Unit 1: Mathematical Programming Formulation and Complexity

Learning goals – Unit 1

I. What is the subject area of multiobjective decision analysis and multiobjective optimization?
II. What is a linear programming problem? How can we solve it graphically?
III. Geometrical meaning of active/non-active constraints.
IV. What are the different types of optimization problems?
V. How can we formulate multiobjective optimization problems?
VI. Why is it hard to solve such problems?

Motivation: Some Multicriteria Problems

What are criteria in these problems? What is the set of alternatives? Why is there a conflict?

(A) Select the best travel destination from a catalogue:

Seach space: Catalogue

Criteria: Sun→max, DistanceToBeach→min, and Travel Distance→min

Constraints: Budget, Safety

Motivation: Some Multicriteria Problems

What are criteria in these problems? What is the set of alternatives? Why is there a conflict?

(B) Find a optimal molecule in de-novo drug discovery:

Search space: All drug-like molecules (chemical space)

Criteria: Effectivity →max, SideEffects→min, Cost→min

Constraints: Stability, Solubility in blood, non-toxic

Motivation: Some Multicriteria Problems

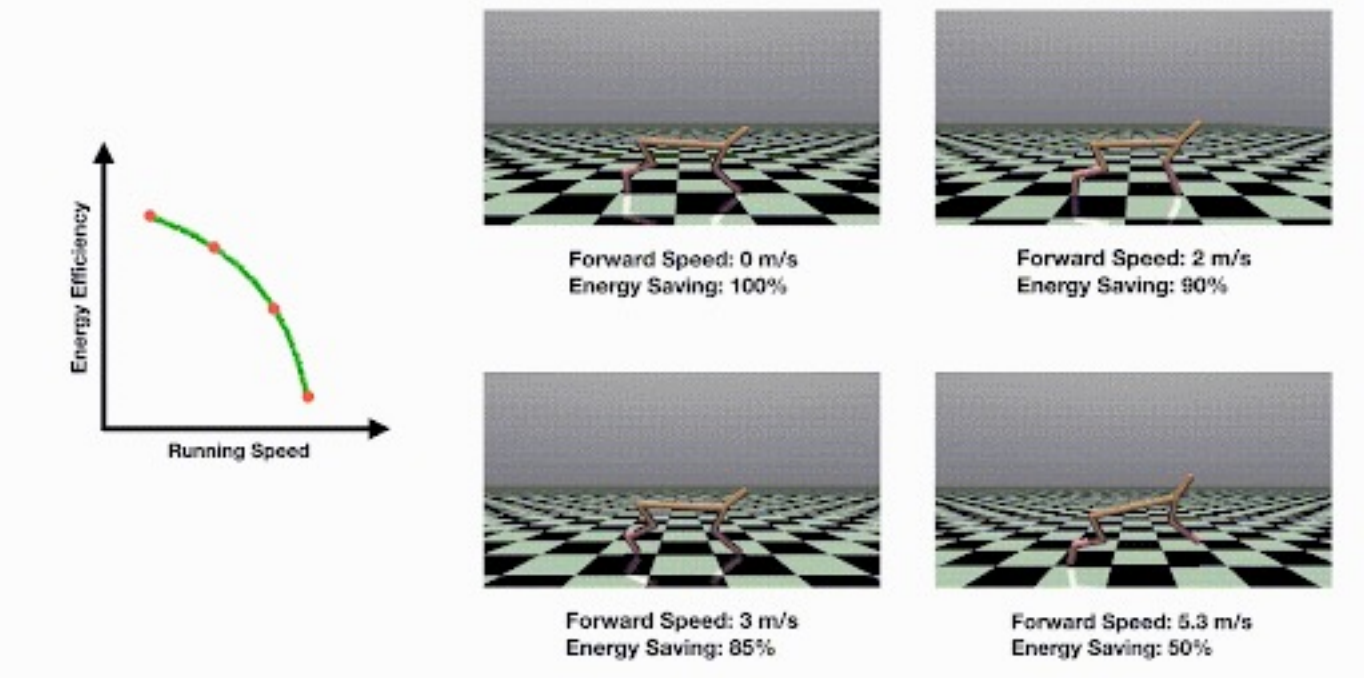
What are criteria in these problems? What is the set of alternatives? Why is there a conflict?

(C) Multi-objective ML/RL:

Search space: (hyper-)parameters of the RL agent

Criteria: running speed →max, energy efficiency → max

Constraints: Stability, Safety, Physical feasibility



Multi-Objective Graph Heuristic Search for Terrestrial Robot Design Jie Xu, Andrew Spielberg, Allan Zhao, Daniela Rus, Wojciech Matusik, *IEEE International Conference on Robotics and Automation (ICRA 2021)*

Multicriteria Optimization and Decision Analysis

- ***Definition: Multicriteria Decision Analysis (MCDA)*** assumes a finite number of alternatives, and their multiple criteria values are known at the beginning of the solution process.

  It provides methods to compare, evaluate, and rank solutions based on this information and how to elicit preferences.

- ***Definition: Multicriteria Optimization (or: Multicriteria Design, Multicriteria Mathematical Programming)*** assumes that solutions are implicitly given by a large search space and objective and constraint functions that can be used to evaluate points in this search space.

  It provides methods for searching large spaces for interesting solutions or sets of solutions.

Def.: Minimum, minimizer

Let $f(x)$ denote a function mapping from a space $\mathbb{S}$ to $\mathbb{R}$. Then

$$f^* = \min\{f(x) \mid x \in \mathbb{S}\}$$

is called the **minimum** of f.

All points $x \in \mathbb{S}$ with $f(x) = f^*$ are called **minimizers** of f. The set of minimizers is denoted with:

$$\arg\min_{x \in \mathbb{S}}(f(x))$$

Analogously, one can define **maximum** and **maximizer**.

Def.: Conflicting objective functions

Let $f_1 : \mathbb{S} \to \mathbb{R}$ and $f_2 : \mathbb{S} \to \mathbb{R}$ denote two objective functions to be minimized.

Then, these objective functions are said to be conflicting, if and only if

$$\arg\min_{x \in \mathbb{S}}(f_1(x)) \cap \arg\min_{x \in \mathbb{S}}(f_2(x)) = \emptyset.$$

This means is it is impossible to find a point that maximizes both functions simultaneously.

Can this be generalized to 3-D?

$$\cap_i \arg\min f_i(x) = \emptyset$$

Standard formulation of mathematical programming

Linear programming

Let $a_1, \ldots, a_d$, $b_{11}, ..., b_{n,d}$, $c_1, \ldots, c_n$ denote real valued constants and $x_1, \ldots, x_d$ real valued decision variables.

Then a problem of the form

$$\begin{aligned} a_1 x_1 + ... + a_d x_d &\rightarrow \text{min} \\ \text{subject to} & \\ b_{1,1} x_1 + \ldots + b_{1,d} x_d &\geq c_1 \\ &\vdots \\ b_{n,1} x_1 + \ldots + b_{n,d} x_d &\geq c_n \\ (x_1, \ldots, x_d) &\in \mathbb{R}^d \end{aligned}$$

is called a **linear programming problem**.

In matrix notation, it can be written in a compact form as

$$\min_{\vec{x} \in \mathbb{R}^d} \vec{a} \cdot \vec{x}, \quad \text{s.t. } B\vec{x} - \vec{c} \geq 0$$

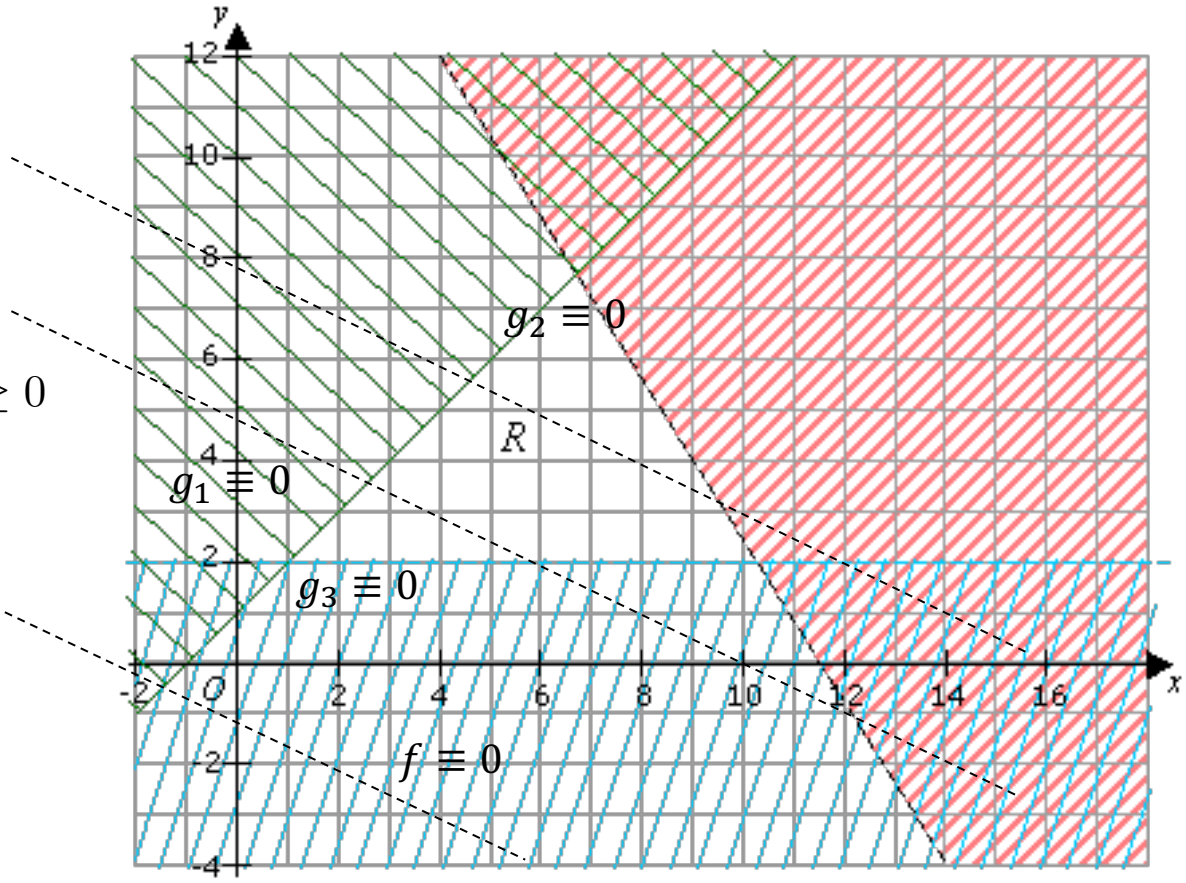
$$B \in \mathbb{R}^{n \times d}, \quad B_{ij} = b_{i,j}$$

Linear programming: Graphical solution in 2D

(http://www.onlinemathlearning.com/linear-programming-example.html)

$$\min_{x,y\in\mathbb{R}} \quad f(x,y) = 2y + x$$

$$\text{s.t.} \quad g_1(x,y) = x - y + 1 \geq 0$$
$$g_2(x,y) = 92 - 5y - 8x \geq 0$$
$$g_3(x,y) = y - 2 \geq 0$$



Auxillary computations:
For parallel iso-utility lines, draw (dashed) line 2y + x = 0 ⇔y=-x/2, indicate parallel lines
For constraint boundaries: y=x+1 (no transformation needed), 5y+8x≤92⇔y≤92/5-8/5x

Where is the maximizer? Which constraints are active?

Terminology: Constraints

Def.:Feasible point: A point $x \in \mathbb{S}$ that satisfies all constraints is called a **feasible point**. All other points in $\mathbb{S}$ are called **infeasible** points.

Def.:Feasible set: The set $\mathbb{F} = \{x \in \mathbb{S} | \ x \text{ feasible}\}$ is called the feasible set of $\mathbb{S}$.

Def.:Active or binding constraints: A feasible point $x \in \mathbb{S}$ the inequality constraints g_i that satisfy $g_i(x) = 0$ are called **binding** or **active** constraints.

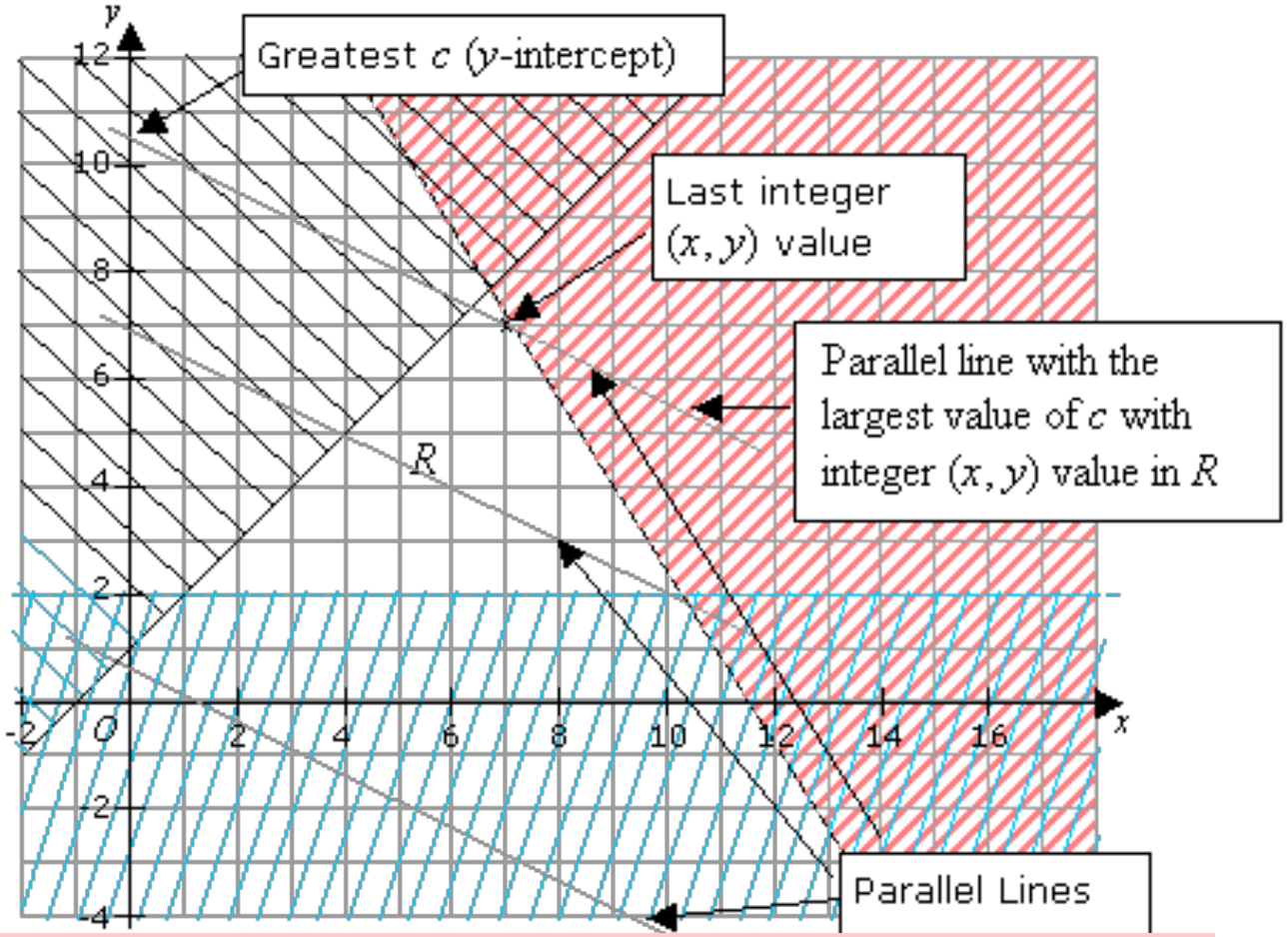
Integer Linear Programming (ILP)

$$\max_{x,y\in\mathbb{Z}} \quad 2y + x$$

$$\text{s.t.} \quad x - y + 1 \geq 0$$

$$92 - 5y - 8x \geq 0$$

$$y - 2 \geq 0$$



DEF: The *integrality gap* of an integer programming problem is defined as the difference $f^* - f_z^*$ where f^* is the minimizer of the continuous problem (=*continuous relaxation*) and f_z^* is the minimum of the integer programming problem.

For solvers of LP/MP problems: see for instance: Google OR Tools (interface to programming languages), LPSolve (LAPack, C/C++, online), pulp (python), Gurobi, CPLEX (IBM).

(Linear) relaxation and integrality gap

- The **relaxation** of an ILP problem is the problem in which integer variables, e.g., $b_i \in \{0,1\} \subset \mathbb{Z}$ are replaced by box-constrained continuous variables, e.g., $x_i \in [0,1]$.

- The solution of the relaxation problem provides a **lower bound** for the solution of the original IP problem (subject to minimization).

- The gap between the relaxation problem's minimum and the original ILP problem's minimum is called the **integrality gap**

(Linear) relaxation and integrality gap

- LPs can be solved efficiently for large number of variables, but ILPs are considered to require exponential time algorithms (under the *exponential time hypothesis*).

- Therefore, often ILPs are relaxed to LPs and solved, provided the integrality gap can be shown to be sufficiently small or even zero.

Mathematical ‘Programming’

The term 'Linear programming' was coined by George Dantzig

It refers to the task of finding an optimal program (schedule) for a planning problem and not to computer programming.

Linear programming originated in millitary but now is widely used in civil applications of economical and logistic planning.

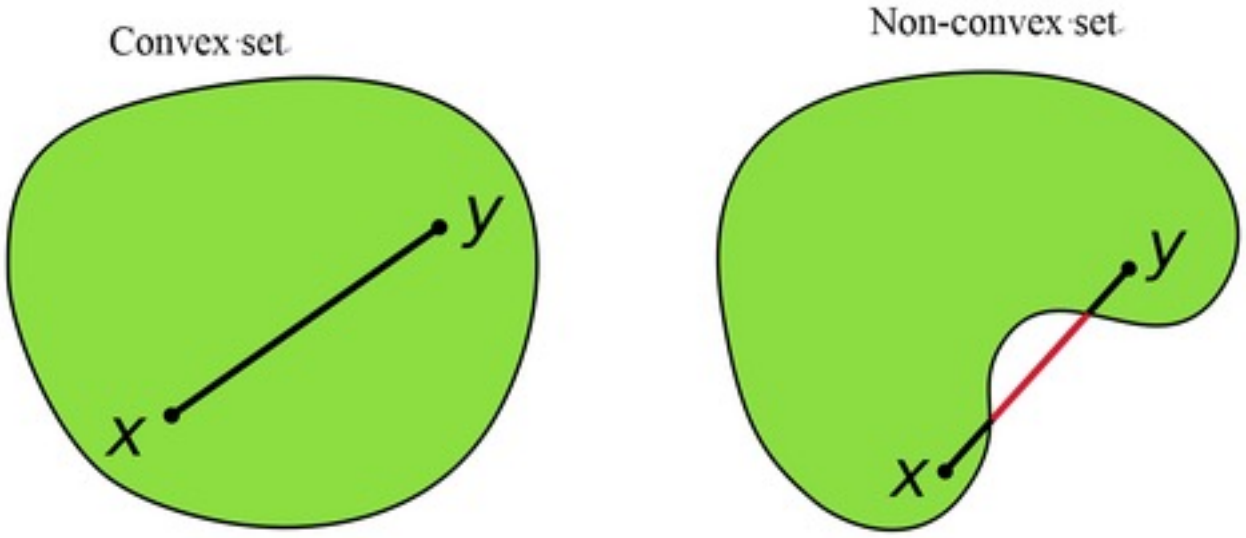
Today 'Linear programming problems' are often also called 'Linear programs'

More general problem definitions using
similar notation: 'Mathematical programming'

*George Dantzig,
US American Mathematician, 1914 – 2005

Convexity of feasible space in LP

- A subset of $S \subseteq \mathbb{R}^n$ is considered to be **convex**, if and only if for every two points $a \in S, b \in S$ the line segment $\ell(a, b)$ connecting a and b is fully contained in S



- We observe that the set of feasible solutions for a single linear constraint is a convex set (a halfspace). (proof: exercise)

Convexity of feasible space in LP

- **Theorem 1**: the intersection of two convex subsets S_1 and S_2 of $\mathbb{R}^n$ is also convex.

- Proof: ?

Convexity of feasible space in LP

- **Theorem 1**: the intersection of two convex subsets S_1 and S_2 of $\mathbb{R}^n$ is also convex.

- Proof: If we take two points a, b with $a \in S_1, b \in S_1$ and $a \in S_2$ and $b \in S_2$ and due to convexity of S_1 and S_2 also $\ell(a, b) \subseteq S_1$ and $\ell(a, b) \subseteq S_2$. Then we can conclude that $a \in S_1 \cap S_2, b \in S_1 \cap S_2$, and $\ell(a, b) \subseteq S_1 \cap S_2$.

Convexity of feasible space in LP

- Theorem 2: for an LP, the set of feasible solutions is a convex set.

- Proof (sketch): by induction. The first constraint has a convex feasible set. The intersection of the feasible set where the second and first constraint are feasible is also convex (Theorem 2), and so on…

Mathematical programs in standard form

Let $x_1, \ldots, x_d$ denote d decision variables (integer, or real valued). Let $c_1, \ldots, c_n$ and $b_1, \ldots, b_q$ denote some constants.

Then a mathematical programming problem (MPP) has the form:

$$f(x_1, \ldots, x_d) \rightarrow \min$$

subject to

$$\begin{array}{rcl} g_1(x_1, \ldots, x_d) & \geq & c_1 \\ & \vdots & \\ g_n(x_1, \ldots, x_d) & \geq & c_n \\ h_1(x_1, \ldots, x_d) & = & b_1 \\ & \vdots & \\ h_q(x_1, \ldots, x_d) & = & b_q \end{array}$$

Here $g_i(.) \geq c_i, \ i \in \{1, \ldots, n\}$ are inequality constraints, and $h_j(.) = b_j, \ j \in \{1, \ldots, q\}$ equality constraints.

Multiobjective Mathematical Program

Let $x_1, \ldots, x_d$ denote d, $c_1, \ldots, c_n$, and $b_1, \ldots, b_q$ be defined as previous. A multiobjective mathematical programming problem (MOP) has the form:

$$\begin{array}{lcl} f_1(x_1, \ldots, x_d) & \rightarrow & \min \\ & \vdots & \\ f_m(x_1, \ldots, x_d) & \rightarrow & \min \end{array}$$

The only difference

subject to

$$\begin{array}{lcl} g_1(x_1, \ldots, x_d) & \geq & c_1 \\ & \vdots & \\ g_n(x_1, \ldots, x_d) & \geq & c_n \\ h_1(x_1, \ldots, x_d) & = & b_1 \\ & \vdots & \\ h_q(x_1, \ldots, x_d) & = & b_q \end{array}$$

For $m > 1$ one can always add the term 'Multiobjective', e.g. Multiobjective LP, Multiobjective MIP, etc..

Classification: Mathematical Programming

	Search-space $\mathbb{S}$	Degree of nonlinearity
Linear Programming (LP)	$\mathbb{R}^d$	linear
Quadratic Programming (QP)*	$\mathbb{R}^d$	quadratic**
Nonlinear programming (NLP)	$\mathbb{R}^d$	nonlinear
Integer programming (IP)	$\mathbb{Z}^d$	arbitary
Integer linear programming (ILP)*	$\mathbb{Z}^d$	linear
Mixed Integer Linear Programming (MILP)	$\mathbb{R}^d \times \mathbb{Z}^r$	linear
Mixed Integer Nonlinear programming (MINLP)	$\mathbb{R}^d \times \mathbb{Z}^r$	nonlinear
Continuous unconstrained optimization: $\mathbb{S} = \mathbb{R}^n$, $n_g = 0$		nonlinear

*A QP is also an NLP; A ILP is also a IP.

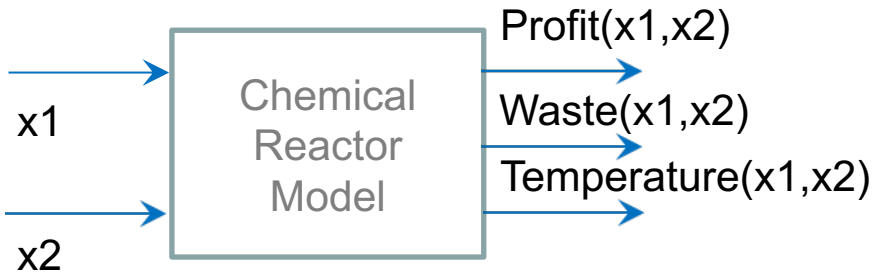
A quadratic function is of the form:

$$c_0 + b_1 x_1 + \ldots + b_d x_n + a_{1,1} x_1 x_1 + a_{1,2} x_1 x_2 + \ldots + a_{d,d} x_d x_d$$

If $A = (a_{ij}, i, j = 1 \ldots d)$ is positive definite (all eigenvalues positive) then the problem is termed **convex QP**. (can be solved efficiently)

Example 1: NLP
Mathematical Program for Reactor

Chemical Reactor



Profit to be maximized, while temperature and waste must not exceed certain thresholds.

How to formulate this as a mathematical program?

This is typically a black-box problem!

- Decision variables:
  Concentrations of educts:
  $x_1 = c_1 / \left[\frac{g}{l}\right], \quad x_2 = c_2 / \left[\frac{g}{l}\right]$
- Mathematical Program:

$$f(x_1, x_2) = \frac{Profit(x_1, x_2)}{[€]} \rightarrow Max$$

subject to

$$g_1(x_1, x_2) = \frac{Temp(x_1, x_2) - T_{max}}{[°C]} \leq 0$$

$$g_2(x_1, x_2) = \frac{Waste(x_1, x_2) - W_{max}}{\left[\frac{kg}{h}\right]} \leq 0$$

$$(x_1, x_2) \in [0,1] \times [0,1]$$

Example 2: Constrained 0/1 Knapsack Problem

Picture: © Michael Emmerich (former lecturer)

The total value of the items in the knapsack (in [$]) should be maximized, while its total weight (in [kg]) should not exceed MaxWeight. Here v_i is the value of item i in [$] and w_i is its weight in [kg]. $i = 1, \ldots, d$ are indices of the items.

$$\max_{x_1,\ldots,x_d \in \{0,1\}} \quad f(x_1, \ldots, x_d) = \sum_{i=1}^{d} v_i x_i$$

$$\text{s.t.} \quad g(x_1, \ldots, x_d) = \sum_{i=1}^{d} w_i x_i - \texttt{MaxWeight} \leq 0$$

What is the role of the binary variables here?

What type of mathematical programming problem is this?

Can this also be formulated as a quadratic programming problem?

Example 2: Knapsack Problem with Cardinality Constraint

The total value of the items in the knapsack (in [$]) should be maximized, while its total weight (in [kg]) should be below MaxN and at most MaxN items can be chosen.

$$\max_{x_1,\ldots,x_d \in \{0,1\}} \quad f(x_1,\ldots,x_d) = \sum_{i=1}^{d} v_i x_i$$

$$\text{s.t.} \quad g_1(x_1,\ldots,x_d) = \sum_{i=1}^{d} w_i x_i - \texttt{MaxWeight} \leq 0$$

$$g_2(x_1,\ldots,x_d) = \sum_{i=1}^{d} x_i - \texttt{MaxN} \leq 0$$

Example 2: Multiobjective 0/1 Knapsack Problem

The total value of the items in the knapsack (in [$]) should be maximized, while its total weight (in [kg]) should be minimized.

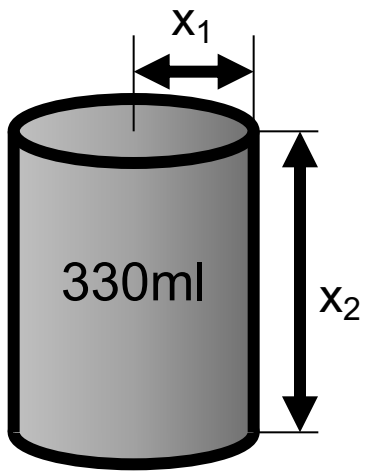
$$\max_{x_1,\ldots,x_d \in \{0,1\}} \quad f_1(x_1, \ldots, x_d) = \sum_{i=1}^{d} v_i x_i$$

$$\min_{x_1,\ldots,x_d \in \{0,1\}} \quad f_2(x_1, \ldots, x_d) = \sum_{i=1}^{d} w_i x_i$$

Example 3: Equality Constraint for Tin Problem

Minimize the area of the surface for a cylinder that contains V = 330 ml sparkling juice! $x_1 = radius/[cm^2], x_2 = height/[cm^2]$

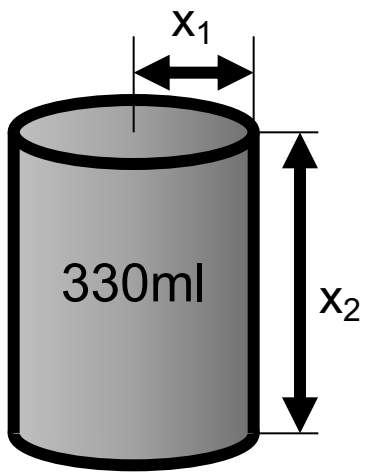
Formulate this problem as a mathematical programming problem?



Problem sketch

Example 3: Equality Constraint for Tin Problem

Minimize the area of the surface for a cylinder that contains V = 330 ml sparkling juice! $x_1 = radius/[cm^2], x_2 = height/[cm^2]$



Problem sketch

$$\min_{x_1, x_2 \in \mathbb{R}_{\geq 0}} \quad 2\pi x_1 x_2 + 2\pi x_1^2$$

$$\text{s.t.} \quad \pi x_2 x_1^2 - 330 = 0$$

This is a non-linear programming problem.
By substitution $x_1 x_2 =: x_3$ it can be transformed to a QP:

$$\min_{x_1, x_3 \in \mathbb{R}_{\geq 0}} \quad 2\pi x_3 + 2\pi x_1^2$$

$$\text{s.t.} \quad \pi x_3 x_1 - 330 = 0$$

Some interesting research questions: Find optimal shapes or given constraints on geometry.

For instance, Convex hull of N points with minimal surface and maximal volume; isoperimetric problem

Example 4: QP - Lasso and Ridge Regression

Given a regression data set/task: $\{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{N} \subseteq \mathbb{R}^d \times \mathbb{R}$

We want to fit a linear regressor to the data:

$$f_{\vec{\beta}}(\vec{x}) = \vec{x} \cdot \vec{\beta}$$

such that (1) we minimize the empirical risk (MSE in this case):

$$\frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \vec{x}^{(i)} \cdot \vec{\beta})^2$$

and (2) we restrict the "complexity" of the linear regressor (recall p-norm):

$$||\vec{\beta}||_1 = \sum_{i=1}^{d} |\beta_i| \leq t \qquad \text{Lasso}$$

$$||\vec{\beta}||_2^2 = \sum_{i=1}^{d} |\beta_i|^2 \leq t \qquad \text{Ridge}$$

Example 4: QP - Lasso and Ridge Regression

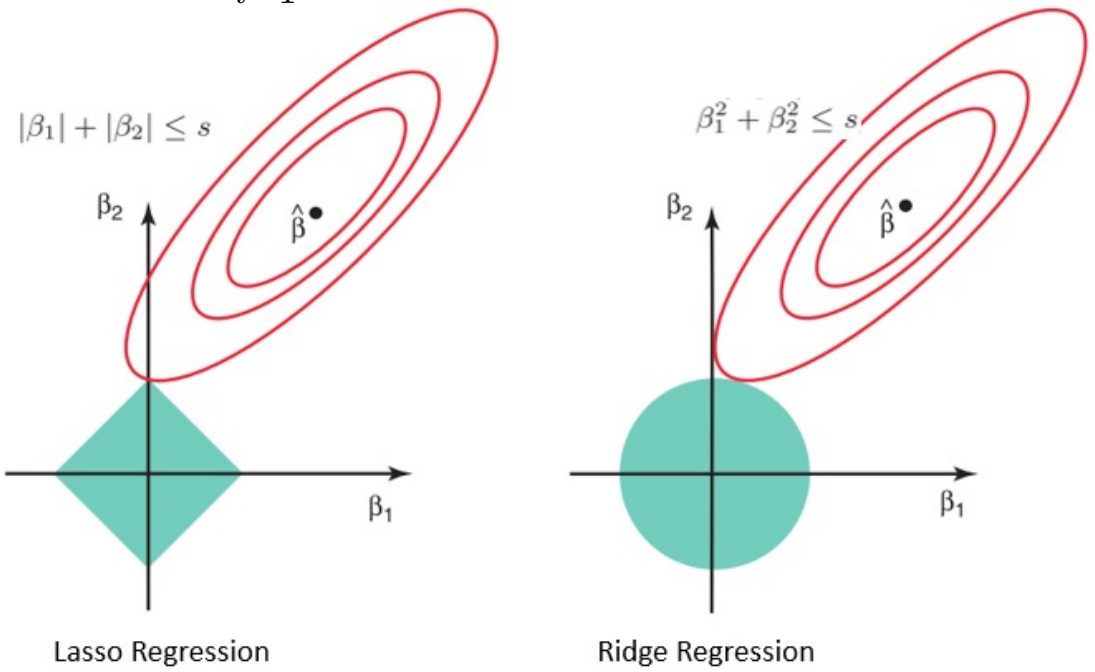
Formally, for Lasso, the optimization problem is:

$$\min_{\vec{\beta} \in \mathbb{R}^d} \quad \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \vec{x}^{(i)} \cdot \vec{\beta})^2$$

$$\text{s.t.} \quad ||\vec{\beta}||_1 = \sum_{i=1}^{d} |\beta_i| \leq t$$

Can you formulate it as a bi-objective problem?

Geometric interpretation:



Example 4: QP - Lasso and Ridge Regression

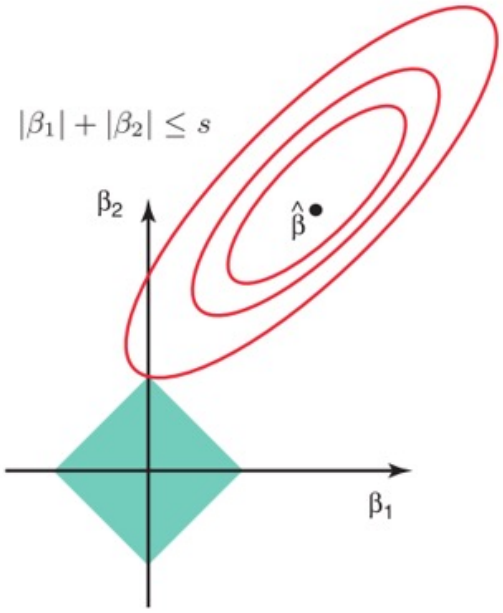
The Lasso problem is equivalent to the following **unconstrained** problem:

$$\min_{\vec{\beta}\in\mathbb{R}^d} \quad \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \vec{x}^{(i)}\cdot\vec{\beta})^2 \qquad \min_{\vec{\beta}\in\mathbb{R}^d} \quad \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \vec{x}^{(i)}\cdot\vec{\beta})^2 + \lambda||\vec{\beta}||_1, \quad \lambda\in\mathbb{R}_{>0}$$

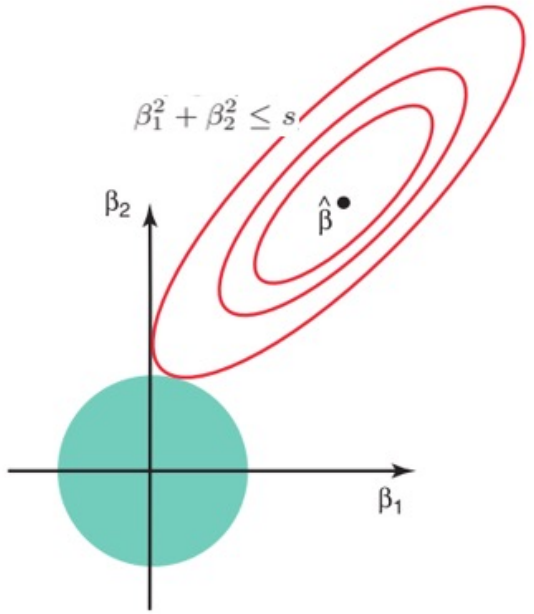
$$\text{s.t.} \quad ||\vec{\beta}||_1 = \sum_{i=1}^{d}|\beta_i| \leq t$$

Why the equivalence?

(We shall prove it formally when learning the optimality conditions)



Lasso Regression



Ridge Regression

Example 5: Mixed-Integer Problem – Placing Discs

Problem: placing as many discs of radius 10cm as possible on a square of side-length 1 meter? The unit discs must not overlap.

$$n = \lfloor 100 \times 100/(2\pi 10^2) \rfloor$$

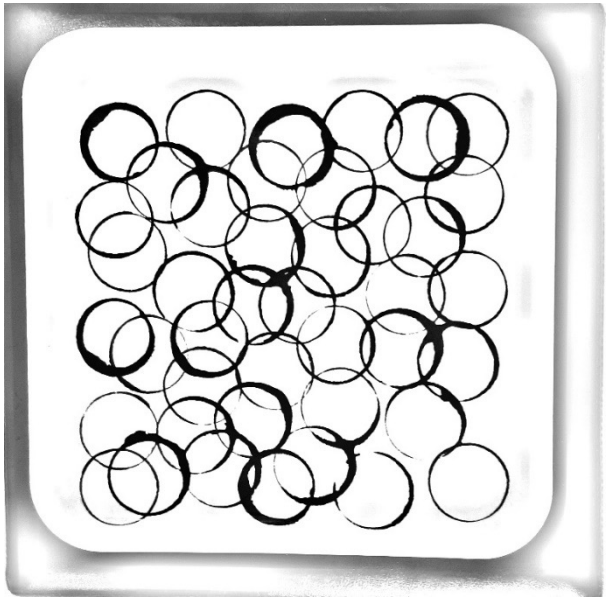
$$\max_{\substack{b_1,\ldots,b_n \in \{0,1\} \\ x_1,\ldots,x_n \in [10,90] \\ y_1,\ldots,y_n \in [10,90]}} \quad \sum_{i=1}^{n} b_i$$

$$\text{s.t.} \quad \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq 20 b_i b_j$$

$$i = 1, \ldots, n-1, j = i+1, \ldots, n$$

$b_i b_j$ = 1 if and only if both disc i and disc j are active

Hmm... But, the constraints are not linear in the binary variables



Unit disc problem: Infeasible design due to overlap.

Example 5: the Large Constant Technique

Alternative: Linear penalty with large constant L

$$\max_{\substack{b_1,\ldots,b_n\in\{0,1\}\\ x_1,\ldots,x_n\in[10,90]\\ y_1,\ldots,y_n\in[10,90]}} \quad \sum_{i=1}^{n} b_i$$

$$\text{s.t.} \quad \sqrt{(x_i-x_j)^2+(y_i-y_j)^2} \geq 20-(1-b_i)L-(1-b_j)L$$
$$i=1,\ldots,n-1, j=i+1,\ldots,n$$

Here L is a very large constant. What is the idea/advantage?

- L is sufficiently large constant, such that constraint is always satisfied when $b_i = 0$ or $b_j = 0$
- Large constant trick is preferable, as constraint gets linear

Example 6: Solving minimax problems as LPs?

The following problem is a special case of the **minimax** problem:

$$\min_{x_1,\ldots,x_d \in \mathbb{R}} \max\{f_1(x_1,\ldots,x_d),\ldots,f_k(x_1,\ldots,x_d)\}$$

where f_1 ... f_k are linear functions.

For instance, f_1 ... f_k can represent different types of costs of some real-world problem.

Can we re-formulate the above problem as a LP equivalently?

Example 6: Solving min/max problems as LPs?

Question: How to solve problems of the following kind with an LP solver:

$$\min_{x_1,\ldots,x_d \in \mathbb{R}} \max\{f_1(x_1,\ldots,x_d),\ldots,f_k(x_1,\ldots,x_d)\}$$

where f_1 ... f_k are linear functions.

Answer: Introduce additional **slack variable** z:

$$\begin{aligned} \min_{z \in \mathbb{R}} \quad & z \\ \text{s.t.} \quad & f_i(x_1,\ldots,x_d) \leq z, \quad i = 1,\ldots,k \end{aligned}$$

Example 7: A constraint that enforces integrality

How to transform linear problems with binary variables into quadratic continuous problems?

Let us assume the ILP problem

$$\min_{x_1,\ldots,x_d \in \{0,1\}} \quad \sum_{i=1}^{d} a_i x_i$$

$$\text{s.t.} \quad \sum_{i=1}^{d} w_i x_i - c \leq 0$$

How can we formulate it as a (continuous) QP problem?

Example 7: A constraint that enforces integrality

Let us assume the ILP problem

$$\min_{x_1,\ldots,x_d \in \{0,1\}} \quad \sum_{i=1}^{d} a_i x_i$$

$$\text{s.t.} \quad \sum_{i=1}^{d} w_i x_i - c \leq 0$$

How can we formulate it as a (continuous) QP problem?

Answer:

$$\min_{x_1,\ldots,x_d \in \mathbb{R}} \quad \sum_{i=1}^{d} a_i x_i$$

$$\text{s.t.} \quad \sum_{i=1}^{d} w_i x_i - c \leq 0$$

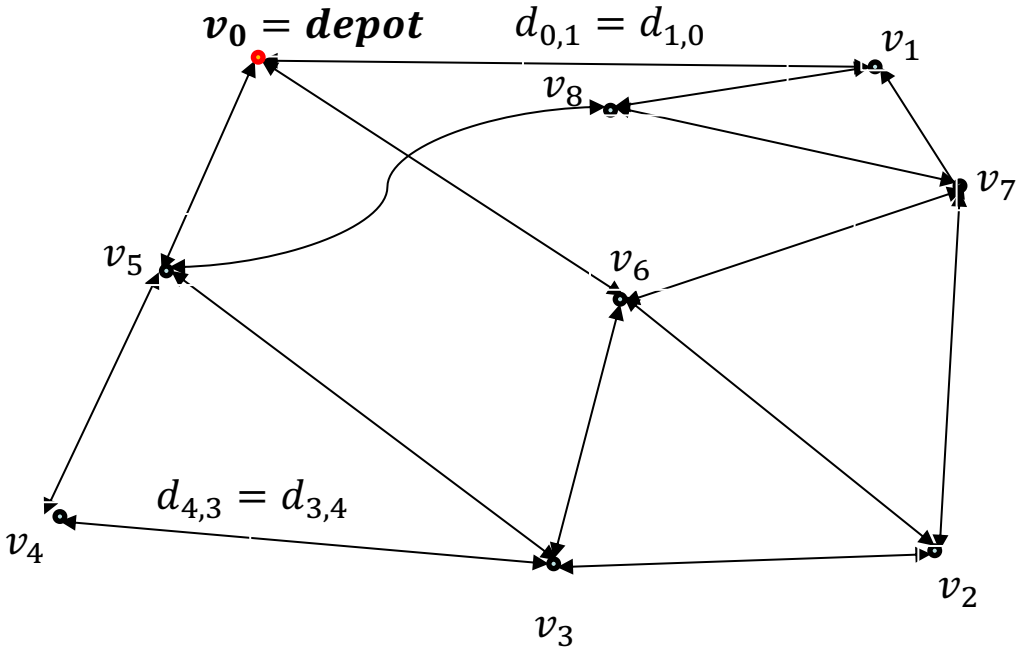
$$(1 - x_i) x_i = 0, \quad i = 1, \ldots, d$$

Note, that this can yield a non-convex optimization problem. Non-convex quadratic optimization, just as ILP, belong to the **NP hard problems**, which are considered to require exponential time for their solution in the size of the input.

Example 8: Traveling Salesperson Problem

Problem: Given a distance matrix $d_{i,j}$, provide a tour

- starting from node 0 (=depot)
- visiting each node exactly once
- returns to node 0



This can be easily described using graph theoretical entities, such as paths, nodes, and edges, but **how to model this in terms of discrete (binary) variables** and (constraint) functions that depend on them?

Transforming Quadratic Binary Programming to Linear Binary Programming Problems

The problem is to determine the value of a knapsack which is the sum of the loaded items.

When items x_i and x_j are together chosen there is a synergetic benefit Δv_{ij} added:

$$\text{s.t.} \quad \sum_{i=1}^{n} \sum_{j=i+1}^{n} \Delta v_{ij} x_i x_j + \sum_{i=1}^{n} v_i x_i \rightarrow max$$

$$\sum_{i=1}^{n} w_j x_j \leq C_{max}, j = 1, \dots, n$$

$$x_i \in \{0,1\}, i = 1, \dots, n$$

How to formulate this problem as an integer linear program (ILP)?

Answer …

Introduce additional variables $q_{ij} \in \{0,1\}$, $i < j$ and constraints:

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \Delta v_{ij} q_{ij} + \sum_{i=1}^{n} v_i x_i \rightarrow max$$

$$\sum_{i=1}^{n} w_j x_j \leq C_{max}, j = 1, \dots, n$$

$$q_{ij} \leq x_i, \qquad q_{ij} \leq x_j,$$

$$q_{ij} \geq x_i + x_j - 1$$

$$q_{ij} \in \{0,1\}$$

$$i = 1, \dots, n, j = i + 1, \dots, n$$

Observe, that $q_{ij} = 1$ if $x_i = 1$ and $x_j = 1$, and $q_{ij} = 0$, otherwise.

How many variables, and constraints does the ILP have?

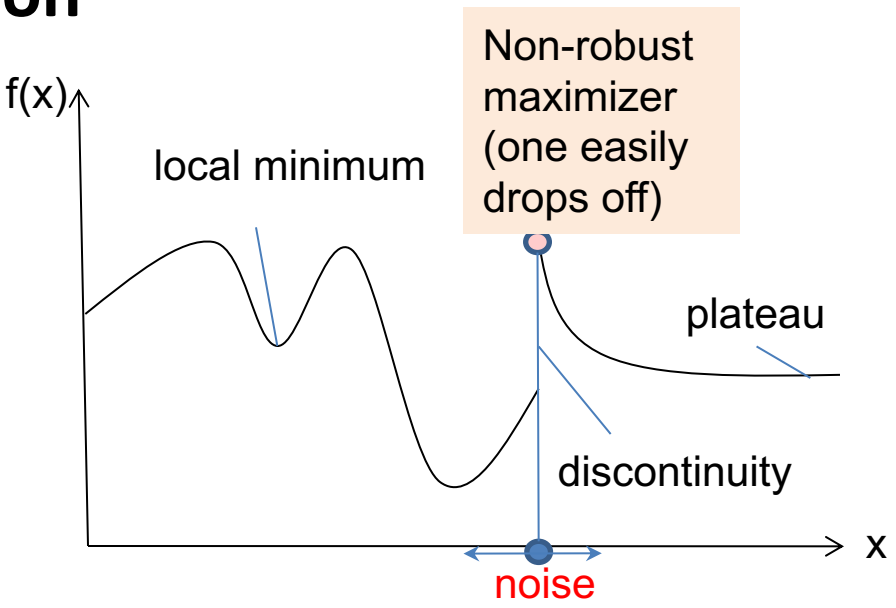
Complexity of optimization problems

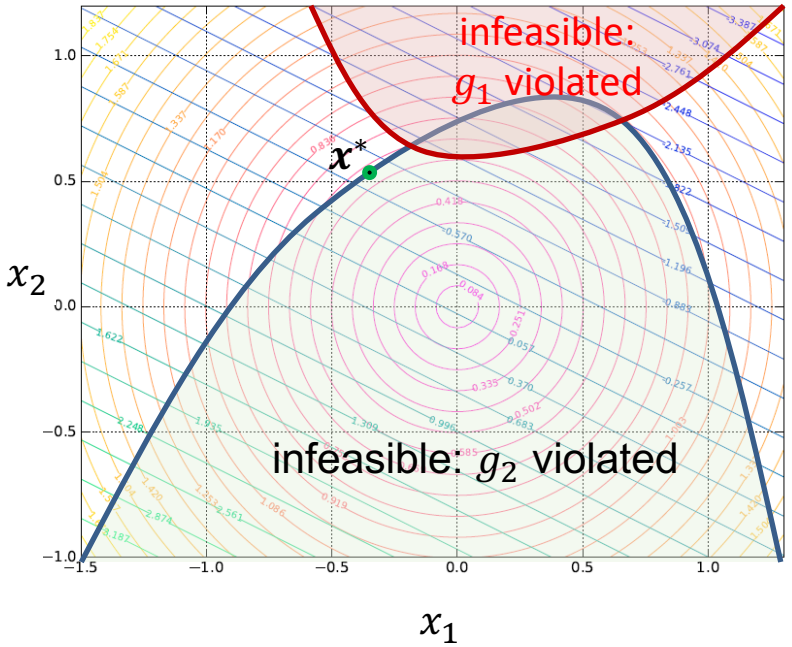
Complexity of optimization problems

CONTINUOUS OPTIMIZATION

Difficulties in Nonlinear Programming and Continuous Unconstrained Optimization

1. Multimodal functions (many local optima)
2. Plateaus and discontinuities
3. Non-differentiability
4. Nonlinear active boundaries of restriction functions
5. Disconnected feasible subspaces.
6. High dimensionality
7. Noise/Robustness





Black-box Optimization & Information Based Complexity

Assume the objective function f is a black box function of some class C that can be evaluated for all points $\mathbf{x} \in [0,1]^d$.

A black box algorithm a sequentially generates and evaluates the points $\mathbf{x}_1, \ldots, \mathbf{x}_t$, and appoints a current best solution $\mathbf{x}_t^*$. For some t it might provide a bound $f(\mathbf{x}_t^*) - f^* \leq \delta$

Let $T_a(f, \delta)$ denote the minimal number of iterations some algorithm a requires to guarantee an accuracy of δ for some function $f \in C$.

Let A denote the set of all algorithms.

Now the information based complexity IBC of f reads

$$\mathrm{IBC}(C, \delta) = \inf_{a \in A} \left(\sup_{f \in C} (T_a(f, \delta)) \right)$$

Fundamental bounds in continuous optimization

Needle in haystack function (not solvable, if black-box): $f(\mathbf{x}) = 0$ for $\mathbf{x} = \mathbf{x}^*$ and $f(\vec{x}) = 1$ elsewhere.

For Lipschitz functions, a small change of the input causes at most a small change of the output. Formally, we can define a Lipschitz constant: $k, 0 < k < \infty$

$$\forall \vec{x}, \vec{x}' \in [0,1]^d, |f(\vec{x}) - f(\vec{x}')| \leq k||\vec{x} - \vec{x}'||, k < \infty$$

If we know a Lipschitz constant and search over a finite domain, e.g. $\mathbf{x} \in [0,1]^d$ in the worst case the time for searching an optimum requires $n \sim (\frac{1}{\epsilon})^d$ function evaluations for finding a solution $\mathbf{x}$ with $|f(\mathbf{x}) - f^*| \leq \epsilon$
(see curse of dimension, reader)

Complexity of Global Black-box Optimization

Problem: $\min_{\vec{x} \in [0,1]^d} f(\vec{x})$

Problem class: Lipschitz functions

$$\forall \vec{x}, \vec{x}' \in [0,1]^d, |f(\vec{x}) - f(\vec{x}')| \leq k||\vec{x} - \vec{x}'||, k < \infty$$

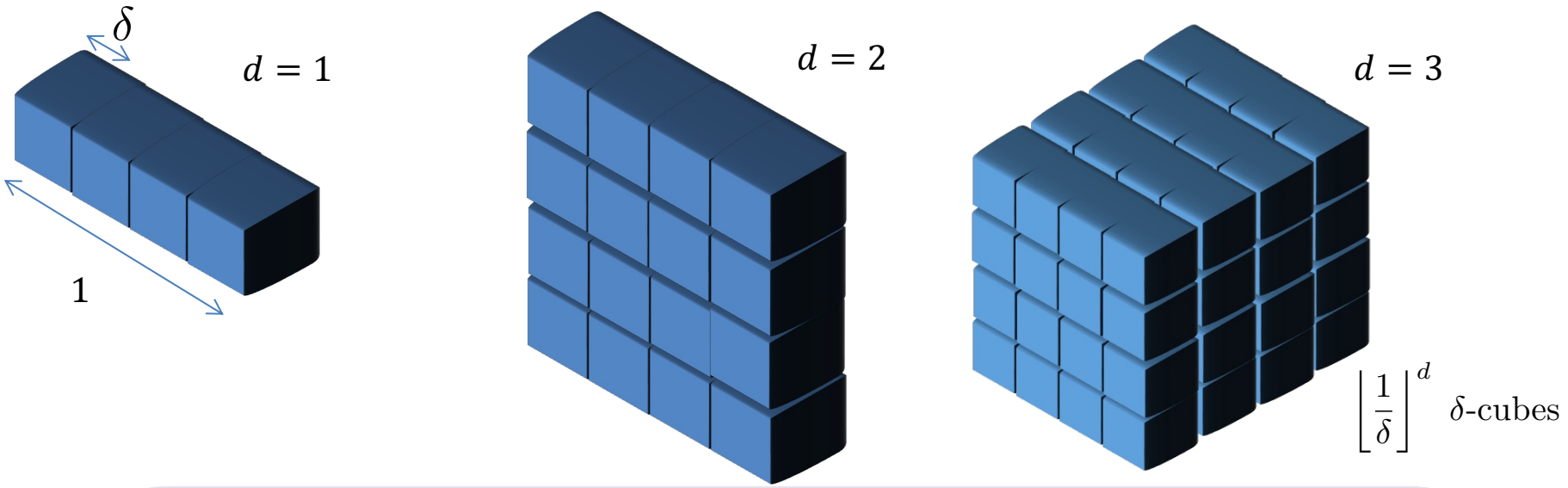
Zero-order oracle: $f(\vec{x})$

Task: find $\vec{x}$: $f(\vec{x}) - f^* \leq \varepsilon$

Theorem: $n(\varepsilon) \geq \left(\frac{k}{2\varepsilon}\right)^d$

Proof: ?

Complexity of Global Black-box Optimization



Theorem: $n(\varepsilon) \geq \left(\frac{k}{2\varepsilon}\right)^d$

- Evenly divide the unit cube into smaller cubes with side-length δ
- One δ-cube must contain the optimum
- The best strategy is to evaluate the center of each δ-cube
- For the δ-cube containing the optimum, we know $|f(\vec{x}_c) - f^*| \leq k\sqrt{\sum_{i=1}^{d}\left(\frac{1}{2}\delta\right)^2} = k\sqrt{\frac{d}{4}}\delta$
- We need to evalute $\lfloor 1/\delta \rfloor^d$ centers
- Let $k\sqrt{\frac{d}{4}}\delta \leq \varepsilon$ we have: $\left(\frac{1}{\delta}\right)^d \geq \left(k\frac{\sqrt{(d)}}{2\varepsilon}\right)^d \in \Omega\left(\left(\frac{k}{2\varepsilon}\right)^d\right)$

Complexity of optimization problems

COMBINATORIAL OPTIMIZATION

Decision version of optimization problem

A decision problem is a problem where for a question on a given input a yes/no answer needs to be computed.

A decision version of an unconstrained optimization problem is given by the problem:

Does there exist $\mathbf{x} \in \{0,1\}^d$ such that $f(\mathbf{x}) < \tau$?

The time complexity of an optimization problem is bounded by the time complexity of its decision version.

If an efficient algorithm for the decision problem exists, then binary search can be used for the (approximation of a) solution to the optimization problem.

In many cases, already the decision version of an optimization problem is difficult to solve.

NP hard problems

- A decision problem is in the *complexity class P* iff. it can be solved on a Turing machine (or a procedural language like C) in polynomial time.
- A decision problem is in the *complexity class NP* if it can be solved by a non-deterministic Turing Machine in polynomial time.
- Alternative definition NP: every input instance can be decided in polynomial time, whether or not it provides a witness (certificate) for the decision problem to evaluate true.
- A decision problem is *NP complete*, if and only if (1) every problem in NP can be reduced to this problem, (2) the decision problem is in P.
- A problem is NP hard if a NP complete problem can be reduced to it in polynominal time. NP hard problems are not necessarily in P.

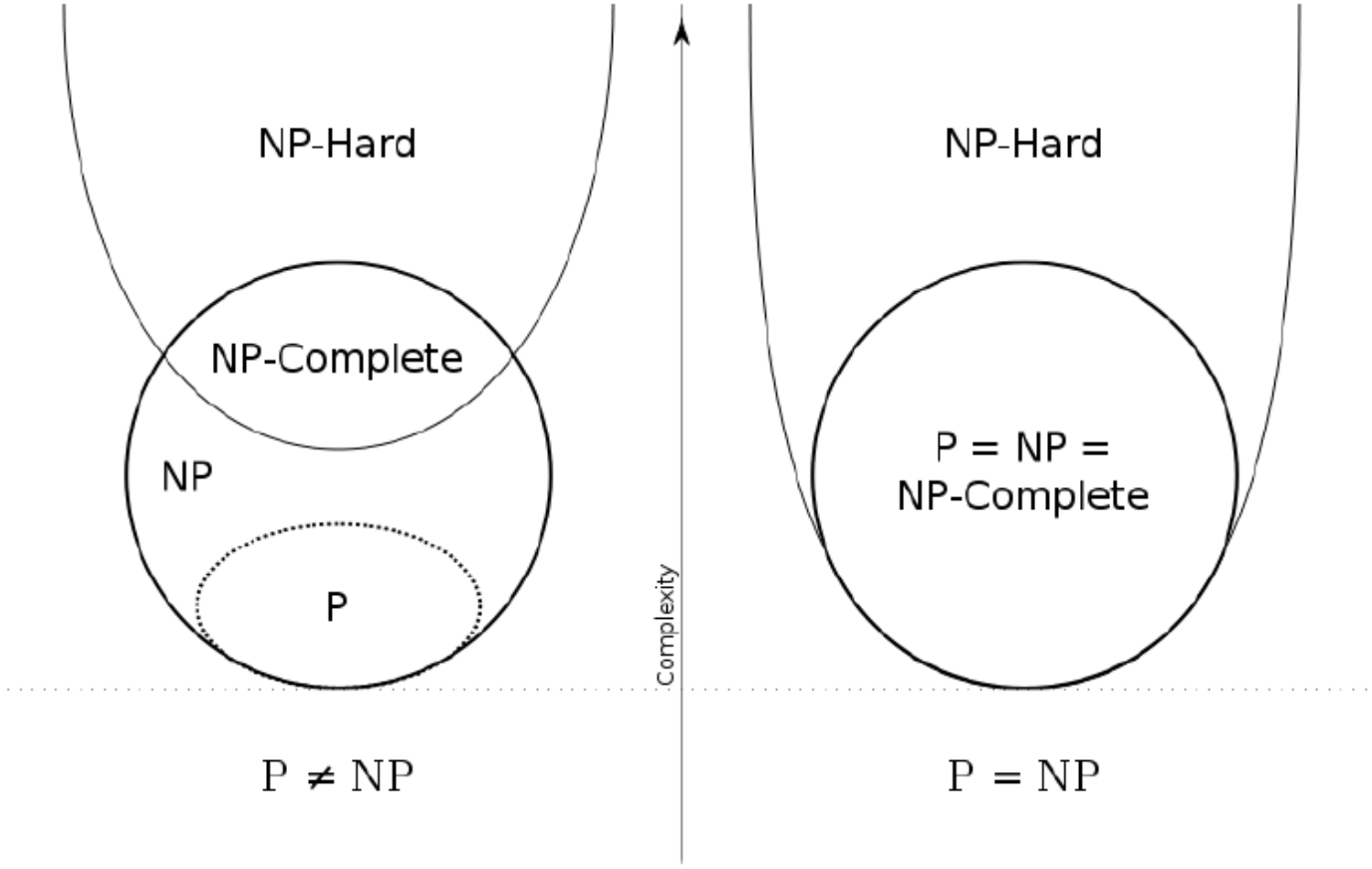
"I can't find an efficient algorithm, but neither can all these famous people."

'Famous' unsolved problem:

P = NP?

(c) https://ac.tuwien.ac.at/people/szeider/cartoon/... cf. @vardi
How we lost the women in computing (based on Garey and Johnson's classic 1979) https://cacm.acm.org/magazines/2018/5/227192-how-we-lost-the-women-in-computing/fulltext... #womenintech #STEM #computerscience

NP, NP-Complete, NP-Hard



Exponential time hypothesis:
Does there exist an algorithm that solves 3-CNF satisfyability in $O(2^{o(n)})$ time complexity.

Source: Wikipedia

2007-11-01 21:30 Behnam 1052×744×0 (9751 bytes) *Venn diagram for P, NP, NP-Complete, and NP-Hard problems, creative commons licence, https://creativecommons.org/licenses/by-sa/3.0/deed.en*

Difficulties in solving mathematical programming problems

Problem	Time Complexity	Method
LP, convex QP	polynomial	interior point
ILP, IP, NLP	exponential?	branch& bound

Karmarkar, N. (1984, December). A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing* (pp. 302-311). ACM.

Monteiro, R. D., & Adler, I. (1989). Interior path following primal-dual algorithms. Part II: Convex quadratic programming. *Mathematical Programming*, *44*(1-3), 43-66.

Androulakis, Ioannis P., Costas D. Maranas, and Christodoulos A. Floudas. "αBB: A global optimization method for general constrained nonconvex problems." *Journal of Global Optimization* 7.4 (1995): 337-363.

ILP is NP hard: Satisfiability to ILP reduction

- We can formulate a logical constraint with AND, OR, and NOT in Boolean logic as an ILP problem.
- Let b_i denote Boolean variables, with $b_i \in \{0,1\}, i = 1, \dots, \mathrm{n}$.
- Here, as usual, we use convention $1 = true, 0 = false$
- Now we can implement Boolean operators as linear constraints

$$x_1 + x_2 \geq 2 \sim x_1 \text{ AND } x_2$$
$$x_1 + x_2 \geq 1 \sim x_1 \text{ OR } x_2$$
$$1 - x_1 \sim \text{NOT } x_2$$
$$-x_1 - x_2 \geq 1 \sim x_1 \text{ NAND } x_2$$

- In general satisfyability of Boolean expressions is a NP complete combinatorial problem and up to date only exponential time algorithms are known.
- We can formulate the minimization problem:

$$\min s, \ \text{s.t. } g_{\mathrm{i}}(\boldsymbol{x}) + \mathrm{s} \geq c_i, s \in \{0,1\}, \boldsymbol{x} \in \{0,1\}^n, i = 1, \dots, q$$

- The variable s is the **slackness**. If s is zero in its minimum, the formula expressed by the constraints $g_i$is satisfyable
- Hence, also the ILP is NP-hard.

Subsumption and Complexity of White Box Mathematical Programming

Mixed Integer Nonlinear Programming (MINLP)

subsumes

Nonlinear Programming	Integer Programming
subsumes	*subsumes*
Convex Quadratic Programming	Binary Quadratic Programming
subsumes	*can be reformulated as*
Linear Programming	Integer Linear Programming (NP hard)
Continuous	Discrete

Remarks:

Efficient polynomial time solver available

NP hard but large scale solvers available and work well in many cases

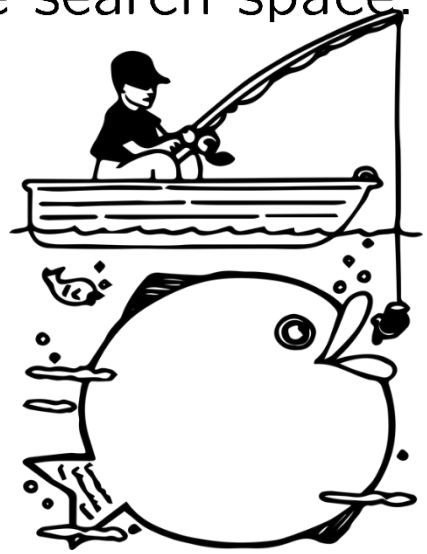
Reformulation as ILP requires $O(n^2)$ additional variables

Search heuristics, Branch and Bound

Search heuristics are algorithms that search for good solutions based on some clever ideas. Works well in practice but cannot guarantee to find the best solution.

Branch & bound methods compute & update upper and lower bounds for function values in subspaces. Based on them they prune the search space.

Search space and problem:
"What is the biggest fish in a very large lake"



Heuristic:
"I used some smart techniques to catch a big fish, but don't know if there is a bigger one?" (https://pixabay.com/illustrations/fishing-fisherman-big-fish-boar-4712416/)

Branch&Bound:
"It's certainly not found in this part"

Summary: Take home messages (1)

1. Modeling, simulation, and optimization are essential tools in systems analysis; A simple black box scheme can be used to capture their definition.
2. In operations research, optimization problems are classified in the form of mathemathical programming problems
3. The most simple mathematical programming task is linear programming
4. The classification scheme refers to the type of variables (e.g. binary, integer) and functions (e.g., quadratic, linear, nonlinear)
5. Standard solvers are available to solve mathematical programming problems; but problems might be difficult;

Summary: Take home messages (2)

7. Difficulties in continuous optimization arise due to local optima, plateaus, discontinuities and constraint boundaries.
8. In black box optimization: curse of dimensionality makes it difficult to guarantee optimal solution, even for Lipschitz continuous functions
9. In combinatorial optimization decision versions of many multiobjective optimization problems are NP-hard
10. Heuristics can find improvements, but often do not guarantee to find the optimum
11. Branch and bound can use relaxation and heuristics efficiently to prune combinatorial search space; but in worst case it still has exponential time complexity