

Unit 1: Mathematical Programming Formulation and Complexity

Learning goals – Unit 2

- I. What is the subject area of multiobjective decision analysis and multiobjective optimization; How does it relate to the more general field of systems analysis and other disciplines?
- 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 can their solution be difficult?

Motivation: Some Multicriteria Problems

(A) Select best travel destination from a catalogue:

Search space: Catalogue

Criteria: Sun \rightarrow max, DistanceToBeach \rightarrow min, and Travel Distance \rightarrow min

Constraints: Budget, Safety

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

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

Criteria: Effectivity \rightarrow max, SideEffects \rightarrow min, Cost \rightarrow min

Constraints: Stability, Solubility in blood, non-toxic

(C) How to control industrial processes:

Search space: Set of control parameters for each point in time

Criteria: Profitability \rightarrow max, Emissions \rightarrow min

Constraints: Stability, Safety, Physical feasibility

Other examples: SPAM classifiers, train schedules, computer hardware, soccer

What are criteria in these problems? What is the set of alternatives?

Why is there a conflict?

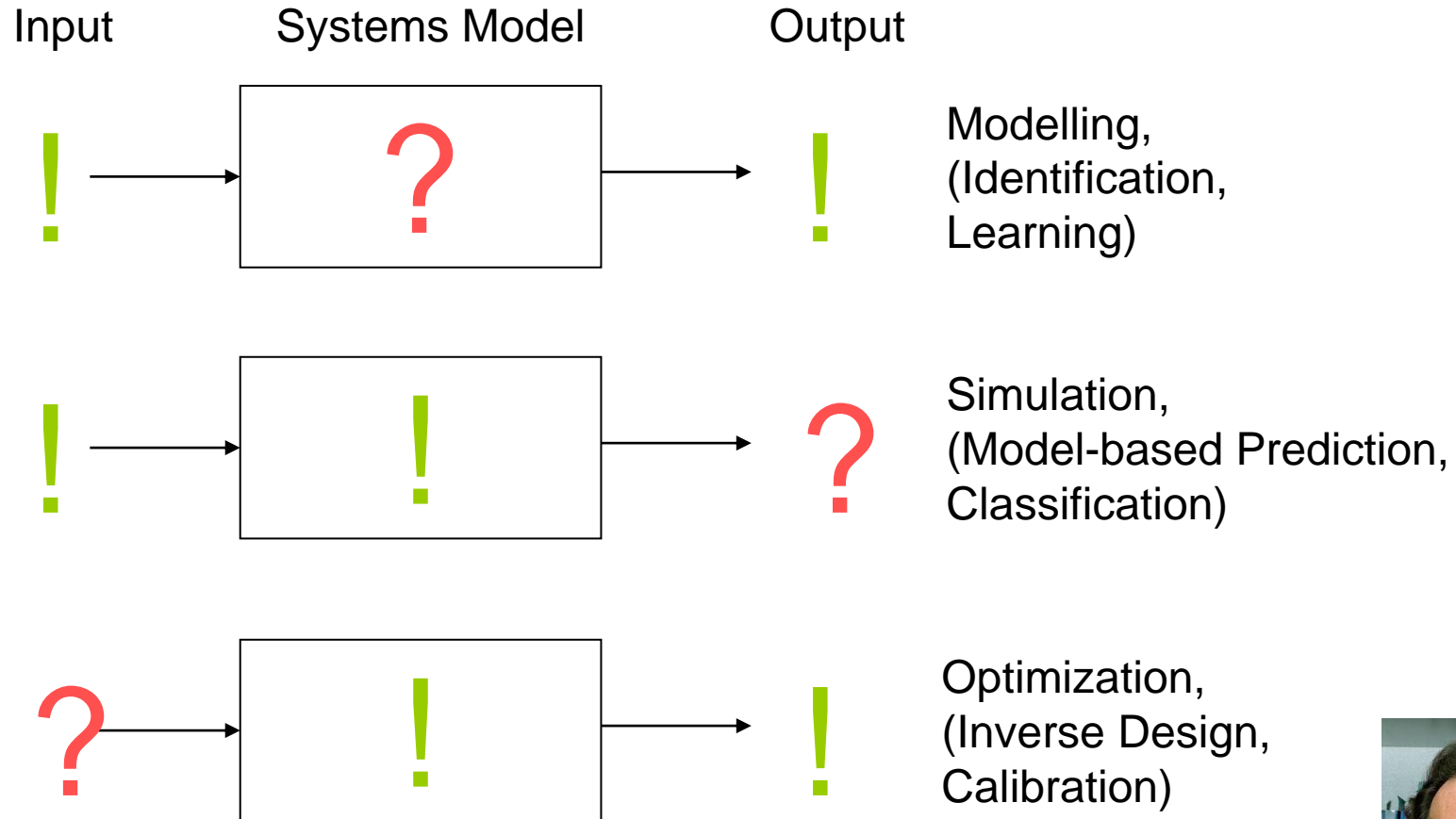
Multicriteria Optimization and Decision Analysis

- **Definition: Multicriteria Decision Analysis (MCDA)** assumes a finite number of alternatives and their multiple criteria value are known in the beginning of the solution process.
- It provides methods to compare, evaluate, and rank solutions based on this information, and how to elicitate 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 search large spaces for interesting solutions or sets of solutions.

Systems Analysis View of Optimization

Optimization in Systems Analysis



Source: Hans-Paul Schwefel:
Technische Optimierung,
Lecture Notes, 1991
((c) picture: personal communication)



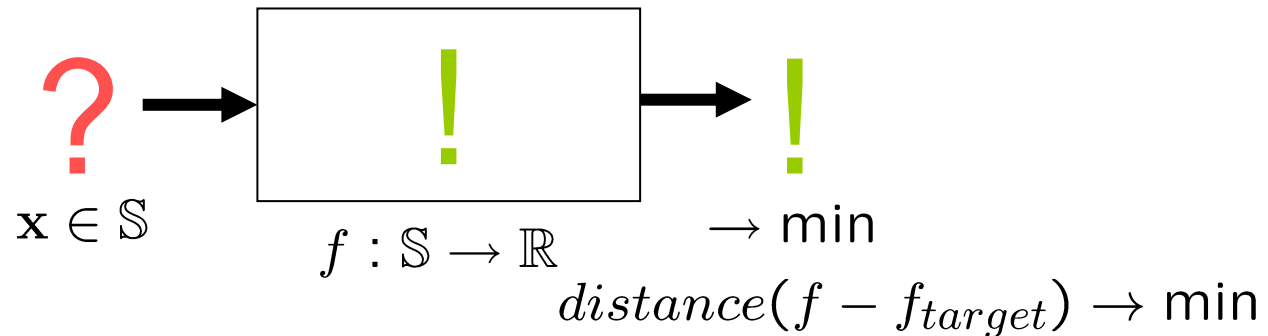
Systems Model of Optimization Task

$$f(\mathbf{x}) \rightarrow \min!, \quad \mathbf{x} \in \mathbb{S}$$

\mathbf{x} : Decision variables

\mathbb{S} : Search space (feasible solutions)

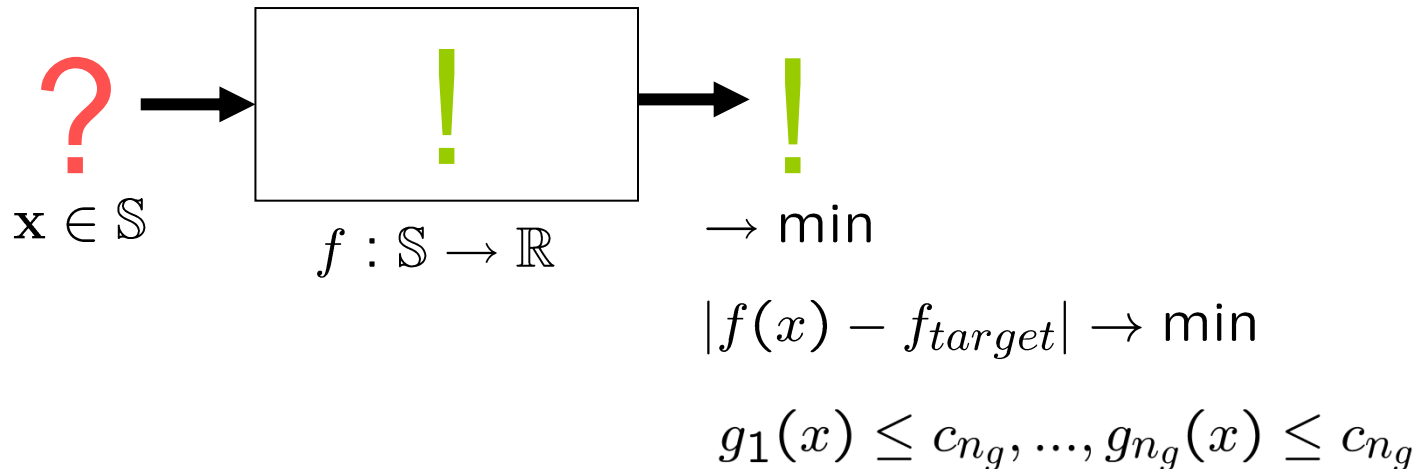
$f : \mathbb{S} \rightarrow \mathbb{R}$: Objective function



Constraints and restrictions

Often, not all solutions in \mathbb{S} are feasible (e.g. quality or budget constraints).

Feasibility can be comfortably handled by means of constraint functions.



Multi-objective optimization task

$$f_1(\mathbf{x}) \rightarrow \min, \dots, f_{n_f}(\mathbf{x}) \rightarrow \min \quad \mathbf{x} \in \mathbb{S}$$

\mathbf{x} : Decision variables

\mathbb{S} : Search space (feasible solutions)

$\mathbf{f} : \mathbb{S} \rightarrow \mathbb{R}^{n_f}$: Objective function



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 $f_1 : \mathbb{S} \rightarrow \mathbb{R}$ and $f_2 : \mathbb{S} \rightarrow \mathbb{R}$ 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 it is impossible to find a point that maximizes both functions simultaneously.

Can this be generalized to 3-D?

Standard formulation of mathematical programming

Linear programming

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

Then a problem of the form

$$\begin{aligned} a_1x_1 + \dots + a_dx_d &\rightarrow \min \\ &\text{subject to} \\ b_{1,1}x_1 + \dots + b_{1,d}x_d &\geq c_1 \\ &\vdots \\ b_{n,1}x_1 + \dots + b_{n,d}x_d &\geq c_n \\ (x_1, \dots, x_d) &\in \mathbb{R}^d \end{aligned}$$

is called a **linear programming problem**.

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

$$\mathbf{a} \cdot \mathbf{x} \rightarrow \min, \quad s.t. \quad \mathbf{B}\mathbf{x} \geq \mathbf{c}$$

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

s.t.

$$y \leq x + 1,$$

$$5y + 8x \leq 92$$

$$y \geq 2$$

$$f(x, y) = 2y + x$$

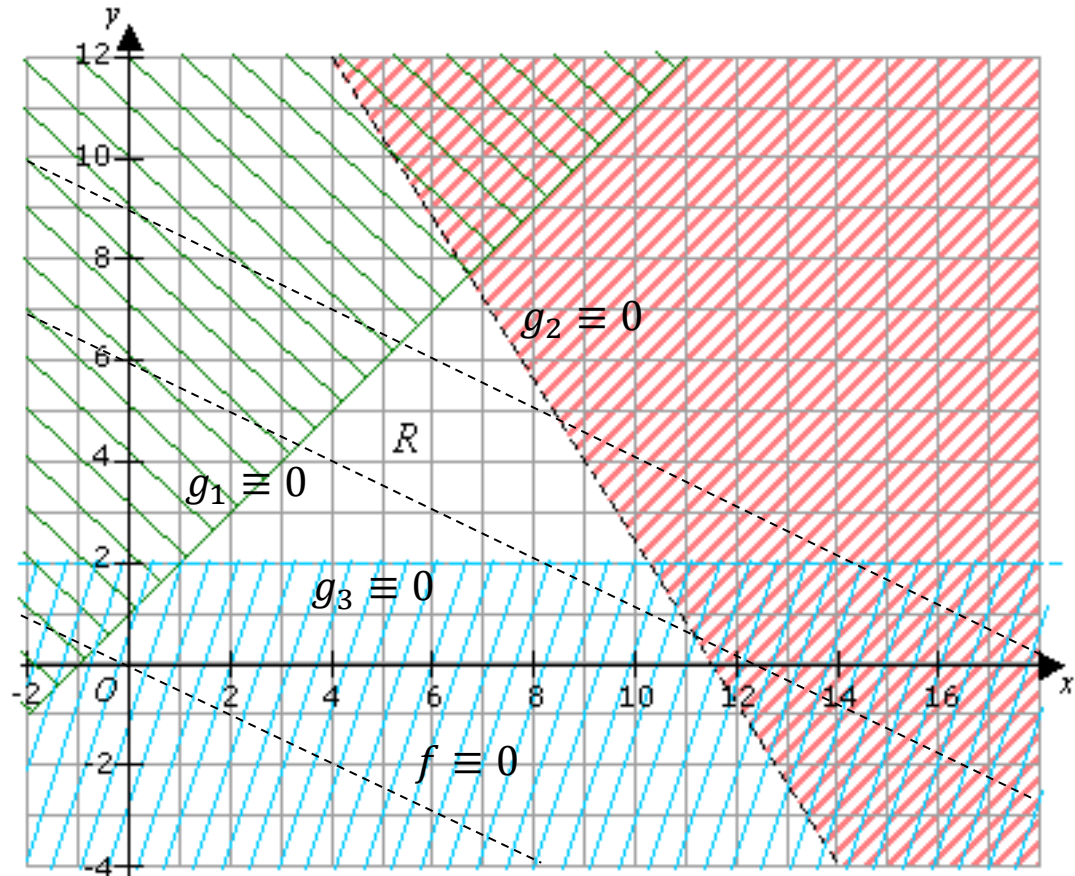
subject to

$$g_1(x, y) = x + 1 - y \geq 0$$

$$g_2(x, y) = 92 - 5y - 8x \geq 0$$

$$g_3(x, y) = y - 2 \geq 0$$

$$x, y \in \mathbb{R}$$



For parallel iso-utility lines, draw (dashed) line $2y + x = 0 \Leftrightarrow y = -x/2$, indicate parallel lines

For constraint boundaries: $y=x+1$ (no transformation needed), $5y+8x \leq 92 \Leftrightarrow y \leq 92/5 - 8/5x$

Where is the maximizer? Which constraints are active?

Linear integer programming

$$2y + x \rightarrow \max$$

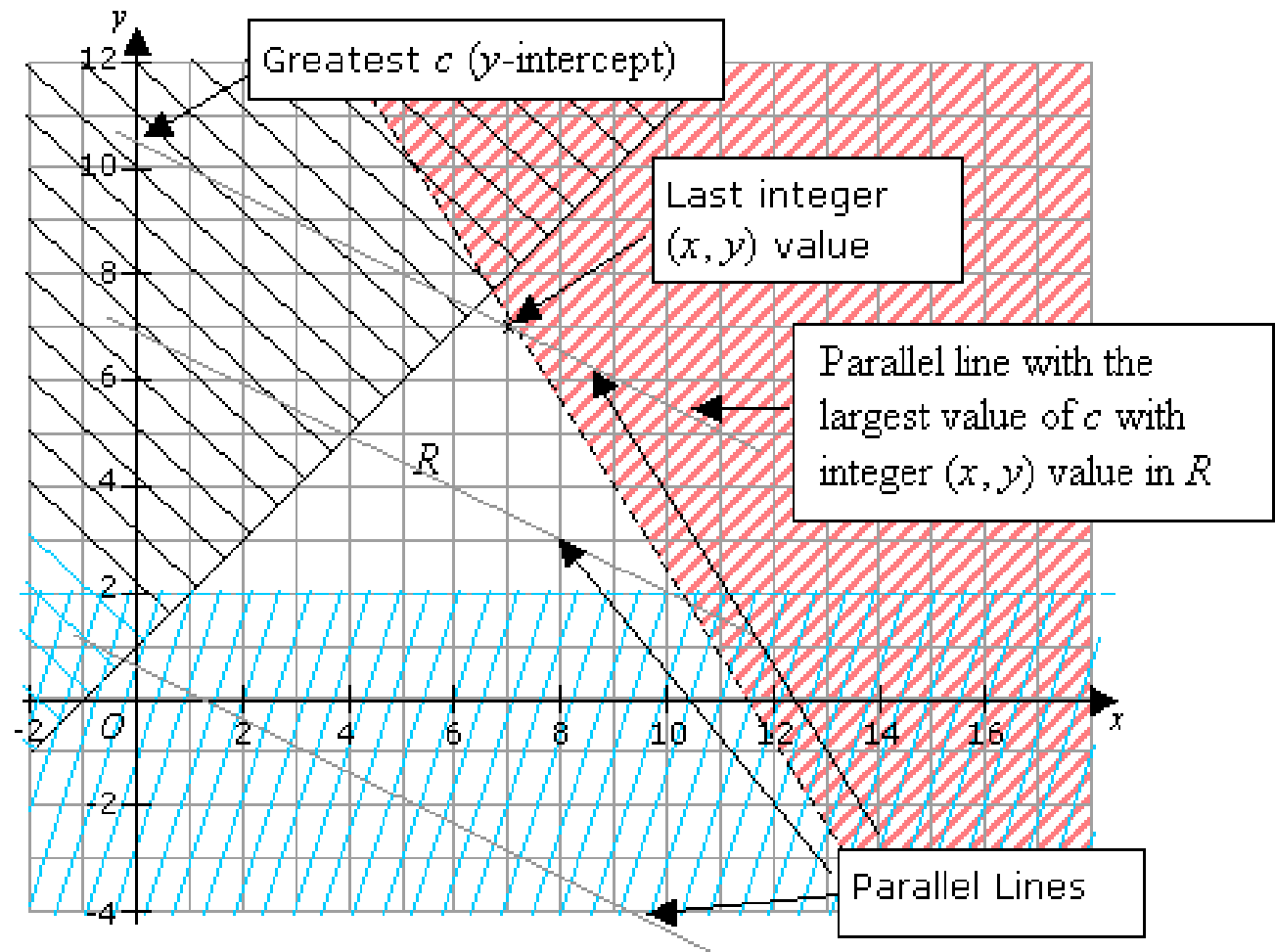
s.t.

$$y = x + 1,$$

$$5y + 8x < 92$$

$$y > 2$$

x, y integer



For solvers of LP/MP problems: see for instance google OR tools:
<https://developers.google.com/optimization/introduction/overview>

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 military 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

Mathematical programs in standard form

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

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

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

subject to

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

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

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 $\mathbb{F} = \{x \in \mathbb{S} \mid x \text{ feasible}\}$ 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 $g_i(x) = c_i$ are called **binding** or **active** constraints.

Classification: Mathematical Programming

| | Search-space \mathcal{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 | arbitrary |
| 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: $\mathcal{S} = \mathbb{R}^n, n_g = 0$ | | nonlinear |

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

A quadratic function is of the form:

$$c_0 + b_1x_1 + \dots + b_dx_n + a_{1,1}x_1x_1 + a_{1,2}x_1x_2 + \dots + a_{d,d}x_dx_d$$

If $A = (a_{ij}, i, j = 1 \dots d)$ is positive definite then the problem is termed convex QP.

For the comprehensive authoritative classification of INFORMS by Dantzig, see:

<http://glossary.computing.society.informs.org/index.php?page=nature.html>

Multiobjective Mathematical Program

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

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

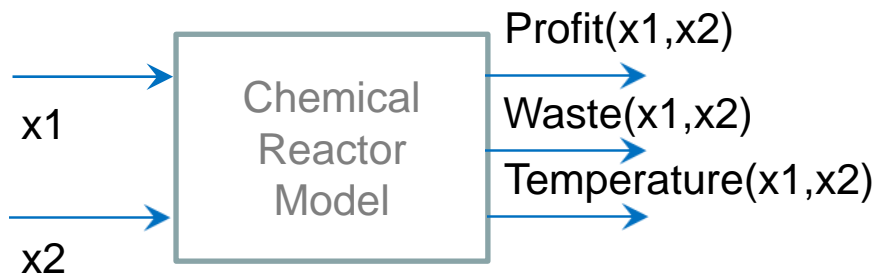
subject to

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

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

Example 1: 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?

- 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:

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

subject to

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

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

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

Example 2: Constrained 0/1 Knapsack Problem



Picture: © Michael Emmerich (instructor)

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]. $i = 1, \dots, d$ are indices of the items.

$$f_1(x_1, \dots, x_d) = \sum_{i=1}^d \frac{v_i}{[\$]} x_i \rightarrow \max$$

$$g_1(x_1, \dots, x_d) = \sum_{i=1}^d \frac{w_i}{[kg]} x_i - \text{MAXWEIGHT} \leq 0$$

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

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: 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.



$$f_1(x_1, \dots, x_d) = \sum_{i=1}^d \frac{v_i}{[\$]} x_i \rightarrow \max$$

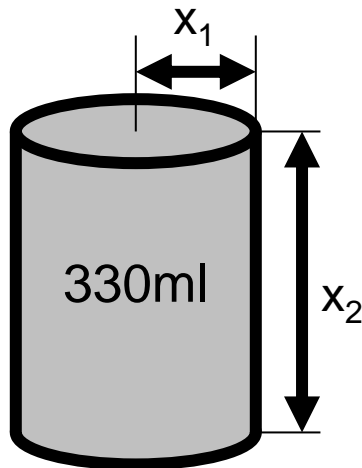
$$f_2(\mathbf{x}) = \sum_{i=1}^d \frac{w_i}{[kg]} x_i \rightarrow \min$$

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

Example: Equality Constraint for Tin Problem

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

Formulate this problem as a mathematical programming problem!



Problem sketch

$$f(\mathbf{x}) = 2\pi x_1 x_2 + 2\pi(x_1)^2 \rightarrow \min$$

$$h(\mathbf{x}) = \pi x_2 (x_1)^2 - 330 = 0$$

$$\mathbf{x} \in \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \infty \\ \infty \end{pmatrix} \right] \subset \mathbb{R}$$

This is a non-linear programming problem.

By substitution $x_1 x_2 =: x_3$ it can be transformed to a QP:

$$f(\mathbf{x}) = 2\pi x_3 + 2\pi(x_1)^2 \rightarrow \min$$

$$h(\mathbf{x}) = \pi x_3 x_1 - 330 = 0$$

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

For instance: Convex hull of N points with minimal surface and maximal volume.

Example: 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.



$$f_1(x_1, \dots, x_d) = \sum_{i=1}^d \frac{v_i}{[\$]} x_i \rightarrow \max$$

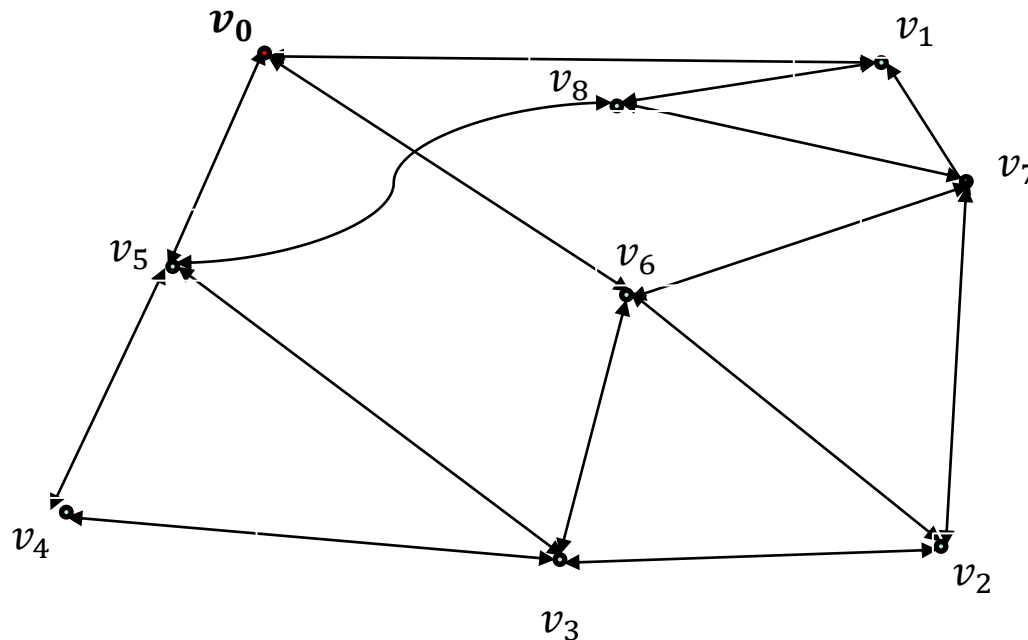
$$g_1(x_1, \dots, x_d) = \sum_{i=1}^d \frac{w_i}{[kg]} x_i - MAXWEIGHT \leq 0$$

$$g_2(x_1, \dots, x_d) = \sum_{i=1}^d x_i \leq MAXN$$

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

Example: Traveling Salesperson Problem

Given a distance matrix, find a tour of a vehicle that visits every city exactly once and starts from a depot and returns to it.



This can be easily described, but how to model this in terms of discrete (binary) variables and (constraint) functions that depend on them?

$$\sum_{i=1}^{n-1} \sum_{j=1}^n \sum_{k=1}^n d_{j,k} x_{i,j} x_{i+1,k} + \sum_{i=1}^n x_{1,i} d_{0,i} + \sum_{i=1}^n x_{n,i} d_{i,0} \rightarrow \min$$

subject to

$$\forall i \in 1, \dots, n : \sum_{j=1}^n x_{i,j} = 1$$

$$\forall j \in 1, \dots, n : \sum_{i=1}^n x_{i,j} = 1$$

$$x_{i,j} \in \{0, 1\}^{n \times n}$$

Here $d_{i,j}$ denotes the distance from i to j . and 0 is the index of the depot. If $x_{i,j} = 1$ and $x_{i+1,k} = 1$ then the vehicle drives in the i -th step from j to k .

How does this equation change if one is starting at node 0 and the tour needs to end in other depot, say node n ?

Encoding of a TSP tour

| City 1 | City 2 | City 3 | City 4 |
|--------|--------|--------|--------|
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |

This matrix encodes the tour

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 0$$

Solving min/max problems as LPs?

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

$$\max\{u_1(x_1, \dots, x_n), \dots, u_m(x_1, \dots, x_n)\} \rightarrow \min$$

where $u_1 \dots u_m$ are linear functions of the type $u_i(\dots) = a_{i1}x_1 + \dots + a_{in}x_n$?

Answer:

$$\begin{aligned} f(x_1, \dots, x_n, x_{n+1}) &= x_{n+1} \rightarrow \min \\ \text{s.t. } a_{i1}x_1 + \dots + a_{in}x_n &\leq x_{n+1}, i = 1, \dots, m \end{aligned}$$

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

Let us assume the problem $a_1x_1 + \dots + a_nx_n \rightarrow \min$

s.t. $w_1x_1 + \dots + w_nx_n \leq c, x_i \in \{0,1\}, i = 1 \dots n$

How can we formulate it as a continuous problem?

Answer:

$$\begin{aligned} (1 - x_i)x_i &= 0, i = 1, \dots, n \\ x_i &\in IR \end{aligned}$$

Placement of unit discs on the square – mixed integer problem; switching on/off units

How to formulate the problem of placing as many unit discs of radius 10cm as possible on a square of sidelength 1meter?

The unit discs may not overlap.

Choose $n = (100 \times 100) / (2\pi \cdot 10^2)$

$$f(b_1, \dots, b_n, x_1, \dots, x_n, y_1, \dots, y_n) = \sum b_i \rightarrow \max$$

$$b_i b_j \sqrt{\{(x_i - x_j)^2 + (y_i - y_j)^2\}} \geq 20$$

Alternative: Linear penalty with large constant LC

$$\sqrt{\{(x_i - x_j)^2 + (y_i - y_j)^2\}} \leq 10 + (1 - b_i) LC + (1 - b_j) LC$$

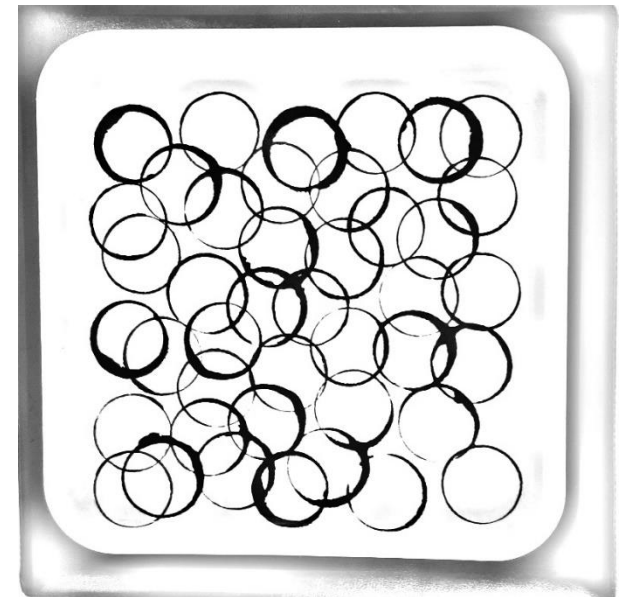
$$i = 1, \dots, n - 1, j = i + 1, \dots, n$$

$$b_i \in \{0, 1\}$$

$$x_i \in [10, 90]$$

$$y_i \in [10, 90]$$

Linear penalty often advantage; problems can remain linear or quadratic in many cases.



Unit disc problem: Infeasible design due to overlap.

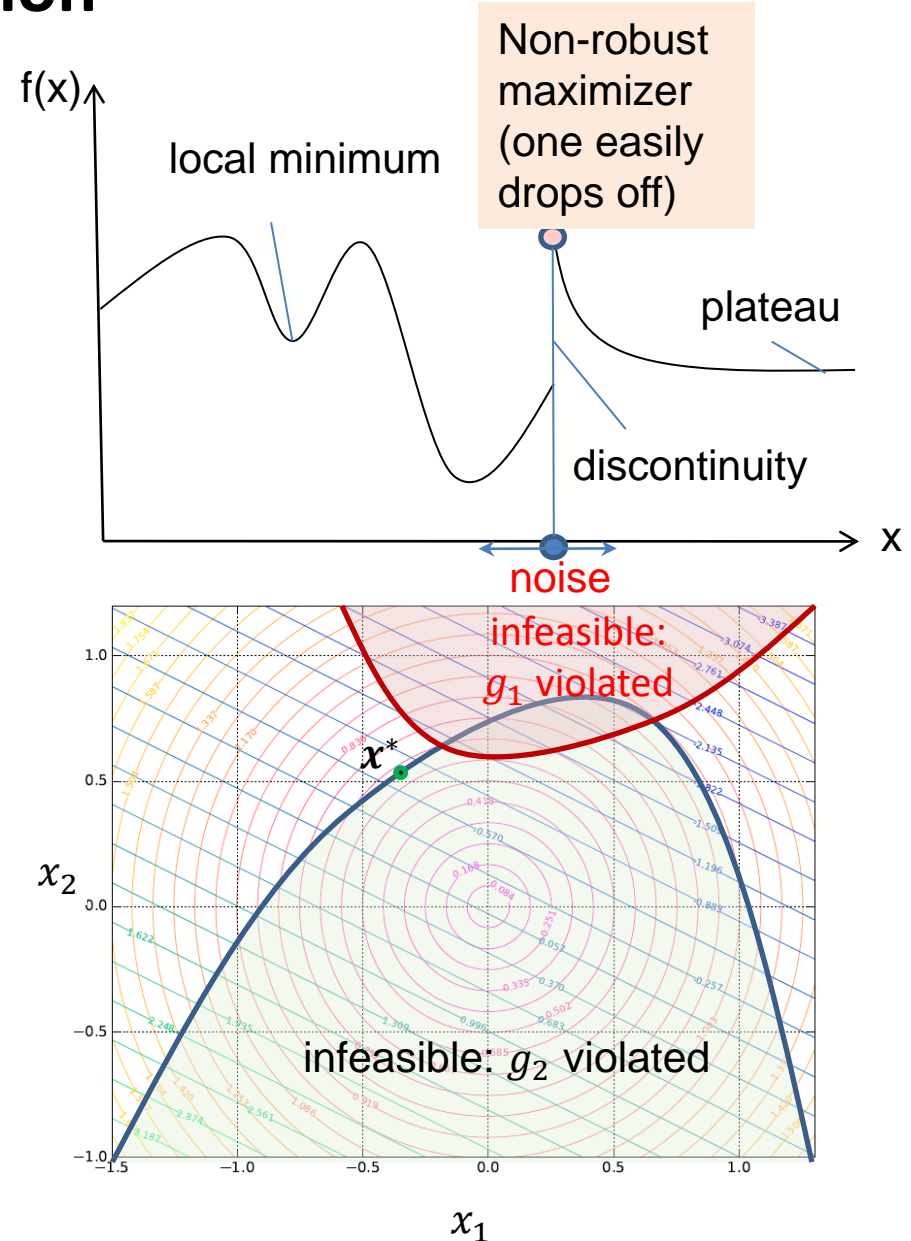
Complexity of optimization problems

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 $\mathbf{x}_1, \dots, \mathbf{x}_t$, 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

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

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:

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


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)

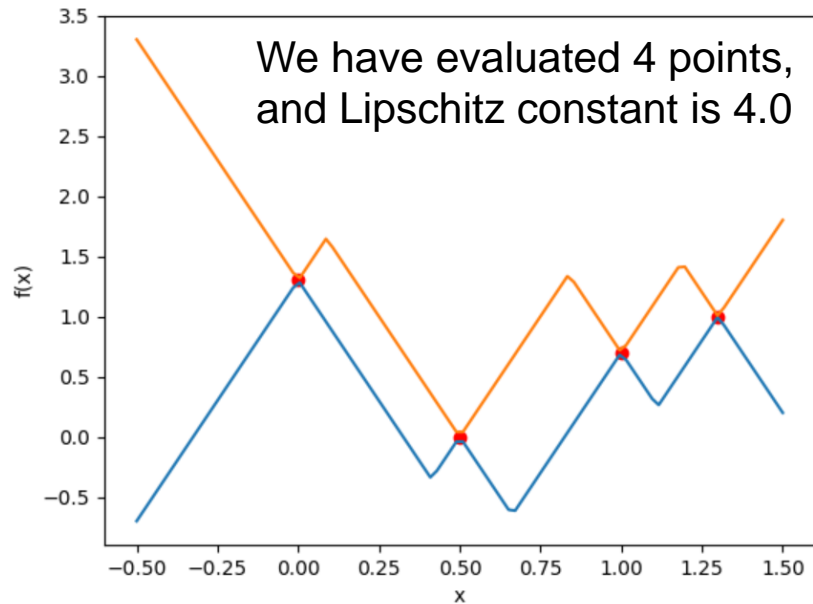
Lipschitz continuity uncertainty margins (1-D python code)



<https://trinket.io/python3/c38e5ebdbc>

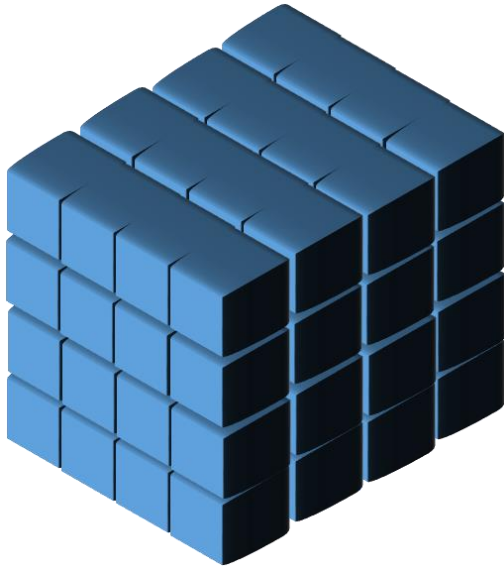
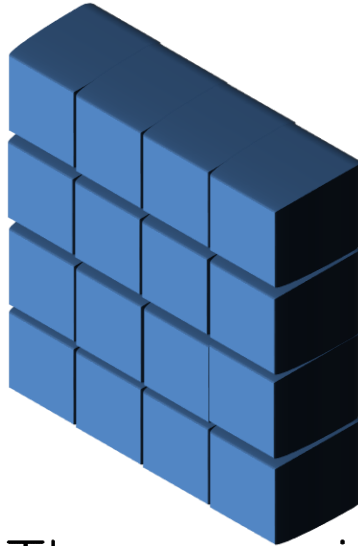
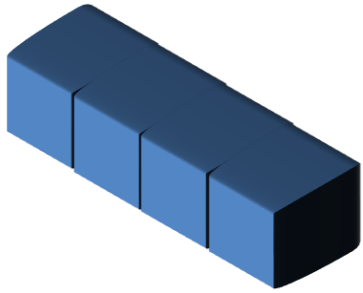
```
31 #####
32 # Enter the evaluated data points here #
33 #####
34 x=[0.,0.5,1,1.3]
35 y = [1.3,0.0,0.7,1.0]
36
37  $\theta = [1, 10]$ 
38
39  $\sigma_0 = \text{exponential\_cov}(\theta, \theta)$ 
40  $\sigma_1 = \text{exponential\_cov}(x, x, \theta)$ 
41  $x\_pred = \text{np.linspace}(-0.5, 1.5, 100)$ 
42  $y\_up = \text{np.linspace}(-0.5, 1.5, 100)$ 
43  $y\_low = \text{np.linspace}(-0.5, 1.5, 100)$ 
44
45 predictions = [predict(i, x, exponential_cov,  $\theta$ ,  $\sigma_1$ , y) for i in x_pred]
46 y_pred, sigmas = np.transpose(predictions)
47 #plt.errorbar(x_pred, y_pred, yerr=sigmas, capsize=0)
48 plt.plot(x, y, "ro")
49 # Add title and axis names
50 #plt.title('One Dimensional Krigin')
51 plt.xlabel('x')
52 plt.ylabel('f(x)')
53 plt.show()
54
55 # Here starts the Lipschitzian bound computation
56 L=4
57 for i in range(len(x_pred)):
58     xnew = x_pred[i]
59     d=[L*np.abs(z-xnew) for z in x]
60     lowenv=max(np.subtract(y,d))
61     upenv=min(np.add(y,d))
62     y_pred[i] = (upenv+lowenv)/2
63     y_up[i]=upenv;
64     y_low[i]=lowenv;
65
66 #print(d)
67 #print(upenv)
68 #print(lowenv)
69 #print(y_pred);
70 #plt.plot(x_pred, y_pred)
71 plt.plot(x_pred, y_low)
72 plt.plot(x_pred, y_up)
73 plt.show();
74
75
```

Powered by  trinket



[trinket_plot.png](#)

Curse of dimensionality (proof)



Assume we partition the d -dimensional space into d -dimensional cuboids with a single point in a cube C_ϵ^d of sidelength ϵ . The best information about the black box Lipschitz function we can obtain for a point in the center of the cube \mathbf{x}_c . We can guarantee an accuracy of:

$$|f(\mathbf{x}_c) - f^*| \leq k \sqrt{\sum_{i=1}^d (\frac{1}{2}\epsilon)^2} = k \sqrt{\frac{d}{4}} \epsilon$$

$\lfloor \frac{1}{\epsilon} \rfloor^d$ ϵ -cubes fit at least into $[0, 1]^d$.

In the worst case all black box evaluations result in the same value.

Then we require $\lfloor \frac{1}{\epsilon} \rfloor^d$ evaluations in order to guarantee

that we found optimum with accuracy $k \sqrt{\frac{d}{4}} \epsilon$

For a desired accuracy of δ we get $k \sqrt{\frac{d}{4}} \epsilon \leq \delta$. And

therefore $\epsilon \leq \frac{\delta}{2k\sqrt{d}}$, and more than $\left\lfloor \frac{2k\sqrt{d}}{\delta} \right\rfloor^d \in \Theta\left(\left(\frac{k}{\delta}\right)^d\right)$ evaluations for a given dimensionality d .

Note: Same bounds hold for approximation and prediction of Lipschitz continuous functions, regardless the learning method (even autotuned deep neural networks cannot do better).

Ergo: (Similar input \Rightarrow Similar Output) assumption does not save us from the curse of dimensionality in the worst case.

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 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 it provides a witness 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 .



'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

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

NP hard problems

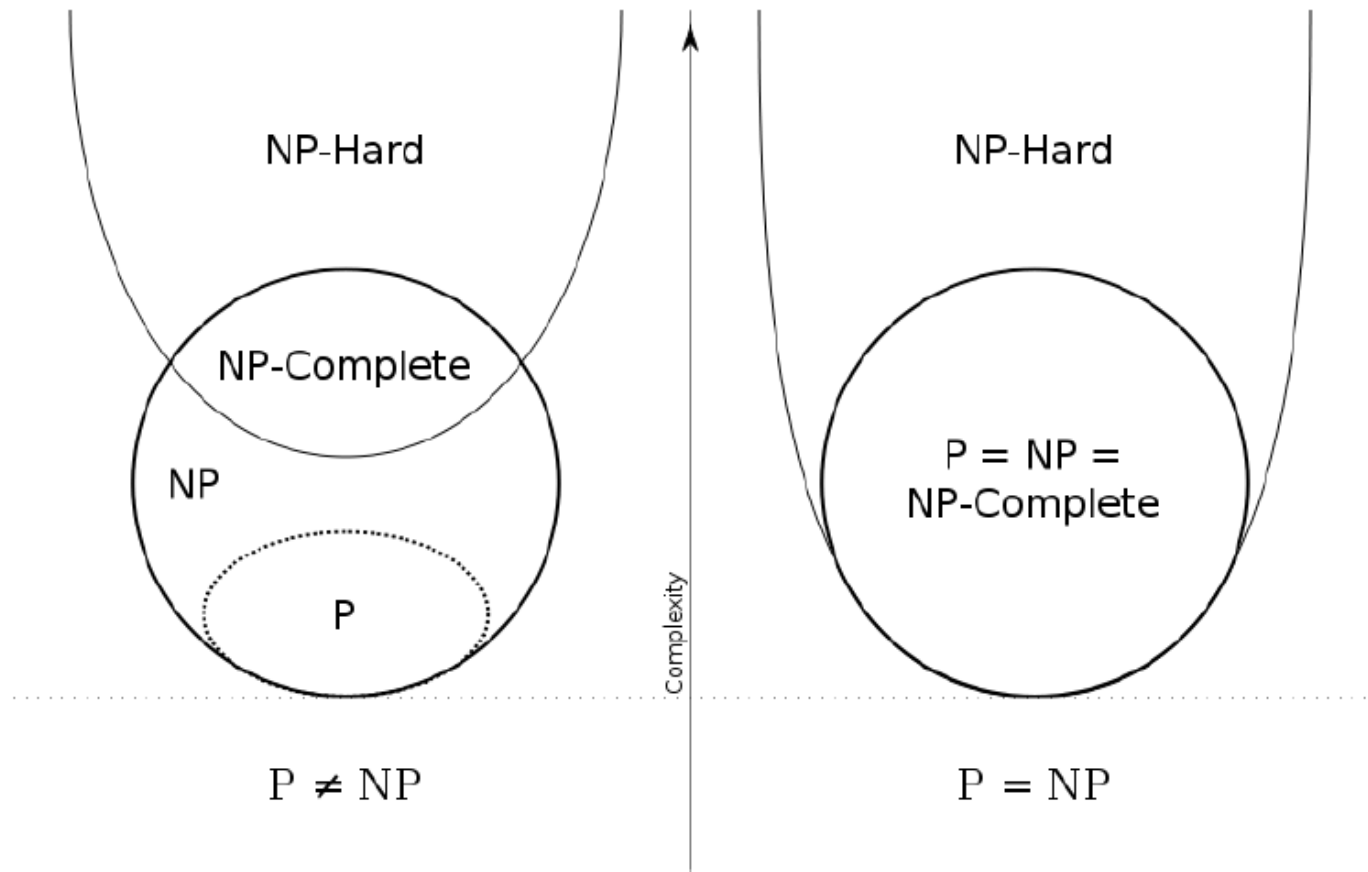
For NP complete problems no polynomial time algorithms are available, yet.

Examples: Decision versions of Knapsack, Traveling Salesperson Problem, Maximal Clique, Minimal Separation of Graph, Scheduling, Integer Linear Programming.

An NP hard problem, is a problem with the property that if it could be solved in polynomial time, also all other problems in NP, including NP complete problems, could be solved in polynomial time (there exists a polynomial time reduction from NP complete problems to it).

Examples: Nonlinear Programming, Quadratic Programming, Capacitated Vehicle Routing

NP, NP-Complete, NP-Hard



Source: Wikipedia

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.

MY HOBBY:

EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS

| CHOTCHKIES RESTAURANT | |
|-----------------------|------|
| ~ APPETIZERS ~ | |
| MIXED FRUIT | 2.15 |
| FRENCH FRIES | 2.75 |
| SIDE SALAD | 3.35 |
| HOT WINGS | 3.55 |
| MOZZARELLA STICKS | 4.20 |
| SAMPLER PLATE | 5.80 |
| ~ SANDWICHES ~ | |
| BARBECUE | 6.55 |

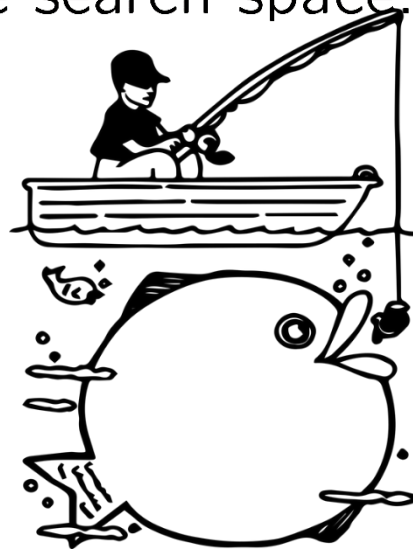


<https://xkcd.com/287/>
This work is licensed under a [Creative Commons Attribution-NonCommercial 2.5 License](https://creativecommons.org/licenses/by-nc/2.5/).

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”

(c) (left, right picture) Michael Emmerich
(instructor)

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-bear-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 mathematical 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