Advanced OR methods in Operations Management

Prof. Dr. Stefan Bock

University of Wuppertal

Business Computing and Operations Research

Information concerning the course

- Lecturer: Prof. Dr. Stefan Bock
  - Office: M12.02
  - Office hour: Monday 4:00pm-5:30pm (11:30am-12:30 pm)
    (or whenever WE want, i.e., just write an email in order to initiate an appointment)
  - Secretary office (just the neighboring one): M12.01
  - Email: sbock@winfor.de

- Supervisor of the tutorial: Anna Katharina Janiszczak
  - Office: M12.33
  - Office hour: After registration by email
  - Email: kjaniszczak@winfor.de

Schedule

- Lecture:
  - Monday 6pm-8pm, room: M12.25
  - Thursday 10am-12pm, room: M15.09
  - Start: April 04th, 2019
  - Slides are available at www.winfor.de

- Tutorial
  - Wednesday 2pm-4pm
  - Room: M12.25
  - Start: April 10th, 2019
  - Moodle login password: orm-sose-2019

Weekly schedule of assignments

Tue

Assignments online (10 a.m.)

Wed

Tutorial: Questions concerning recent assignments

Tue

Due date for handing in assignments (10 a.m.)

Wed

Tutorial: Assignments review

Agenda (Part I)

Section 1: Introduction

1. Optimization Problems – The continuous and the discrete world
2. Introduction to Complexity theory
3. Dealing with NP-Completeness
4. Some basic OM definitions
5. Levels of planning

Section 2: Dynamic Programming (DP)

1. Basic motivation
2. A simple starter: The Knapsack Problem
3. The asymmetric Traveling Salesman Problem with hard time windows
4. Solving a variant of the Line-TSP to optimality
5. Dynamic Programming in Scheduling

Agenda (Part II)

Section 3: Branch&Bound approaches (B&B)

1. Basic search and enumeration strategies
2. A B&B approach for the Knapsack Problem
3. Little's B&B approach for the aTSP
4. Scholls B&B approach SALOME for the Simple Assembly Line Balancing Problem I (SALBP-1)

Section 4: Column generation (CG)

1. Basic motivation
2. The Cutting-Stock Problem

Agenda (Part III)

Section 5: Lagrangian Relaxation (LR)

1. Basic idea
2. Applying LR to the Knapsack Problem – An illustrative, but not reasonable case
3. A solution approach of the sTSP

Section 6: Heuristics and Meta heuristics

1. Basic ideas and simple construction heuristics
2. Simulated Annealing (SA) (solving the QAP)
3. Genetic Algorithms (GA)
4. Sophisticated Tabu Search approaches for the Vehicle Routing Problem (TS)

Agenda (Part IV)

Section 7: Real-time control (RC)

1. Basic motivation
2. Dynamic Vehicle Routing Problems
3. Integrating diversion into DVRP approaches
4. Exploiting knowledge about future demands
5. Dynamic scheduling

1. Introduction

- In what follows, we consider specific sophisticated approaches providing decision support for business
- These approaches are applicable to complex decision problems in Production and Logistics
- Specifically, in what follows, we consider specific problem models mapping realistic industrial problems
- It turns out that these problems have a considerable computational complexity
- Consequently, in order to solve these problems efficiently, we propose sophisticated solution approaches
- Thus, the lecture commences with some basics in production planning theory and Combinatorial Optimization

1.1 Optimization problems

- In what follows, we consider various optimization problems
- Therefore, we start with a brief characterization of these problems
- Main attributes
  - **Given:**
    - Parameters
    - Restrictions
  - **Sought:**
    - Variables (determine a complete solution, this solution is denoted as feasible if all restrictions are fulfilled)
    - Hence, only values of variables are changeable
  - **Quality is measured according to a pursued objective function:**
    - Assessment of feasible solutions
    - Minimization or maximization of an objective function value

Continuous LPs

- Are widely addressed during your Bachelor class
- All LPs have in common that…
  - …the variables are continuous
  - …the objective function is linear
  - …the restrictions are linear
  - …the objective function is either a maximization or minimization
  - …restrictions require the fulfillment of a minimum or maximum bound
- Continuous LPs can be efficiently solved by the Simplex Method

Forms of LPs

- **Linear Programs (LP) are defined in a specific form**
- In Literature, different forms of LPs are distinguished. Specifically, it can be found for instance
  - LP in general form
  - LP in canonical form
  - LP in standard form
- Reader should be warned that this classification is far away from being unambiguous
- Moreover, what we will denote as an LP in standard form is frequently introduced as the LP in canonical form

General Form

Let $A \in IR^{m \times n}$ with $A = \begin{pmatrix} a_1'^T \\ \dots \\ a_m'^T \end{pmatrix} \wedge a_i' \in IR^n, i \in \{1,\dots,m\}$ and $b \in IR^m$.

Furthermore, let M be the set of row indices corresponding to equality constraints, and let $\bar{M}$ be the set of row indices corresponding to inequality constraints. Additionally, let N be the set of column indices corresponding to constrained variables and let $\bar{N}$ be the set of column indices corresponding to unrestricted variables. Then, the feasible solution space P is

$P = \{x \in IR^n \mid \forall j \in N : x_j \geq 0 \wedge \forall i \in M : a_i'^T \cdot x = b_i \wedge \forall i \in \bar{M} : a_i'^T \cdot x \leq b_i\}$.

Furthermore, for $c \in IR^n$, we pursue the maximization of $z(x) = c^T \cdot x$.

Note that M and $\bar{M}$ form a partition of $\{1,\dots,m\}$. Moreover, N and $\bar{N}$ are a partition of $\{1,\dots,n\}$.

Canonical Form

Let $A \in IR^{m \times n}$ and $b \in IR^{m}$:

Then the set of feasible solutions is defined as follows:

$P = \{x \in IR^{n} \mid x \geq 0 \text{ and } A \cdot x \leq b\}$

Solutions that belong to P are denoted as feasible.

In order to evaluate the found solution, we introduce an additional vector. Hence, let $c \in IR^{n} : z(x) = c^{T} \cdot x$

In the following, we pursue the maximization of z under the constraints $x \geq 0$ and $A \cdot x \leq b$

Standard Form

Let $A \in IR^{m \times n}$ and $b \in IR^{m}$:

Then the set of feasible solutions is defined as follows:

$$P = \left\{ x \in IR^{n} \mid x \geq 0 \text{ and } A \cdot x = b \right\}$$

Solutions that belong to P are denoted as feasible.

In order to evaluate the found solution, we introduce an additional vector. Hence, let $c \in IR^{n}$: $z\left(x\right) = c^{T} \cdot x$

In the following, we pursue the minimization or maximization of z under the constraints $x \geq 0$ and $A \cdot x = b$

Conclusion

- Since all these forms are equivalent, we almost always use the canonical or the standard from
- Thus, we consider LPs of the form

$$\boxed{\begin{array}{l}\text{Minimize } \sum_{j=1}^{n} c_j \cdot x_j, \text{ subject to} \\ A \cdot x = b \text{ and } x \geq 0, \text{ with } x \in IR^n, A \in IR^{(m \times n)}, b \in IR^m\end{array}}$$

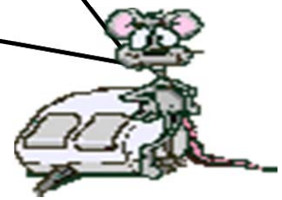
- or

$$\boxed{\begin{array}{l}\text{Maximize } \sum_{j=1}^{n} c_j \cdot x_j, \text{ subject to} \\ A \cdot x \leq b \text{ and } x \geq 0, \text{ with } x \in IR^n, A \in IR^{(m \times n)}, b \in IR^m\end{array}}$$

LPs are nice

I am familiar with LPs, Simplex Algorithms, Duality Theory and some well-solvable combinatorial problems

For those who are not familiar with these issues, download our free slides of the Bachelor class "Combinatorial Optimization"



The reward of our knowledge

- Applying this knowledge, we can solve huge continuous LPs with thousands of restrictions and variables within a few milliseconds
- However, this course deals with much more complex problems and applies LPs only to derive needed bounds
- Nevertheless, our knowledge is very important to derive adequate solution methods and tools
- These methods are – by far not – so universal, but may be effective for specific real-world problems

Integer problems

- Unfortunately, most interesting (but complex) decisions in life and business are characterized by the fact that they are not continuous
- “Should I or should I not marry her/him?”
- Thus, some decisive variables are binary or integer
- Consequently, our problem definition has to be adapted to:

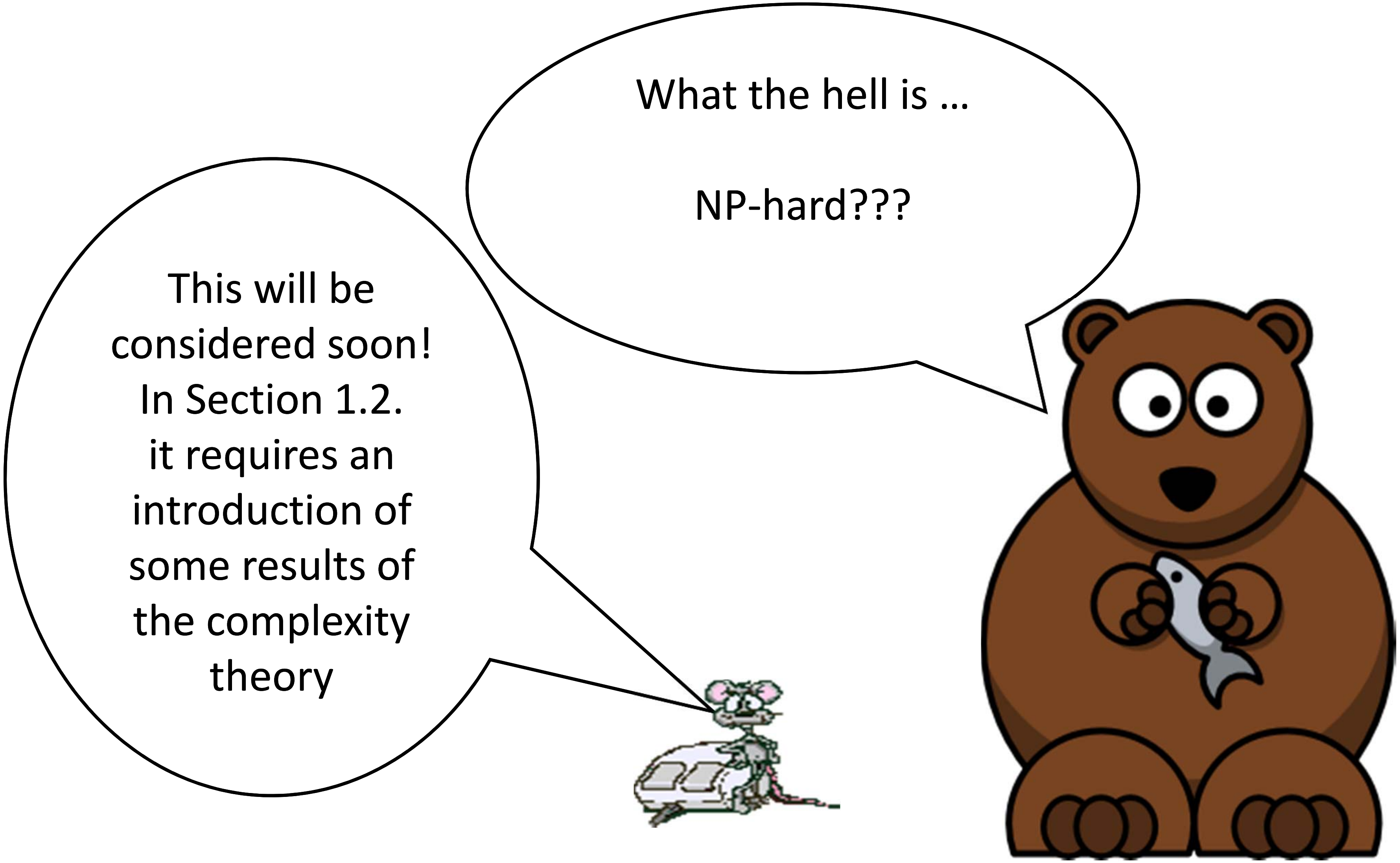
$$\text{Maximize } \sum_{j=1}^{n} c_j \cdot x_j \text{ subject to}$$

$$A \cdot x = b \text{ and } x \geq 0, \text{ with } x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \text{ and } x_1 \in Z^{n_1}, x_2 \in IR^{n_2}, n = n_1 + n_2$$

Integer problems

- Linear integer programming is unfortunately NP-hard
- I.e., out of current knowledge, we assume that it is not possible to solve this problem with an algorithm whose running time is strongly polynomial
- Unfortunately, since those problems are of significant interest, we have to provide new techniques
  - that find best integer solutions
  - but cannot avoid exponential running times for specific worst case scenarios
- This is addressed in the following sections

Now, some of you may say…



1.2 Basics of computational complexity theory

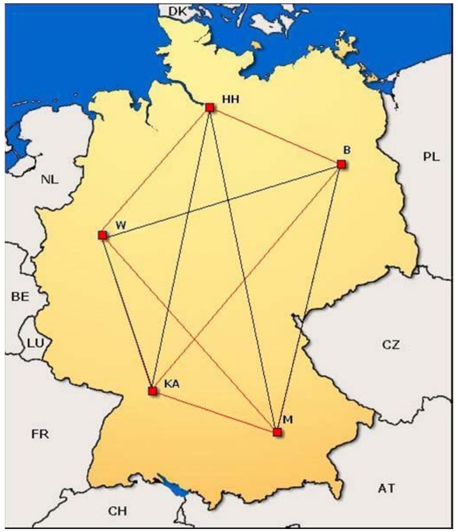
- In this lecture, we consider different mathematical optimization problems
- We want to generate solutions for our defined models generating production plans of the highest possible quality
- But practical experiments show that some computational problems are easier to solve than others
- Additionally, there are a lot of problems where we assume that finding an optimal solution can be generated only by a total enumeration of all possible constellations

Basics of computational complexity theory

- Therefore, in order to get a satisfying answer to our problems, we need some rules or better a complete theory which tells us which kind of problems are hard to solve
- In this subsection, we want to consider the main attributes of such a theory – the Theory of *NP*-Completeness
- Basic results of this theory will be referenced throughout this lecture

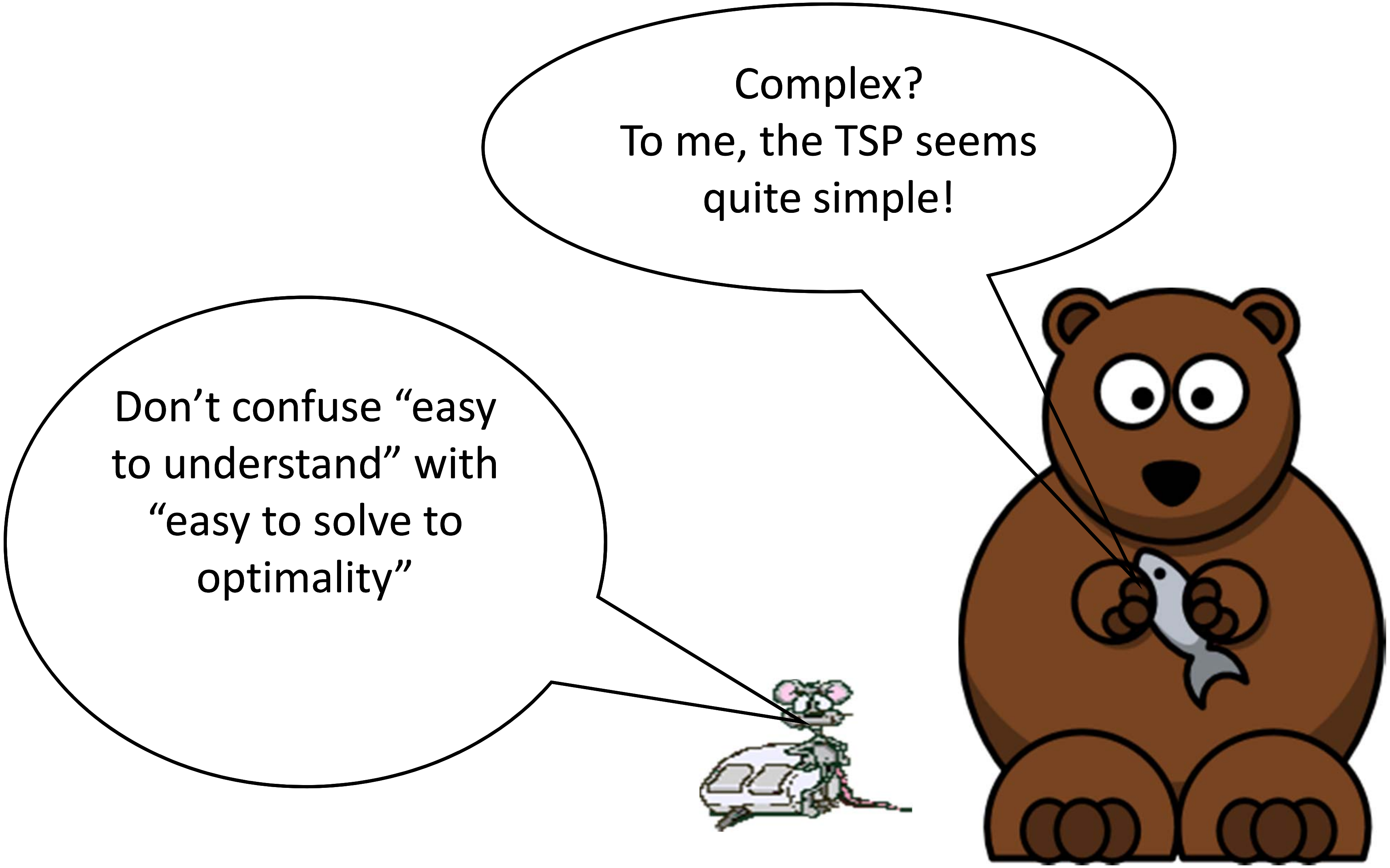
An example: The Traveling Salesman

- Let C={c_1, c_2, …, c_n} be a finite set of cities and a distance given for each pair of cities.
- Optimization problem: Determine the tour, i.e., a circle passing each node exactly once and all cities in C with minimal length!



- Decision problem: Does a tour exist with length lower than or equal to a given threshold B≥0?

Complexity of the TSP



Finding an optimal solution

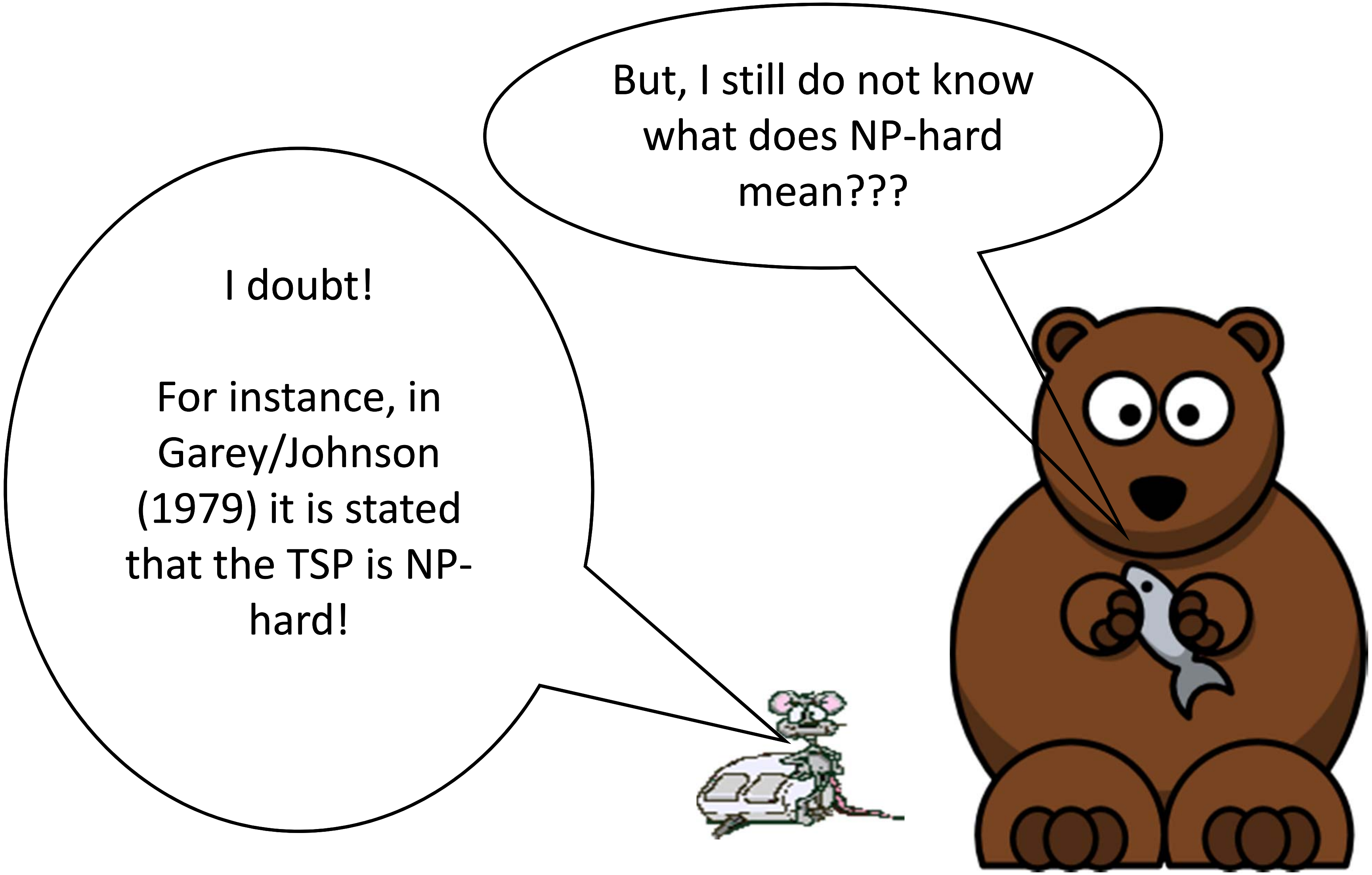
- The TSP is one of the most frequently considered optimization problems in the literature
- Hence, numerous exact solution approaches were proposed
- However, applied to worst case scenarios, each approach has to asymptotically consider and assess all possible tour plans
- These are up to (N-1)!
- This may be very time-consuming (see the next slide)

Time consumption

N=	Number of tours	Seconds per tour evaluation	Computational time (in seconds)	In days (86400 sec. per day)
6	120	1/1000	0,12	$0{,}1389 \cdot 10^{-5}$
11	3628800	1/1000	3628,8	0,042
12	39916800	1/1000	39916,8	0,461990741
13	479001600	1/1000	479001,6	5,544
14	6227020800	1/1000	6227020,8	72,072
15	87178291200	1/1000	87178291,2	1009,008
16	$1{,}307674368 \cdot 10^{12}$	1/1000	1307674368	15135,12
17	$2{,}092278999 \cdot 10^{13}$	1/1000	20922789888	242161,92

- Exponential increase can be observed
- However, may be there exists a better solution algorithm and we just have to spent more effort in finding it.

Can we do it better?



How difficult is it to show intractability?

I can't find an efficient
just too dumb.

To prove this is the aim of the complexity theory
(At least, if we assume that an efficient solution is not possible)

However:
NP-hart means – at least theoretically – something different

ficient algorithm, because no
; possible.

Reference: Garey and Johnson 1979

What is NP-hardness?

- Roughly speaking, an NP-hard problem P is a decision problem that is that complex that,
  - all problems of a class, in what follows denoted as NP (among these are numerous important and famous problems with unknown efficient solution algorithm), are reducible to this problem P
  - i.e., P is able to map or model all these famous but complex problems of NP
  - i.e., if we provide an efficient solution algorithm for P we simultaneously solve all problems of class NP

The message of NP-completeness

From a practical point of view, this a comparable result since we can assume that a new algorithm that efficiently solves all these problems simultaneously does not exist

Unfortunately, this is **not** mathematically proven by the complexity theory (np-completeness theory)

I can't find an efficient algorithm, I guess I'm just too dumb.

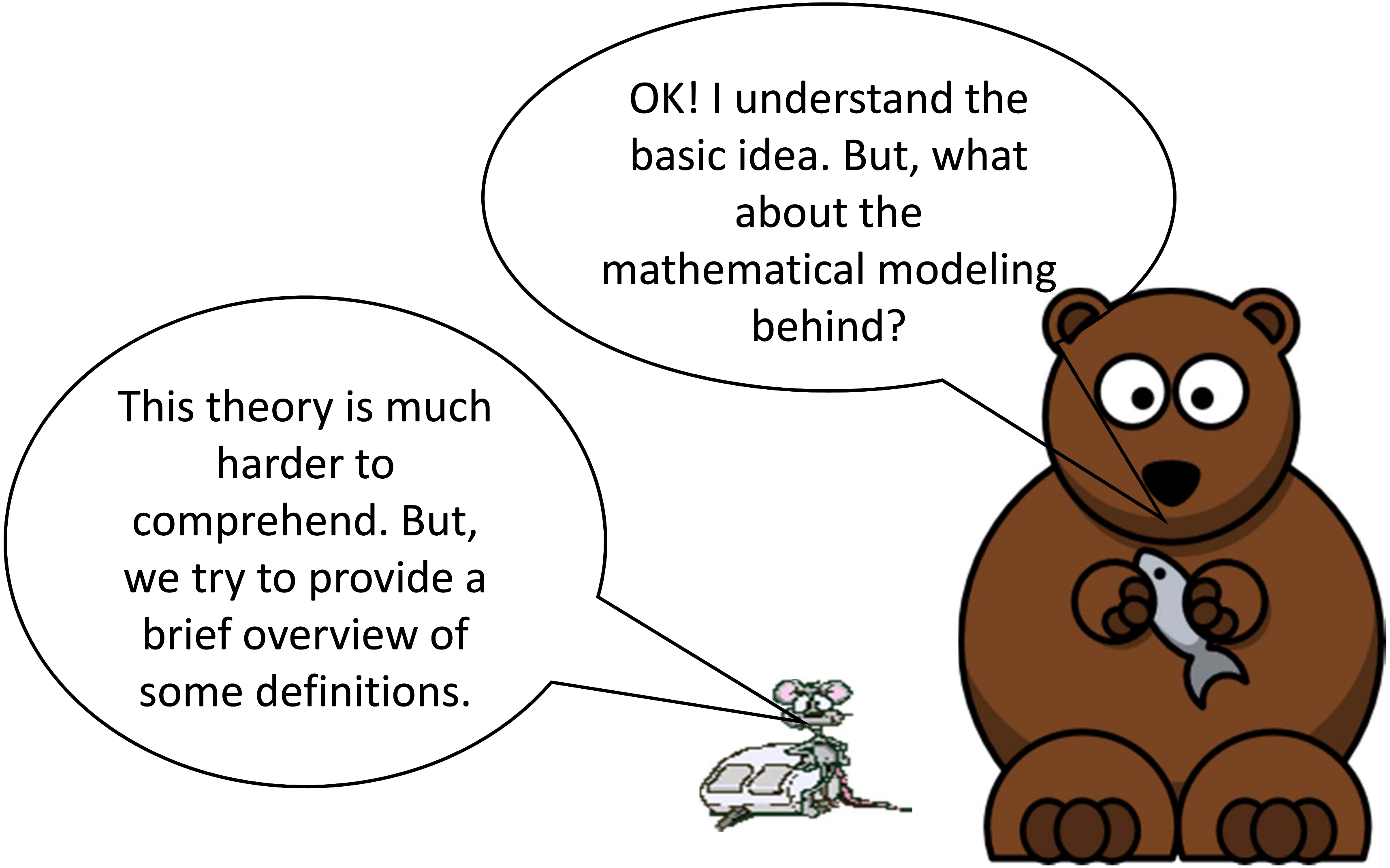
I can't find an efficient algorithm, because no such algorithm is possible.

This is mathematically proven by the complexity theory / NP-completeness theory (on the basis of a very long and increasing list of problems)

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

Reference: Garey and Johnson 1979

The theory behind the idea



Basic definitions

- A computational problem can be viewed generally as a function h that maps each input x of a given domain to a well defined output h(x) of another given domain

- The respective program generates h(x) for each input x
- In what follows, we compare the performance of different procedures according to their computational effort. We define T(n) as the number of necessary steps the algorithm takes at most to compute each output for an input of length n. T is called the **run-time function** of the respective algorithm

Basic definitions

- Note that in most cases a precise definition of T becomes a nearly unsolvable task
- Therefore, we introduce **performance or complexity classes** of run-time functions. As a consequence, these classes are sets of functions
- We say **T(n) belongs to the class O(g(n))** if and only if there exists a constant $c>0$ and a nonnegative integer n_0 such that $T(n) \leq c \cdot g(n)$ for all integers $n \geq n_0$

Important O classes

	O class	Example
constant	$O(1)$	Multiplying two numbers
logarithmic	$O(\log n)$	Binary search
linear	$O(n)$	Sum of n numbers
n-log-n	$O(n \cdot \log n)$	Sorting n numbers (heap sort)
quadratic	$O(n^2)$	Wagner-Within algorithm
polynomial	$O(n^k)$, $k \geq 1$	Matrix multiplication $O(n^3)$
pseudo-polynomial	$O(n^k m^l)$, $k,l \geq 1$	Knapsack Problem (Dynamic Programming)
exponential	$O(b^n)$, $b>1$	Simplex Algorithm

Polynomially solvable / Decision problems

Definition 1.2.1

A problem is called polynomially solvable if there exists a polynomial p such that $T(n) \in O(p(n))$ for all possible inputs x of length n $(|x| = n)$, i.e., if there exists k such that $T(|x|) \in O(|x|^k)$.

Definition 1.2.2

A problem is called a decision problem if the output range is restricted to $\{\text{yes, no}\}$. We may associate with each combinatorial minimizing problem a decision problem by finding a threshold k for the corresponding objective function f. Consequently, the decision problem is defined as:

Does there exist a feasible solution S such that $f(S) \le k$?

The classes *P* and *NP*

Definition 1.2.3 (The class *P*):

The class of all decision problems which are polynomially solvable is denoted by P.

Definition 1.2.4 (The class *NP*):

The class *NP* comprises all decision problems fulfilling the following attributes:

(i) Each input x that leads to an yes-output posseses a certificate y, such that $|y|$ is bounded by a polynomial in $|x|$.

(ii) There exists an algorithm that verifies in polynomial time that y is a valid certificate for x.

What is *NP*?

- The definition of the class *P* seems to be obvious. It comprises all problems which can be solved with a reasonable effort
- In contrast to this, the definition of the class NP seems to be somehow artificial. Therefore, we give the following additional hints for a better understanding:
  - The string y can be seen as an arbitrary solution possibly fulfilling the restrictions of our defined problem
  - If this string y is feasible and fulfills the defined restrictions of the decision problem, a “witness for the yes-answer is found”

What is *NP*?

- Therefore, the class *NP* consists of all decision problems where for all inputs with a yes-output an appropriate witness can be generated and certified in polynomial time, e.g., **generating the representation of the solution and the corresponding feasibility check only needs polynomial computational time**
- In theoretical computer science, the **model of a non-deterministic computer system** is proposed which is able to guess an arbitrary string
  - After generating this string, the system changes back to a deterministic behavior and checks the feasibility of this computed solution.
  - An input x is accepted (output is yes) by such a system, if there is at least one computation starting with x and leading to the output yes. Its effort is determined by the number of steps of the fastest accepting computation

What is *NP*?

- *NP* and *P* separate problems from each other
  - In *P,* there are the problems that we can solve efficiently.
  - But from the problems in *NP,* we only know that the representation of a solution and its feasibility check can be computed in a reasonable amount of time

Conclusions

First of all, we can easily derive: $\boxed{P \subseteq NP}$

However:

- One of the major open problems in theoretical computer science or modern mathematics is whether P equals NP or not.
- It is generally conjectured that this is not the case.
- In order to provide the strong evidence that P is not equal to NP, a specific theory was generated by different authors, especially by Cook.
- This theory is called the **Theory of NP-Completeness** and is discussed subsequently.

NP-Completeness – instrument of reduction

- The basic principle in defining the NP-Completeness of a given problem is the **method of reduction**
  - For two decision problems S and Q, we say S is reduced to Q if there exists a polynomial-time computational function g that transforms inputs of S into inputs of Q such that **x is a yes-input for S if and only if g(x) is a yes-input for Q**
  - How can we explain the meaning of this method playing the central role in the Theory of NP-Completeness?
    - By reducing a problem S to Q, we say somehow that if we can solve Q in polynomial time, we can solve S in polynomial time, too
    - That means Q is at least as hard to solve as S (according to our classification)

NP-Completeness – instrument of reduction

- To understand this, imagine we have a polynomial time restricted solution algorithm A deciding Q. Then, we can use the computed reduction defined above in order to decide S in the following way:
  - Input: x (for S)
  - Generate g(x) (Input for Q) in polynomial time
  - Decide by using A whether g(x) belongs to Q (polynomial effort)
  - Output: yes if g(x) belongs to Q, otherwise no

  Consequently, we have designed a polynomial time restricted decision procedure deciding S
- Consequence: If Q belongs to P, by knowing S can be reduced to Q, S also belongs to P

$\mathcal{NP}$-hardness, $\mathcal{NP}$-Completeness

1.2.5 Definition:

A problem P is called **$\boldsymbol{NP}$-hard** if all problems belonging to NP can be reduced to P.

1.2.6 Definition:

A problem P is called **$\boldsymbol{NP}$-complete** if P is NP-hard and P belongs to NP.

Consequences

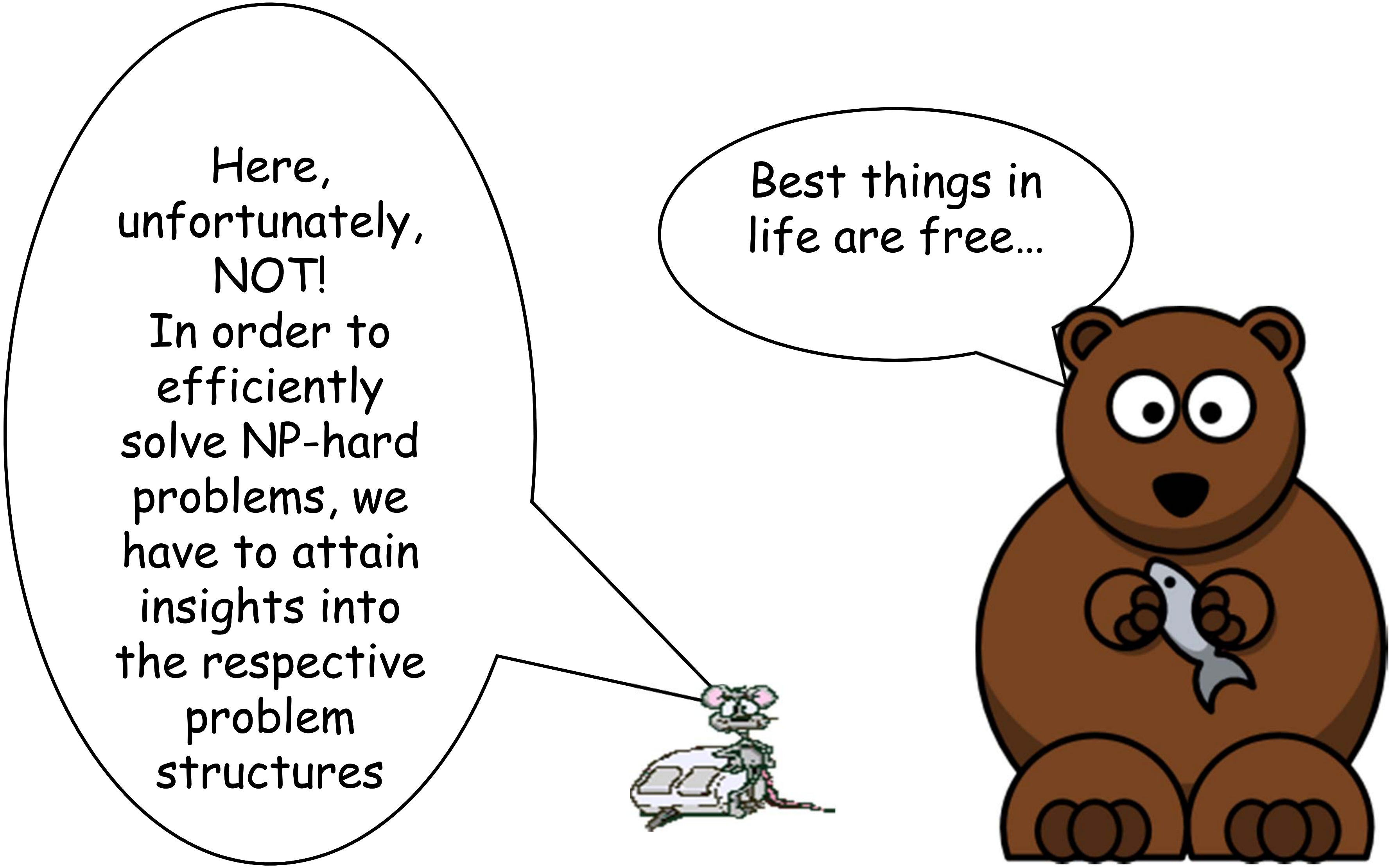
- If any single NP-complete problem P could be solved in polynomial time, all problems in NP can be solved in polynomial time, too. Therefore, in this case we can derive $P=NP$
- In order to prove that a specific problem Q is NP-hard, it is sufficient to show that an arbitrary NP-hard or NP-complete problem C can be reduced to Q
- But how can we find a **starting point of this Theory**? What do we need is a **first NP-complete or NP-hard problem**
- Cook has shown that the problem **SAT (Satisfiability)**, **which consists of all satisfiable boolean terms in Disjunctive Normalized Form (DNF)**, **is the first NP-complete problem**

The well-known SAT Problem

- Let U be a set of binary variables
- A truth assignment for U is a function t:U→{true,false}={T,F}
- If t(u)=T we say u is true
- If t(u)=F we say u is false
- If u is a variable in U, u and ┐u (not u) are literals with the following meaning: ┐u is true if and only if t(u) is false and u is true if and only if t(u) is true
- A clause over U is a set of literals which is true if and only if at least one these literals is true
- A boolean term in DNF is a collection of clauses which is true if and only if all clauses are true

➢ Cook has shown that for each non-deterministic program p and input x a boolean term can be defined which is satisfiable if and only if p accepts x. Therefore, all problems in NP can be reduced to SAT.

1.3 Dealing with NP-Completeness



Dealing with NP-Completeness

- Unfortunately, in contrast to the continuous world, we cannot provide a general solution procedure that solves all considered problems (typical integer problems) to optimality
- Therefore, we have to do research in order to
  - …analyze the respective problems in detail and attain necessary insights
  - …derive sophisticated methods for solving these problems adequately
  - …to be able to decide whether the development of an exact solution approach is reasonable

Dealing with NP-Completeness

- Exact procedures
  - Most efficient ones are constructed as **Branch&Bound procedures**. These algorithms can be characterized as enumeration methods testing all possible constellations while reducing their computational effort by using specific bounding techniques. The computation is tree-oriented while the generation process can be realized in a depth-first search as well as breadth-first search manner
  - But: The application of exact algorithms to NP-complete problems seems to be reasonable for small sized problems only

Dealing with NP-Completeness

- Alternatively, exact solution procedures are designed as recursive algorithms of a specific type
- This type does not require any knowledge about the parameter sizes of the needed subproblems building an optimal solution to the entire problem
- Therefore, all subproblems are solved and subsequently combined in order to find the best performing solution
- This technique is denoted as **Dynamic Programming**

Dealing with NP-Completeness

- Heuristics

  In order to find good but not necessarily optimal solutions, NP-complete problems are frequently solved by

  - Approximation algorithms
  - And specific heuristics, e.g.,
    - Construction procedures
    - Improvement procedures
      - Simulated Annealing
      - Tabu Search
      - Genetic algorithms
      - ....

1.4 Some basic OM definitions

- Operations Management focuses on managing the processes to produce and distribute products and services.
- Since processes are complex, sophisticated decision support is necessary. This directly addresses the application of information systems
- I.e., **specific problems** from the field of **Production and Logistic Management** are considered
- **Solutions** are designed as programmable, i.e., we provide **solution procedures**. Quality is measured by an attained objective function value

Some basic OM definitions

- **Operations Research** (in German frequently denoted as "Unternehmensforschung")
  - Development and implementation of quantitative models and methods in order to provide decision support in management
  - Instruments are: Optimization and simulation
  - Methods are applied to specific models, i.e., we map reality by mathematical models that have to be solved
  - Model structure
    - Parameters, variables, restrictions, i.e., solution space
    - Objective function, system of objective functions

Some basic OM definitions

- **Management Science**
  - Mainly used in the United States for practical application of Operations Research methods in order to provide scientific methods for management
  - Focus is set on management decisions, i.e., the support of decision makers
  - Specifically, the focus is application-oriented, i.e., how OR methods are applied to management problems and how practicable they are
  - Owing to the strong interdependencies between the development of OR methods and the specific demands of the applications, authors use the generalized combined notation OR/MS

Production and Logistics Management

- Operations Management pursues an efficient execution of Production and Logistics processes
- Thus, we have to introduce the terms
  - Production Management
  - Logistics Management
  - Management itself

  beforehand…

Management

Institutional perspective:

Persons as “carriers” of management activities

Functional perspective:

=all activities to do with

- planning
- deciding and
- the continuous control

of the activities and processes in a company

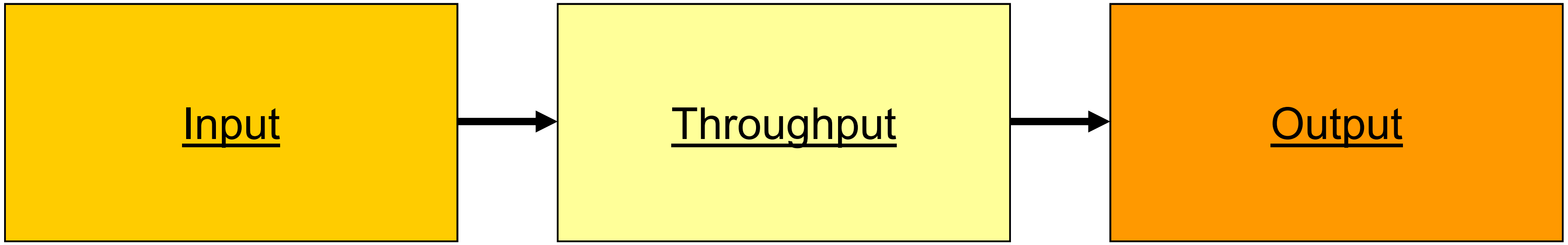
➢ Management process

Management process

1. Definition of objectives
2. Analyzing the current situation
    1. Internal
    2. External
3. Forecasting
    1. Estimating possible scenarios
    2. Qualitative forecasting
    3. Quantitative forecasting
4. Problem definition
5. Generation of existing alternatives
6. Decision making
7. Implementation
8. Continuous control

Production Management

- Production can be characterized as a transformation process
  - Transformation of the input goods in the throughput in order to generate the output goods
  - Production system is interpreted as an input-output system



Different levels of production management

- Material Resource Planning:

  Management of the supply of the necessary input goods

  (not considered in this lecture)

- Production program planning:

  Planning of the program of the offered output goods

  (not considered in this lecture)

- **Managing the production process:**

  Management of the throughput, i.e., planning and controlling the respective processes in order to attain the pursued objectives

Transformation processes

- Time transformation

  Different time assignment of the respective input and output goods
- Location transformation

  Different location assignment of the respective input and output goods
- State transformation

  Different states of the respective input and output goods

Definition Logistics Management

- A huge set of different definitions can be found in literature

- In this lecture, we use the following definition:

"Logistics Management should guarantee an efficient execution of all logistics-activities (movement of materials or movements of products or transportation-activities or transportation-services) in the considered supply chain."

- These tasks are often summarized in the so-called **mission of logistics**, which defines

  "***that the logisticians have to provide goods and services to customers according to their needs and requirements in the most efficient manner possible***."

Logistics Management systems

- Basically, three different Logistics Management systems are distinguished in literature

- Procurement Logistics Management system:
  - Procurement of needed materials and primary products (forecasting and supply strategies)
  - The planning, control and execution of the transport of materials to the production process
  - Designing the storage systems
  - Material handling, good consolidation cf. load unitization or containerization

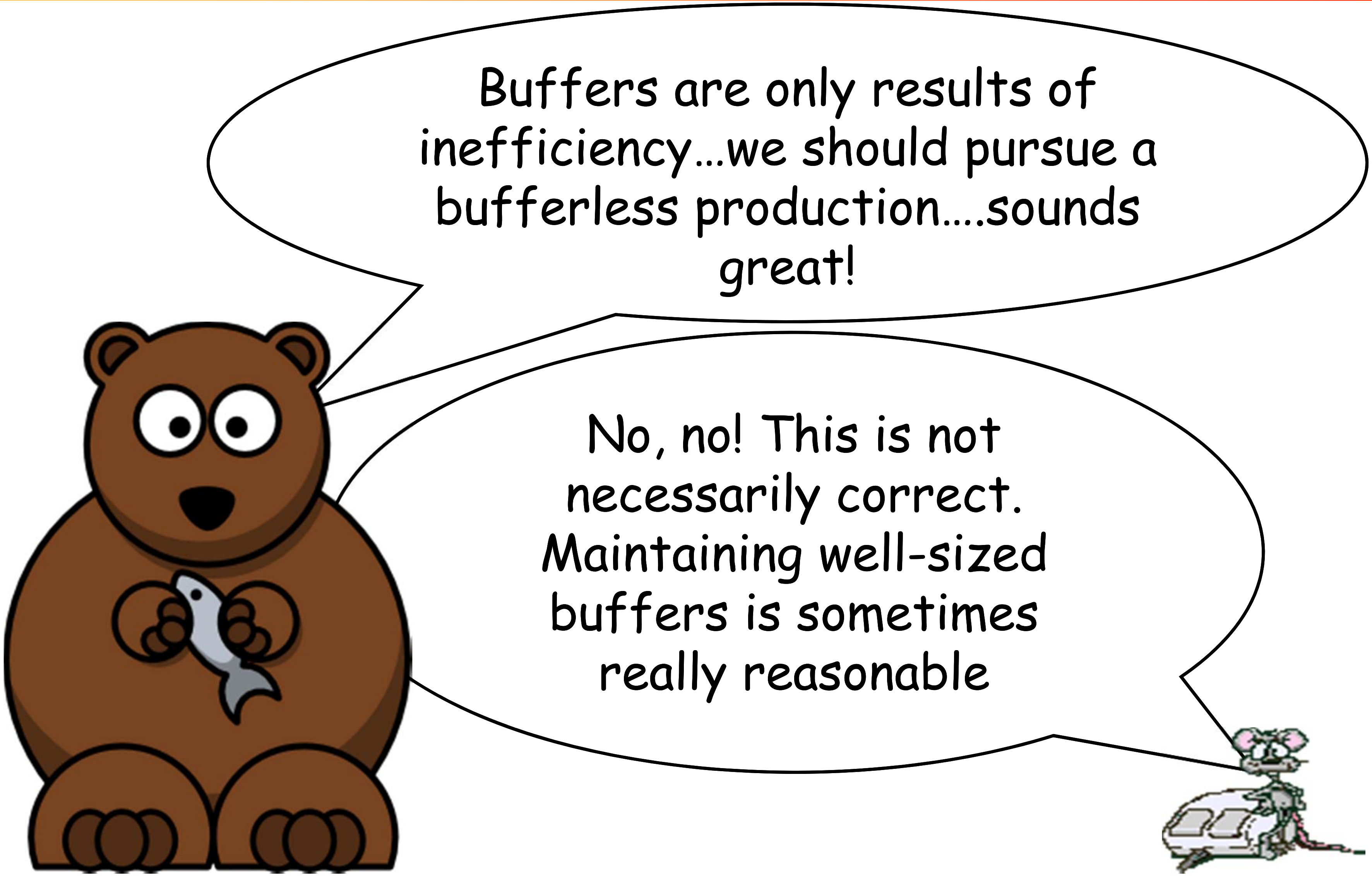
Logistics Management systems

- Production Logistics Management system
  - Movement of goods or products in the production process
  - Internal supply of the production process
  - Storage of products, goods and materials

Logistics Management systems

- Distribution Logistics Management system
  - Transportation of the products from the point of production to warehouses or resellers
  - Design of the transportation system
    - Choice of transportation mode
    - Design of the network
      - Direct / indirect transport
      - Installation of transshipment depots

Buffers – Do we really need them?



Bufferless production

- Trend towards a minimization of inventory levels can be observed (buffers are considered as a result of inefficiency)
- However, although this is not wrong, it can be also stated:
  - Procuring in small quantities leads to higher prices
  - Buffers may help bridging varying performances in connected production systems
  - Extreme problems occur if there are disturbances inside a bufferless production system. Typical disturbances can be for example
    - Machine breakdowns
    - Drop out of a worker
    - Lacking supply
    - Supplied materials are of too low quality
- Consequence: Cost consequences of inventory levels have to be adequately mapped in a mathematical model

1.5 Levels of planning

- Planning
  - Therefore, we have to anticipate realistic scenarios that may occur in future
  - Analyzing its impact on the pursued objectives
  - Estimation of the given data constellations
  - Determination of possible measures
  - All in all:

Anticipation of future scenarios

Conclusion

- **Planning level**

  = comprises all activities dealing with a still theoretical anticipation of future processes before its execution, i.e., all tasks of the planning level end with the begin of the process execution

  - ➢ necessary assumptions according to future events
  - ➢ estimated data for possible constellations

Levels of planning – Definitions

- **Long-term planning**
  - Strategic decisions that should create the prerequisites for the development of an enterprise/supply chain in the future
  - Deals with the design and structure of a supply chain with long-term impacts
- **Mid-term planning**
  - Within the scope of the strategic decisions
  - Determines an outline of the regular operations, in particular rough quantities and times for the flows and resources in the given supply chain
  - The planning horizon ranges from 6 to 24 months, enabling the consideration of seasonal developments, e.g., of demand
- **Short-term planning**
  - Specifies all activities for immediate execution and control
  - Requires the highest degree of detail and accuracy
  - The planning horizon is between a few days and three months
  - Restricted by the decisions on structure and quantitative scope from the upper levels
  - Important factor for the actual performance of the supply chain, e.g., concerning lead times, delays, customer service and other strategic issues

Real-time level

- **Real-time level**

  =comprises all activities necessary for an efficient feasible execution of the respective process, i.e., all tasks of the planning level are passed to the real-time level to ensure a feasible and – if possible – efficient execution of the process

  - receives the output of the planning level as an initial process plan
  - has to handle occurring disturbances
  - has to handle dynamic problem constellations
  - has to implement necessary plan adaptations simultaneously to the plan execution

References of Section 1

- Brucker, P.: Scheduling Algorithms. 5th edition, Springer, Berlin, Heidelberg, 2007. (**ISBN-10:** 3-5402-0524-1)
- Brucker, P.; Knust, S.: Complex Scheduling. 2nd edition, Springer, Berlin, Heidelberg, New York, 2012. (**ISBN-13:** 978-3-642-23928-1)
- Domschke, W.; Scholl, A.; Voß, S.: Produktionsplanung – Ablauforganisatorische Aspekte (in German). 2nd Edition, Springer, Berlin, 1997. (**ISBN-10:** 3-5406-3560-2)
- Fleischmann, M.: Quantitative Models in Reverse Logistics. 1st edition, Springer, Berlin, 2001. (**ISBN-10:** 3-5404-1711-7)
- Garey, M.R.; Johnson, D.S.: Computers and Intractability – A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, 1990. (**ISBN-10:** 0-7167-1045-5)
- Nahmias, S.: Production and Operations Analysis. 7th edition, Waveland Press, Long Grove Illinois, 2015. (**ISBN-10:** 1-4786-2306-9)
- Pinedo, M.L.: Planning and Scheduling in Manufacturing and Services. 2nd edition, Springer, New York, 2009. (**ISBN-10:** 1-4419-0909-5)
- Tompkins, J.A. et al.: Facilities Planning. 4th edition, Wiley, New York et al., 2010. (**ISBN-10:** 0-4704-4404-7)