectives, and constraints?
- If either of these steps is done poorly, even the most powerful solver won't give useful results.
- Optimization is as much about modeling and understanding the problem as it is about computation.

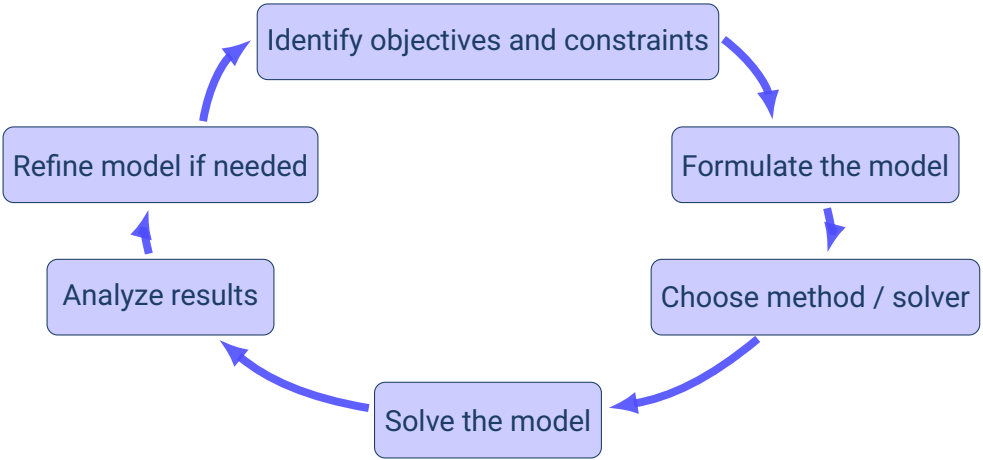
The Four Levels of Data Analysis



The Importance of Problem Formulation

- Optimization results are only as good as the model behind them.
- In practice, problem formulation is often an iterative process, refined as more information becomes available.
- Especially in engineering, the importance of problem formulation is frequently underestimated.
- What you shouldn't do:
  Start by listing decision alternatives, then define objectives to evaluate them.
  (Alternative-focused thinking)
- What you should do:
  First identify what values and objectives matter in decision-making, then use systematic methods like optimization to find suitable alternatives.
  (Value-focused thinking)

The Optimization Process



Number of Objectives

- In multiobjective optimization, the objectives to be optimized are assumed to be conflicting.
  - Correlated objectives can often be removed.
- Typically, only 2–3 objectives are used.
  - Many MOO methods perform well mainly for 2–3 objectives.
  - Visualizing the Pareto front becomes difficult if there are more than 3 objectives.
  - However, nowadays there are many methods that can handle more objectives.
- Sometimes, there are too many objectives to consider.
  - Cognitive limits of the decision maker are exceeded [1]
  - Dimension reduction is needed — keep only the most relevant objectives.

[1] Miller, The magical number seven, plus or minus two: Some limits on our capacity for processing information, Psychological Review, 1956

Group Exercise: Identifying Objectives

Task

Imagine that you are choosing your Master's or PhD thesis (or a project) topic.

- What are the values that matter to you?
- What makes a topic good or bad?
- Write down as many values as you can in **5 minutes**.

Select Relevant Values

- Is of interest to you
- Work with interesting people
- Is a good match with your talents
- Improves your future career prospects
- Allows you to learn new stuff
- Is of interest to other people
- Knowledgeable supervisors are available and willing to teach you
- Research freedom from supervisor
- Involves travelling
- Funding available to do research
- Internationally reputed supervisor
- Reasonable amount of work
- Allows you to get good grades
- Is challenging enough
- There is good literature available
- Fits your previous studies
- Not studied recently at your institute
- Helps make good networking contacts
- Helps select a job after graduation
- Easy to finish thesis
- Industry oriented

Most important objectives

Now choose your top three most important values from all the values that you have selected.

Reflection Questions

Now answer the following questions:

- How many values did you generate when you first wrote them down?
- How many values did you select from the list?
- How many of the three most important values were among those you first wrote down?

Summary of Similar Exercises

- Participants often generate only about half of the relevant values initially.
- The missed values are not trivial; they are roughly as important as those first identified.
- This occurs even when decision makers think they know the problem well (e.g., professional and personal decisions).
- Initial studies on this were conducted with MBA students selecting internships. [2]

[2] Samuel D. Bond, Kurt A. Carlson, Ralph L. Keeney, Generating objectives: can decision makers articulate what they want?, Management Science 54, pages 56-70, 2008

Formulating the Correct Problem

- Choosing the right objectives, data sources, and problem elements is crucial.
- Consider data availability and whether objectives and constraints can be evaluated mathematically or otherwise.
- A well-formulated problem ensures optimization aligns with real-world goals and constraints.
- Poorly chosen objectives or inaccurate data can cause even the best methods to produce useless solutions.
- Careful problem formulation is a foundational step that directly impacts the success of the optimization process.

The Mathematical Formulation

- The mathematical form of the optimization problem strongly affects its solvability.
- Specific problem types (e.g., linear, convex, quadratic) are easier to solve with existing solvers and algorithms.
- Clever formulation can transform a hard problem into a more tractable one, enabling the use of powerful, efficient methods.
- Understanding solver capabilities and tailoring your model accordingly can significantly improve solution quality and computational performance.

Key takeaway: Problem formulation is not just about modeling reality, but also about leveraging mathematical structures to your advantage.

Classes of Optimization Problems

- **Linear, continuous**
  Objective and constraints are linear; all variables are continuous.
- **(Mixed-)Integer Linear Programming (ILP/MILP)**
  Linear models with integer and possibly continuous variables; combinatorially hard.
- **Convex (e.g., Quadratic Programming)**
  Objective and feasible region are convex; globally solvable with efficient methods.
- **Mixed-Integer Convex Programming (MICP)**
  Convex functions combined with integer variables; significantly more complex.
- **Non-convex**
  Non-convex objective or constraints; global optimum is difficult to guarantee.
- **Black-box / Simulation-based Optimization**
  Objective and/or constraints cannot be algebraically expressed or evaluated directly; evaluated through simulations, heuristics, or experiments.

Mathematical tricks for problem formulation

Converting Complex Optimization Problems to MILP

- Many real-world optimization problems are mathematically complex, nonlinear, or non-convex.
- Such problems are often difficult to solve directly with standard solvers.
- By introducing binary variables and techniques like the big-M method, these problems can be reformulated as Mixed-Integer Linear Programs (MILPs).
- This transformation enables the use of powerful MILP solvers, leveraging advances in integer programming.
- Key advantages:
  - Model logical conditions, switches, and piecewise linear functions.
  - Handle non-convexities by encoding them through integer variables.
  - Gain access to powerful commercial MILP solvers.
- However, care must be taken with big-M values to ensure numerical stability and solver efficiency.

Making an Inequality Constraint Ineffective

Consider the inequality constraint

$$A\mathbf{x} \leq \mathbf{b}.$$

How can we make the constraint active only when some condition is met?

The inequality constraint can be made ineffective by adding a large number M to one side:

$$A\mathbf{x} \leq \mathbf{b} + M.$$

If M is larger than any possible value of $A\mathbf{x}$ in the problem context, the constraint becomes meaningless.

By controlling whether to add M using a binary variable y, we can switch the constraint off or on as desired:

$$A\mathbf{x} \leq \mathbf{b} + My$$
$$y \in \{0, 1\}$$

Making an Equality Constraint Ineffective

Consider the equality constraint

$$A\mathbf{x} = \mathbf{b}.$$

How can we switch an equality constraint on or off when we can't just add a large number to one side?

The equality constraint can be replaced by two inequalities:

$$A\mathbf{x} \leq \mathbf{b}$$
$$A\mathbf{x} \geq \mathbf{b}$$

Both inequalities can be controlled by the same binary variable y, which determines whether the constraints are active:

$$A\mathbf{x} \leq \mathbf{b} + My$$
$$A\mathbf{x} \geq \mathbf{b} - My$$
$$y \in \{0, 1\}$$

Switching Between Constraints

Suppose we have two inequality constraints

$$A'\mathbf{x} \leq \mathbf{b}'$$

and

$$A''\mathbf{x} \leq \mathbf{b}'',$$

and we want sometimes one to be active, sometimes the other.

How can this be done?

By controlling with a binary variable y whether a large number M is added to the constraint or not, we can make constraints ineffective as needed:

$$\begin{aligned} A\mathbf{x} &\leq \mathbf{b} + My \\ y &\in \{0,1\} \end{aligned}$$

By linking two different constraints to the same binary variable, we ensure exactly one constraint is active:

$$\begin{aligned} A'\mathbf{x} &\leq \mathbf{b}' + My \\ A''\mathbf{x} &\leq \mathbf{b}'' + M(1-y) \\ y &\in \{0,1\} \end{aligned}$$

Switching the Objective Function

The objective function can also be switched depending on the situation between

$$\min \mathbf{c}'^{\top}\mathbf{x}$$

and

$$\min \mathbf{c}''^{\top}\mathbf{x}.$$

By replacing the original objective or part of it with a decision variable z, the objective can be moved into constraints, allowing the same tricks that are possible with constraints:

$$\begin{aligned} \min_{\mathbf{x},y,z} \quad & z \\ \text{s.t.} \quad & z \geq \mathbf{c}'^{\top}\mathbf{x} - My \\ & z \geq \mathbf{c}''^{\top}\mathbf{x} - M(1-y) \\ & \mathbf{x} \in \mathbb{R}^n,\ y \in \{0,1\},\ z \in \mathbb{R} \end{aligned}$$

Usually there are also other constraints limiting $\mathbf{x}$ and y:

$$A\mathbf{x} + Dy \leq \mathbf{b}$$

Warning about the Big-M Constant

- Using a large constant M (sometimes several different values) to switch parts of a problem on and off allows many flexible problem formulations.
- However, using the big-M constant can cause numerical instability if M is not chosen carefully.
- The best M is usually one that is just large enough to do its job.
- There is no universal rule for choosing a good M in complex problems.
- It is advisable to test and verify that the solution remains the same when M is varied in a range where it should not affect the outcome.

Max

- Constraint $z \geq \max(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$, can be represented by linear constraints

$$z \geq x_i, \quad \forall i = 1, ..., n$$

- However, $\max$ is not a linear function, so the constraint $z = \max(\mathbf{x})$ cannot be represented in a continuous linear problem.
- By introducing a binary vector $\mathbf{y}$ and a big number M

$$\begin{aligned} z &\geq x_i && \forall i = 1, ..., n \\ z &\leq x_i + (1 - y_i)M && \forall i = 1, ..., n \\ \sum_{i=1}^{n} y_i &= 1 \end{aligned}$$

we can formulate the equivalent constraints in a MILP.

Min

- The $\min$ function is very similar to $\max$.
- By introducing a binary vector $\mathbf{y}$ and a big number M

$$z \leq x_i \qquad \forall i = 1, ..., n$$
$$z \geq x_i + (1 - y_i)M \qquad \forall i = 1, ..., n$$
$$\sum_{i=1}^{n} y_i = 1$$

  we can formulate the $\min$ constraints in a MILP.
- Note that both $\min$ and $\max$ formulated this way can often cause numerical issues in the solver.
- Often, using only the upper or lower bound is sufficient for the optimization problem, reducing the likelihood of such issues.

Absolute Value

- The absolute value function can be expressed as

$$|x| = \max(x, -x).$$

- Since we already showed how to represent $\max$ using MILP with binary variables and big M, the same approach applies here.
- Introduce a binary variable $y \in \{0, 1\}$ and big M, then formulate:

$$\begin{aligned} z &\geq x \\ z &\geq -x \\ z &\leq x + (1 - y)M \\ z &\leq -x + yM \\ y &\in \{0, 1\} \end{aligned}$$

- Similarly to $\min$ and $\max$, one should only use the upper or the lower bounds if both are not needed.

Multiplying by a Binary Variable

- Consider $\mathbf{x} \in \mathbb{R}^n_+$ and $y \in \{0, 1\}$.
- We want the constraint $\mathbf{z} = y \cdot \mathbf{x}$, where $\mathbf{z} \in \mathbb{R}^n$.
- The corresponding linear constraints for the upper bound are

$$\begin{aligned} \mathbf{z} &\leq \mathbf{x} \\ \mathbf{z} &\leq My, \end{aligned}$$

  where M is a big number (upper bound for $\mathbf{x}$ is a good candidate).
- For the lower bound, we have

$$\begin{aligned} \mathbf{z} &\geq \mathbf{x} - M(1 - y) \\ \mathbf{z} &\geq 0. \end{aligned}$$

Multiplying by an Integer Variable

- We want to find linear constraints for $\mathbf{z} = y' \cdot \mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^n_+$ and $y' \in \{0, 1, \ldots, m\}$ are decision variables.
- Introduce a binary vector $\mathbf{y} \in \{0, 1\}^m$.
- The corresponding linearized upper and lower bounds are

$$\mathbf{z} \leq j\mathbf{x} + (1 - y_j)M \qquad \forall j = 1, ..., m$$
$$\mathbf{z} \leq M \sum_{j=1}^{m} y_j$$
$$\sum_{j=1}^{m} y_j \leq 1.$$

$$\mathbf{z} \geq j\mathbf{x} - (1 - y_j)M \qquad \forall j = 1, ..., m$$
$$\mathbf{z} \geq 0$$
$$\sum_{j=1}^{m} y_j \leq 1.$$

- This is such a cursed technique that it's probably not taught anywhere.

Multiplication with Binary Encoding

- We want linear constraints for $\mathbf{z} = y' \cdot \mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^n_+$ and $y' \in \{0, 1, \ldots, m'\}$ are decision variables.
- Define $y' := 1y_0 + 2y_1 + 4y_2 + \cdots + 2^m y_m$, where $m = \lfloor \log_2(m') \rfloor$ and $y_j \in \{0, 1\}$ for all $j = 0, 1, \ldots, m$.
- The corresponding linearized upper and lower bounds are

$$\mathbf{z}_j \leq 2^j \mathbf{x} \qquad \forall j = 0, 1, ..., m$$

$$\mathbf{z}_j \leq M y_j \qquad \forall j = 0, 1, ..., m$$

$$\mathbf{z} \leq \sum_{j=0}^{m} \mathbf{z}_j$$

$$\sum_{j=0}^{m} 2^j y_j \leq m'.$$

$$\mathbf{z}_j \geq 2^j \mathbf{x} - (1 - y_j) M \qquad \forall j = 0, 1, ..., m$$

$$\mathbf{z}_j \geq 0 \qquad \forall j = 0, 1, ..., m$$

$$\mathbf{z} \geq \sum_{j=0}^{m} \mathbf{z}_j$$

$$\sum_{j=0}^{m} 2^j y_j \geq m'.$$

Logic with Binary Variables

- It is probably not surprising that binary variables can also be used to model logical truth values.
- Let x_i be a binary variable that equals 1 if event i occurs, and 0 otherwise.
- For example, to form linear constraints corresponding to the statement $y = x_1 \wedge x_2$ (i.e., x_1 AND x_2), we use:

$$y \leq \frac{x_1 + x_2}{2}$$
$$y \geq x_1 + x_2 - 1$$
$$y, x_1, x_2 \in \{0, 1\}$$

Logical Operations

- The OR operation can also be modeled.
- The statement $y = x_1 \vee x_2$ corresponds to:

$$\begin{aligned} y &\geq \frac{x_1 + x_2}{2} \\ y &\leq x_1 + x_2 \\ y, x_1, x_2 &\in \{0, 1\} \end{aligned}$$

- Negation $y = \neg x$ corresponds to:

$$\begin{aligned} y &= 1 - x \\ y, x &\in \{0, 1\} \end{aligned}$$

Logical Operations with n Variables

- AND and OR operations can also be generalized to n variables:

$$y \leq \frac{\sum_{i=1}^{n} x_i}{n}$$
$$y \geq 1 - n + \sum_{i=1}^{n} x_i$$
$$y, x_i \in \{0,1\}, \quad i = 1, \ldots, n$$

$$y \geq \frac{\sum_{i=1}^{n} x_i}{n}$$
$$y \leq \sum_{i=1}^{n} x_i$$
$$y, x_i \in \{0,1\}, \quad i = 1, \ldots, n$$

- More complex operations can be formed, such as requiring at least m of the n variables to be true:

$$y \leq \frac{\sum_{i=1}^{n} x_i}{m}$$
$$y \geq \frac{1 - m + \sum_{i=1}^{n} x_i}{n - m + 1}$$

Answering Questions with Binary Variables

- Logical operations become significantly more useful if they can be linked to variables that are not binary.
- For this, it is usually necessary to add a binary variable representing some interesting property of the decision variables.
- For example, to know whether $2x_1 + 3x_2 \geq 6$, we can encode this information in a binary variable y with the constraints:

$$2x_1 + 3x_2 \leq 6 + My$$
$$2x_1 + 3x_2 \geq 6 - M(1 - y)$$

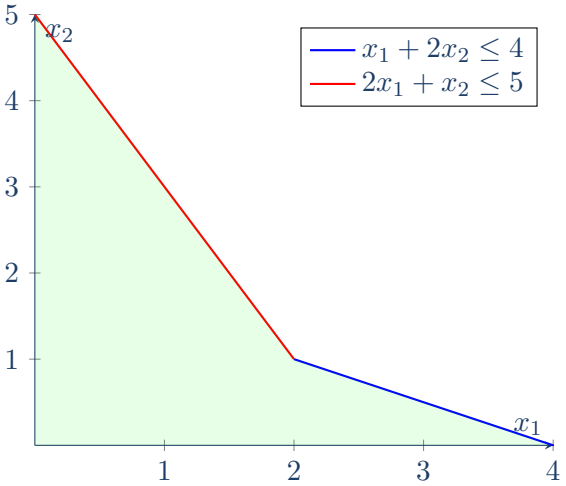
- With these constraints, y can be either 0 or 1 if $2x_1 + 3x_2 = 6$, but this can be fixed by adding a small number ϵ on the right side of the lower inequality.
- Many solvers, such as Gurobi, also support indicator variables that model logical constraints without the need for big M or small ϵ, both of which can introduce numerical instability.

Piecewise Linear Functions

- We want to define the non-convex feasible region shown in the figure.
- This can be done using binary variables y_i that activate constraints depending on the value of x_1.

$$
\begin{aligned}
My_1 &\geq 2 - x_1 \\
My_2 &\geq x_1 - 2 \\
x_1 + 2x_2 &\leq 4 + M(1 - y_1) \\
2x_1 + x_2 &\leq 5 + M(1 - y_2)
\end{aligned}
$$

- This feasible region could also be defined by having exactly one of the two constraints be active at a time.



Linearization of a Product

$$\log\left(\prod x_i\right) = \sum \log(x_i),$$

- The logarithm is a monotonic function (strictly increasing), meaning

$$\log(a) > \log(b) \Leftrightarrow a > b, \quad \forall a, b \in \mathbb{R}_+.$$

- From this it follows that the optimal solution x^* to the problem $\max_{\mathbf{x}} \log\left(\prod x_i\right)$ is the same as the solution to $\max_{\mathbf{x}} \prod x_i$.
- Thus, if we take the decision variables as $y_i := \log(x_i)$, we obtain a solution to the original problem by solving the seemingly simpler linear problem

$$\max_{\mathbf{y}} \sum y_i.$$

- Defining the constraints, however, may be challenging.

Other Useful Monotonic Functions

- A function $f : \mathbb{R} \to \mathbb{R}$ is monotonic increasing if for all x, y,

$$x \leq y \implies f(x) \leq f(y).$$

- Some useful examples of monotonic increasing functions (on typical domains):
  - $f(x) = x^n$, for $x \geq 0$
  - $f(x) = \sqrt[n]{x}$, for $x \geq 0$
  - $f(x) = e^x$, for all $x \in \mathbb{R}$
  - $f(x) = \log(x)$, for $x > 0$ (already discussed)
- In all these examples the function value is minimized (maximized) when x is minimized (maximized).
- Such functions preserve the order and can be used for transformations in optimization problems.

Surrogate Models in Optimization

- Surrogate models (also called *metamodels* or *response surfaces*) are approximations of expensive or complex objective functions and/or constraints.
- They are used when the true function is:
  - Expensive to evaluate (e.g., simulation, physical experiment)
  - Noisy or black-box
  - Not available in closed form
- Surrogate models are typically have a closed form and are much cheaper to evaluate and thus can be optimized more easily.
- Common surrogate modeling techniques:
  - Gaussian Processes (Kriging)
  - Polynomial Regression
  - Radial Basis Functions
  - Neural Networks

Embedding ML Models into MILP

- Machine learning models can be embedded into MILP to combine data-driven models with optimization.
- **Decision Trees / Random Forests:**
  - Encode tree structure as logical constraints.
  - Use binary variables to represent traversal through nodes.
  - Leaf predictions modeled with continuous or integer variables.
- **Neural Networks (ReLU-based):**
  - Represent each neuron's output using piecewise-linear constraints.
  - ReLU activation: $z = \max(0, x)$ can be modeled using big-M formulation and binary variables.
  - Deep networks require many variables and constraints.
- **Other Models:**
  - Support Vector Machines: encode hinge loss constraints.
  - Linear/Logistic Regression: directly representable in MILP.
- Challenges: Model size, numerical stability, and solver scalability.