Decision Support Systems

Winter Course 2018 / 2019

Prof. Dr. Stefan Bock

University of Wuppertal
Business Computing and Operations Research

Information concerning the course

- Lecture:
  - Each Monday, 10:15am-11:45am
  - Room: M12.25
  - Start: October 15th, 2018
- Lecturer: Prof. Dr. Stefan Bock
  - Office: M12.02
  - Office hour: Monday, 4pm-6pm (appointment is mandatory, email to iwuester@winfor.de)
  - Email: sbock@winfor.de

Tutorial

- Time: Tuesday, 10:15am to 11:45am
  Room: M15.09
- Start of Tutorials: October 23th, 2018
- First assignment: October 22th, 2018
- Supervisor: David Bachtenkirch
  - Contact for questions concerning weekly assignment
  - Office M12.34
  - Office hour: Tuesday 4pm-6pm or by arrangement (appointment is mandatory)
  - Email: dbachtenkirch@winfor.de

Exercises

- Sheets with tasks matching lecture‘s content
- Procedure:

Mon	Tue	Mon	Tue
10:15 a.m.: Lecture 2:00 p.m.: New assignments online	10:15 a.m.: Tutorial: Questions concerning weekly assignments	10:15 a.m.: Lecture 12:30 p.m.: Due date for assignment sheet submission	10:15 a.m.: Tutorial: Weekly assignment review

Tutorial

- New assignments available online
  - Always on Monday, approximately 2:00pm
- Assignment sheet submitting
  - Submitting in **groups of two or three appreciated**
  - Accepted via **e-mail** to supervisor in PDF format or **postbox** in room M11.25
  - **Only accepted** if names or matriculation numbers of submitters are included

Moodle course

- Weekly assignments and exercise slides
- Message boards for news and discussions concerning the course
- frequent first, before sending an email or asking for an appointment regarding content-related issues

- https://moodle2.uni-wuppertal.de/
- Course: Decision Support Systems
- Password: dssws1819

Preliminary Agenda I

1. Introduction
   1. Basic notations
   2. Planning concepts in production
   3. Objectives
   4. Problem classification
   5. Basic instruments
   6. Introduction to Complexity Theory
2. Project planning
   1. Basic definitions
   2. Analyzing the project structure
   3. Time analysis and planning
   4. Analysis of the time table flexibility

Preliminary Agenda II

3. Lot-sizing
   1. The EOQ model
   2. Extensions to multiple product cases
   3. The SLULSP model (WW model)
   4. The CLSP model
   5. The CLSP*L* Model

Preliminary Agenda III

4. Scheduling
   1. Preliminaries
   2. Single stage systems
   3. Sequencing problems with heads and tails
   4. Job-shop scheduling
   5. Flow-shop scheduling

1. Introduction

- 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 notations

- **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

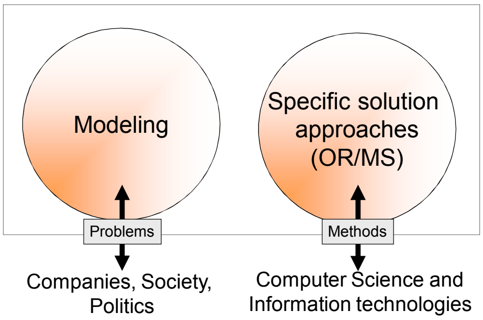
Some basic notations

- **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 manage 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

Some basic notations

- **Decision Support Systems**
  - Practical problems are complex and have to be solved under specific circumstances. Thus, not all aspects can be appropriately mapped
  - Consequently, efficient Decision Support Systems should provide a sophisticated user interface at which decision maker can interact with
  - Specifically, plans provided by OR/MS methods can be trimmed / modified by the user in order to respect, e.g., informal aspects
  - Are based on OR/MS methods
  - Make use of modern information and communication systems

Focus of the Lecture



Modeling

- Modeling and analyzing of production and logistic processes in industry
  - Specific models (stochastic, deterministic)
  - Specific scenario analyses
  - Mapping of the interdependencies
- Modeling of complete supply networks / supply chains
  - Coordinating aspects
  - Agent approaches
  - Mapping of time restrictions

General problem fields

- Location planning
- Layout planning
- Distribution
- Transportation
- Lot-size planning
- Forecasting
- Inventory management
- Project planning
- Scheduling

Production and Logistic Management

- Operations Management pursues an efficient execution of production and 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

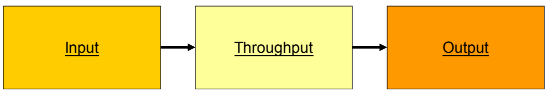
➢ Management process

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

Logistics Management – A characterization

- …it can be stated that "as far as humankind can recall, the goods that people wanted were not produced **where** they wanted to consume them or were not accessible **when** desired to consume them. Food and other commodities were widely dispersed and were available in abundance only at certain times of the year" (Ballou, 1999, p.3)……..

Logistics Management – A characterization

- That means we have different local and temporal circumstances that prevent a "free and unlimited consumption of goods and products"
- These problems need a solution that provides **local or temporal transformation processes**
- Therefore, the supply chains or networks have to cover so-called logistical functions

Logistics Management – A characterization

- Therefore, in many companies and in the scientific community "Logistics-Methods" yielded a significant increasing interest
- In literature, there are often four phases itemized that characterize the development of the understanding of what we call Logistics Management

Historical development

- 1970-1979
  - Logistics comprises the basic functions to control the material and commodity flows in the companies
  - For example, these basic functions deal with the problems of transportation, storage, stock turn activities, consolidation, and packaging
  - Logistics has to guarantee an efficient material supply of the production process
  - This phase is also known as the so-called **classical logistic**
  - Owing to this, logistics was seen in **a pure functional way**

Historical development

- 1980-1989
  - In the 80ies the "logistics understanding" was extended to a more **flow-oriented perspective**
  - This becomes necessary by the integration of the interfaces between the interacting functions of the production, procurement, and the distribution
  - Therefore, logistics becomes to a company-wide coordination instrument

Historical development

- 1990-1999
  - In this phase, the "logistics understanding" was extended to a company-wide optimization of the supply chain by using the current information in the considered flow
  - **This development was enabled by the use of improved information systems**
  - In addition to this, this phase was characterized by the **integration of the functions of research and development** and the so-called **reverse logistics**
  - Reserve logistics deals with the problems of managing the returned flows induced by different forms of reuse of products and materials (cf. Fleischmann p.6)

Reverse logistics

"Reverse logistics is the process of planning, implementing, and controlling the efficient, effective inbound flow and storage of secondary goods and related information opposite to the traditional supply chain direction for the purpose of recovering value or proper disposal"

(Fleischmann, M. (2001); page 6)

Historical development

- Since 2000
  - Today, the "logistics understanding" is extended from the consideration of separated companies to **an integrated optimization of complete supply networks**
  - Since the interactive dependencies between different companies increase due to the installation of storage reducing concepts like Just-in-Time or Just-in-Sequence, this development becomes to a crucial point for the installation of a modern Logistics Management

1.1 Selected Tasks of OM

Tasks	Problems
Lot-size planning	Number of product items continuously produced without preemption
Procurement	Number of raw materials and/or semi-finished products to be procured / Time of procurement
Process planning	Which kind of processes are applied to realize the production process?
Work distribution	Which processes / tasks are executed in which lot sizes by which production processes?
Time planning	Generating the time table for all tasks to be executed in the controlled production processes; Personnel planning
Scheduling	Generating the sequence of the different tasks at every machine

Interdependencies

- …between the decisions of the different task levels complicate the solution process considerably
- …underline that an isolated optimal solution for one task level can result in a poor constellation for the subsequent levels
- Therefore, a comprehensive planning approach has to be generated to deal with the existing interdependencies efficiently

Possible planning concepts I

- Successive/iterative planning concept
  - In this approach the total problem is divided into smaller, much simpler tasks
  - A solution to the original problem is generated iteratively by solving resulting smaller subproblems one by one in a predefined sequence

Intention:

Reduction of the total complexity. Finding a feasible production plan by the iterative generation of sub-plans. Finding good constellations for the smaller subproblems

Structure of the iterative concept

Planning	Program planning	Planning the production program (job-oriented, forecast-oriented), planning of due dates
	Material planning	Planning the demand of assemblies, components, materials by respecting the respective lead times; lot-size and order planning
	Time and resource / capacity planning	Planning of the master time table Master capacity requirement planning
	Execution preparation	Check of capacity availabilities Order release
Real-time control	Process start	Sequencing, work distribution, computation of the operative time tables (next shift), personnel employment
	Process realization	Real-time control of the production process

Main problems in iterative concepts

- Neglecting the dependencies in one direction
- Imprecise assumptions on the higher levels (e.g., cost estimations for the throughput in the production program planning level)
- “Wrong decisions” at the top levels can lead to poor constellations for the subsequent levels

Possible planning concepts II

- Simultaneous planning concept
  - Simultaneous examination of different planning levels in a single model (cf. lot-size planning and scheduling, production program planning and scheduling)
  - Computation of a combined production plan

Intention:

Respecting as much interdependencies as possible during the planning process. Finding elaborated production plans

Problems in simultaneous planning concepts

- Extreme model complexity
- Data of different planning levels are frequently not available simultaneously (cf. weekly production program and currently available capacities)
- Missing reliability of the presupposed data

Possible planning concepts III

- Hierarchical planning concept
  - Division of the total planning problem in smaller subproblems
  - Iterative processing of the smaller problems in a predefined sequence
  - No isolated consideration of the different subproblems but coordinated solution process by using specific instruments:
    - Building a solution hierarchy: Top-down approach. Restrictions from higher levels and feedbacks from lower ones. By receiving feedbacks adjustments on higher levels, it is possible to integrate existing interdependencies between the different decisions
    - Aggregation: Combining data of lower levels on higher levels (cf. group of products)
    - Rolling approach: Repeated execution of the planning process to respect possible feedbacks and, therefore, existing interdependencies

Possible planning concepts III

Intention:

Combining the advantages of iterative and simultaneous planning approaches (Complexity reduction as well as the necessary consideration of interdependencies)

Problems in hierarchical planning concepts

- Finding an unambiguous definition of the necessary interfaces between the different planning levels to support an efficient solution process
- Defining the data exchange between the different planning levels

➢ Currently, the most promising approach in practice

1.2 Objective function

- To evaluate the different production plans, we have to define a general objective function or a system of objective functions
- For an efficient realization of production processes, cost- or profit-oriented definitions of the objective functions are the most appropriate approaches (e.g., chose a production plan that enables a cost-minimizing execution of the production process)
- Examples:
  - Minimization of manufacturing costs
  - Minimization of storage costs
  - Minimization of costs for tardiness
  - Minimization of costs for unproductive machine times

Substitute objectives

- But: Frequently, cost consequences of production plans cannot be identified as accurate as necessary
- Therefore, we have to use substitute objectives as for example:
  - Minimization of lead time
  - Minimization of tardiness
  - Minimization of unproductive machine times
  - Minimization of unproductive job times

Trade-offs of objectives

- The use of multiple objective functions can result in trade-offs
- For example:
  - Minimization of inventory

  against
  - Minimization of tardiness

1.3 Classification of problems

- In what follows, we propose specific solution methods that are generated for different problems in order to deal with their specific attributes. To do so, first of all, we have to generate some criteria for classification of production processes
- Criterion 1: Output quantity
  - Small sized
  - Medium sized
  - Large sized

Small-sized output quantities

- The production program in the considered planning horizon comprises the production of one or only a few items of a very small variety of product types
- Single-product production
- The production of the single item can be interpreted as a project
- Therefore, project planning instruments are applied

Medium-sized output quantities

- The production program in the considered planning horizon comprises the production of a larger number of items of a larger variety of product types
- Batch-oriented production system
- Frequently implemented as job-shop or flow-shop systems depending on the requirements of the production programs

Large sized output quantities

- The production program in the considered planning horizon comprises the production of an extremely large number of items of a small or larger variety of product types
- The size of the production program stays large for the next planning periods
- Mass production
- Mainly supported by the use of assembly lines

Classification of the problems

- Criterion 2: Number of echelons in the production system
  - Single-echelon production systems
    Production process comprises only a single production stage. One-stage cases with parallel machines are integrated. Problems are often quite simple and can be solved optimally by appropriate algorithms
  - Multi-echelon production systems
    Production process comprises more than one production stage with or without parallel machines. These problems are often NP-hard and cannot be solved optimally in reasonable time. Therefore, specific heuristic methods are applied

Classification of the problems

- Criterion 3: Facility layout principles in the departments
  - Product planning departments / Production line departments
    - Combine all workstations which produce similar products and/or components
    - Additionally subdivided according to the characteristics of the products being produced
    - Grouping of all workstations required to produce the product →Line shape arrangement
  - Fixed materials location departments
    - Used in case of large immoveable products
    - Include all workstations required to produce the product and the respective staging area

Criterion 3: Facility layout principles

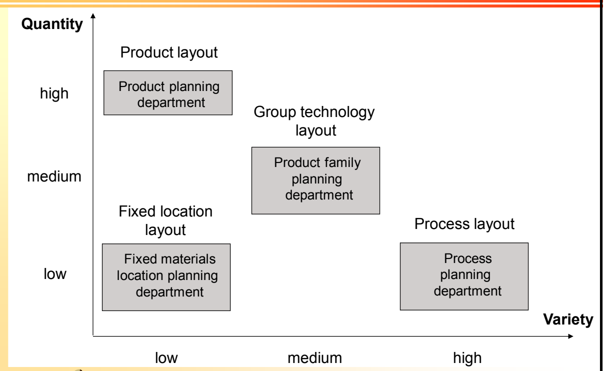
- Product family departments:
  - Grouping of workstations to produce a “family of components”
  - Combination of these groups of workstation results in a product planning department
- Process departments:
  - Combination of workstations performing similar processes
  - E.g., metal cutting departments, gear cutting departments, turning shop departments, paint shop departments …

Procedural guide for departmental planning

(Tompkins et al. p.73)

If the product is…	Department type	Method of combining
standardized and has a large stable demand	Production line, product department	Combine all workstations required to produce the product
a large product, awkward to move, and has a sporadic demand	Fixed material location, product department	Combine all workstations required to produce the product with the staging area
capable of being grouped into families of parts that may be produced by a group of workstations	Product family, product department	Combine all workstations required to produce the respective family of products
none of the above	Process department	Combine process related workstations while respecting interrelationships

Volume-variety layout classification



Layout types – Pros and Cons

Type of layout	Pros	Cons
Product layout	Easier to control High production rates Low variable costs	Small flexibility Unreliable High investments necessary Problems with large varieties Error prone
Process layout	Large flexibility Reliable	Hard to control Low production rates High variable costs
Product family layout	Job enrichment Motivation of the employees Complexity reduction	Finding product families is not always possible Loss of competence Worse performance

Classification of the problems

- Criterion 4: Degree of collaboration
  - Isolated system planning
    Consideration of isolated problems
  - Intra company collaboration
    Generation of problems resulting from planning and controlling of the company-wide supply chain
  - General Supply Chain Management
    Advanced planning problems dealing with the coordination of inter-company supply chains

1.4 Basic instruments

Hereinafter, we introduce the following instruments…

- Graphs
- Gantt diagrams
- Specific matrices
- Disjunctive graph

Graphs

1.4.1 Definition

Let Σ a finite set and $\Gamma : \Sigma \to P(\Sigma), \Gamma^{-1} : \Sigma \to P(\Sigma)$ mappings. Then, we call the tuple (Σ,Γ) directed graph. The elements of Σ are called vertices or nodes.

A vertex $i \in \Sigma$ is called **direct successor** of $j \in \Sigma$ if and only if $i \in \Gamma(j)$.

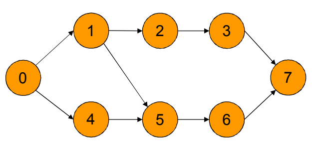
Additionally, a vertex $i \in \Sigma$ is called **direct predecessor** of $j \in \Sigma$ if and only if $i \in \Gamma^{-1}(j)$.

The tuple (i,j) with $i, j \in \Sigma$ and $j \in \Gamma(i)$ is called an **directed edge**.

Each node i with $\Gamma(i) = \varnothing$ is called a **sink**

Each node i with $\Gamma^{-1}(i) = \varnothing$ is called a **source**

Illustration



Sucessors of vertex 1:	$\Gamma(1)$	$= \{2,5\}$
Predecessors of vertex 3:	$\Gamma^{-1}(3)$	$= \{2\}$
Sources:	$\{i \mid \Gamma^{-1}(i) = \varnothing\}$	$= \{0\}$
Sinks:	$\{i \mid \Gamma(i) = \varnothing\}$	$= \{7\}$

Paths / Coherence

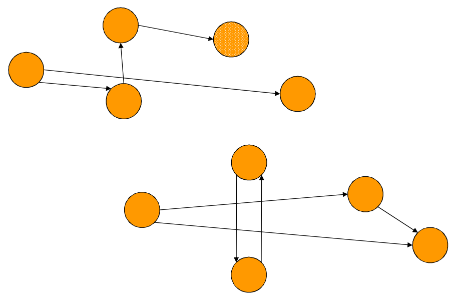
1.4.2 Definition (path)

Let $G = (\Sigma, \Gamma)$ a directed graph as defined in definition 1.4.1 and for $n \in IN$ $i_1, \ldots, i_n \in \Sigma$ vertices in G. If it holds that $\forall l \in \{1,2,\ldots,n\text{-}1\}$: $i_{l+1} \in \Gamma(i_l)$ $\langle i_1, \ldots, i_n \rangle$ is called **a path from source i_1 to destination i_n**. If additionally $i_1 = i_n$, $\langle i_1, \ldots, i_n \rangle$ is called a **cycle in G**

1.4.3 Definition (connected graph)

Let $G = (\Sigma, \Gamma)$ a directed graph as defined in definition 1.4.1. Then, G is called connected if and only if for each pair of vertices i and j a sequence of vertices $\langle i_0, \ldots, i_n \rangle$ exists $(n \geq 1)$ with: $i_0 = i$ and $i_n = j$ and $\forall c \in \{0, \ldots, n-1\} : i_c \in \Gamma^{-1}(i_{c+1})$ or $i_c \in \Gamma(i_{c+1})$

Connected graphs…?



Strongly connected graph

1.4.4 Definition (strongly connected graph)

A directed graph that has a path from each vertex to every other vertex is called **strongly connected graph**

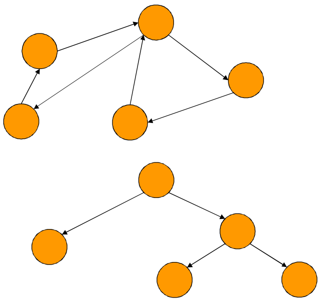
1.4.5 Definition (tree)

A directed graph G is called a **tree** if and only if:

- There is a single source i in G
- Each vertex j unequal to i possesses a single definite predecessor in G

Sinks in the tree are called **leafs**

Illustration



Weighted directed graph

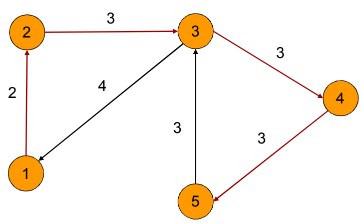
1.4.6 Definition (network)

Let $G = (\Sigma, \Gamma)$ a directed graph as defined in definition 1.4.1 and c a function that assigns a real number $c_{i,j}$ to each existing edge (i,j) in G. Then, $N = (\Sigma, \Gamma, c)$ is called a weighted directed graph or network.

Additionally, let $p = \langle (i_0, i_1), \dots, (i_{n-1}, i_n) \rangle$ a path in N. Then:

$l(p) = \sum_{v=0}^{n-1} c_{i_v, i_{v+1}}$ is the **length of the path p**.

Length of a path



Scheduling problems

(cf. Brucker pp.1)

- Suppose that M machines have to process N jobs.
- A single job is denoted by $n=1,\dots,N$ and a single machine by $m=1,\dots,M$.
- A schedule for each job is a sequenced assignment of processes executed on different machines.
- Schedules can be illustrated by Gantt charts.
- These Gantt charts may be **machine- or job-oriented**.

Job data

Each job n consists of O_n operations $o_{1,n},\dots, o_{i,n},\dots, o_{O_n,n}$.

A release date r_n may specify when the first operation of job n becomes available for processing.

Associated with each operation $o_{i,n}$ is a subset of all machines, denoted as $\Pi_{i,n} \subseteq \{1,\dots,M\}$, that can process this operation.

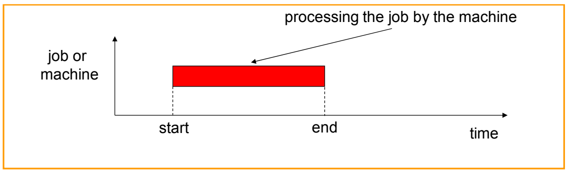
The processing of $o_{i,n}$ on machine $m \in \Pi_{i,n}$ requires $p_{m,n}$ time units.

Summary of Notations

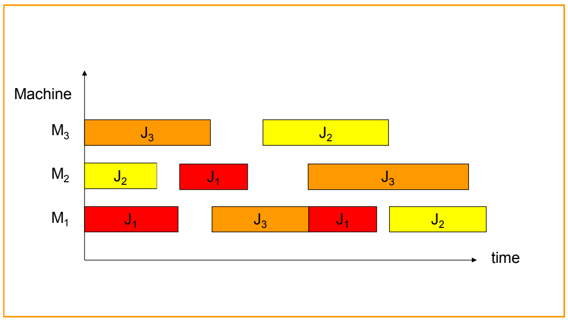
Symbol	Description
M	Number of machines
N	Number of jobs
$m=1,\dots,M$	Machine index
$n=1,\dots,N$	Job index
O_n	Number of operations of job n
$i=1,\dots,O_n$	Operation index of job n
o_{in}	i-th processing step of job n (Operation)
r_n	Release date of job n
$\Pi_{i,n}$	Set of machines assigned to operation $o_{i,n}$
$j=1,\dots,N$	Job sequence index (cf. slide 71)

Gantt charts

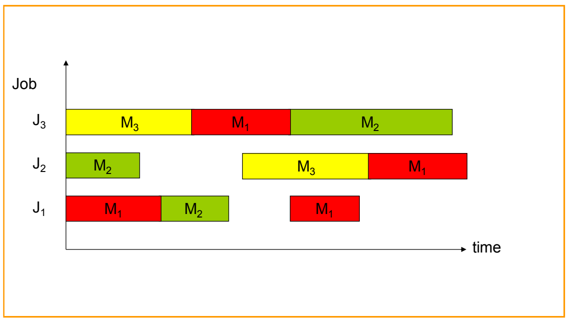
- Gantt charts are used to illustrate specific constellations in production processes
- Intention:
  - Identification of specific bottlenecks
  - Obtaining insights into the problem structure



Machine-oriented Gantt chart



Job-oriented Gantt chart



Machine restrictions

- In a job-shop environment we have precedence relations of the form:

$$o_{1n} \rightarrow o_{2n} \rightarrow o_{3n} \rightarrow \ldots \rightarrow o_{O_n,n}$$

- We distinguish problems with and without machine repetition of jobs, i.e., in the case of forbidden repetition, we require additionally:

$$\forall n \in \{1,\ldots,N\} : \forall m,k \in \{1,\ldots,o_{O_n,n}\}, m \neq k : \Pi_{m,n} \cap \Pi_{k,n} = \varnothing$$

- In the following, we assume no repetition, i.e., every job has to be processed on each machine exactly once, O_n = M for all n = 1, …, N.

Specific matrices

- In order to determine specific restrictions, as for instance precedence constraints of the operations, specific matrices are defined
- Altogether, we distinguish between three different kinds of matrices:
  - Matrix of processing times (PT)
  - Machine sequence matrix (MS)
  - Job sequence matrix (JS)
- Note that PT and MS are predefined while the definition of JS is part of the solution finding process

Matrix of processing times

$$PT = \begin{pmatrix} p_{1,1} & \cdots & p_{1,N} \\ \cdots & p_{m,n} & \cdots \\ p_{M,1} & \cdots & p_{M,N} \end{pmatrix}$$

with:

$\forall n \in \{1,\ldots,N\} : \forall m \in \{1,\ldots,M\} :\ p_{m,n} :$

Processing time of job *n* on machine *m*

Machine sequence matrix

$$MS = \begin{pmatrix} [1]_1 & \cdots & [1]_N \\ \cdots & [i]_n & \cdots \\ [M]_1 & \cdots & [M]_N \end{pmatrix}$$

↓ Interpretation in this direction

$with: \forall n \in \{1,...,N\}: \forall i \in \{1,...,M\}: \ [i]_n:$

The index of the machine that processes the operation of the *i*-th processing step of the *n*-th job. We assume that repetition is not allowed and each job is processed on each machine once in a predefined sequence.

Job sequence matrix

$$JS = \begin{pmatrix} [1]_1 & \cdots & [N]_1 \\ \cdots & [j]_m & \cdots \\ [1]_M & \cdots & [N]_M \end{pmatrix}$$

→ Interpretation in this direction

$\text{with:} \forall j \in \{1,...,N\}: \forall m \in \{1,...,M\}: \ [j]_m:$

The index of the job that is processed at the *j*-th position on machine *m*.

Example

3 jobs on 2 machines

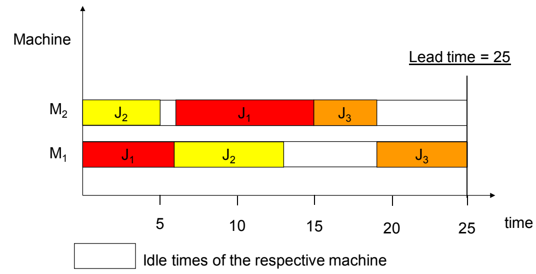
$$PT = \begin{pmatrix} 6 & 7 & 6 \\ 9 & 5 & 4 \end{pmatrix} \quad MS = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 1 \end{pmatrix}$$

The following job sequence matrix has been generated:

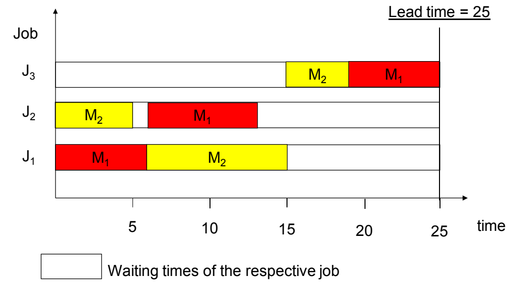
$$JS = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$$

Generate the machine- and job-oriented Gantt.

Machine-oriented Gantt chart



Job-oriented Gantt chart



Gantt charts – Pros and Cons

- Pros:
  - Simple instrument, easy to generate
  - Very demonstrative illustration
  - Supports the search for improvements
  - Illustrates existing bottlenecks
- Cons:
  - Complete regeneration for each modification
  - No solution process instrument

Adjacency matrix

Definition 1.4.7

Let $G=(\Sigma,\Gamma)$ be a directed graph with $\Sigma=\{1,...,l\}$.

The matrix $A(G)=(a_{i,j})_{1\le i,j\le l}\in\{0,1\}^{l\times l}$ with:

$$a_{i,j}=\begin{cases}1 & \text{if } j\in\Gamma(i)\\ 0 & \text{otherwise}\end{cases}$$

is called an **adjacency matrix of G**.

Distance matrix

Definition 1.4.8

Let $N=(\Sigma,\Gamma,c)$ be a network with $\Sigma=\{1,...,l\}$.

The matrix $K(N)=(k_{i,j})_{1\le i,j\le l}\in IR^{l\times l}$

with: $$k_{i,j}=\begin{cases}c_{i,j} & \text{if } j\in\Gamma(i) \text{ and } i\neq j\\ 0 & \text{if } i=j\\ \infty & \text{otherwise}\end{cases}$$

is called **distance matrix of G**.

Disjunctive graphs

- Disjunctive graphs are used to define general shop problems. These graphs can map arbitrary feasible solutions to be implemented electronically.
- For a given instance of a general job-shop problem, the disjunctive graph G=(V,C,D) is defined as follows:
  - V is the set of nodes defining all operations of each job to be executed. Additionally, there are two special nodes, a source 0 and a sink s belonging to V. While all operation nodes have a weight equal to their processing time, these additional nodes have the weight 0.
  - C is the set of directed conjunctive arcs. These arcs reflect the precedence relations between the operations. Additionally, there are conjunctive arcs to guarantee that the source is indirect predecessor and the sink is indirect successor of all other operation nodes.

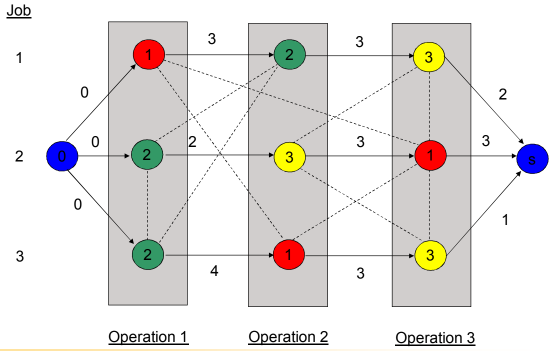
Disjunctive graphs

- D is the set of all undirected arcs connecting each pair of operations to be executed on one machine. These operations compete for this machine, wherefore the arcs should determine a possible ordering.

An example

$$MS = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 3 & 1 \\ 3 & 1 & 3 \end{pmatrix}; PT = \begin{pmatrix} 3 & 3 & 3 \\ 3 & 2 & 4 \\ 2 & 3 & 1 \end{pmatrix}$$

Disjunctive graph



Interpretation

- In order to define a schedule, each disjunctive arc has to be oriented
- Question: What about cycles? How do you interpret a schedule comprising a cycle in its oriented disjunctive graph?
- We will focus on those questions in section 4

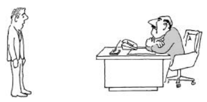
1.5 Basics of computational complexity theory

- 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 only be generated 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 NP-completeness theory
- Basic results of this theory will be referenced throughout this lecture

Motivation of computational complexity 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.

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

Reference: Garey and Johnson 1979

Basic definitions

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

- The respective program generates h(x) for each input x
- To compare the performance of different procedures rated by the caused 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 concentrate on finding appropriate **performance or complexity classes** a specific run-time function belongs to. 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 2 numbers
logarithmic	$O(\log n)$	Binary search
linear	$O(n)$	Sum of n numbers
n-log-n	$O(n \log n)$	Sorting n numbers (heap sort)
quadratic	$O(n^2)$	Wagner-Whitin 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 1.5.1

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 1.5.2

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 $f(S) \leq k$?

The classes P and NP

Definition 1.5.3 (The class P):

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

Definition 1.5.4 (The class NP):

The class of all decision problems,
where each input x with an yes-output
- has a certificate y, such that $|y|$ is bounded by a polynomial in $|x|$ and
- there is a polynomial time algorithm to verify that y is a valid certificate for x,

is denoted as NP.

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 an 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 nondeterministic 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 started 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 we can solve efficiently. But about 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: $P \subseteq NP$

But:

- 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 **NP-completeness theory** and is discussed subsequently.

The Theory of NP-completeness

- 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 function g that is computable in polynomial time and 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**
  - Using reduction in order to prove the hardness of a considered problem
    - 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

The Theory of NP-completeness

  - 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, no otherwise

  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

NP-hardness, *NP*-completeness

1.5.5 Definition:

A problem P is called ***NP*-hard** if all problems belonging to *NP* can be reduced to P

1.5.6 Definition:

A problem P is called ***NP*-complete** if P is *NP*-hard and P belongs additionally 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 and false otherwise
- If u is a variable in U, u and ┐u (not u) are literals, while not 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 literal belonging to it 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 nondeterministic 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.

The Hamiltonian circuit problem

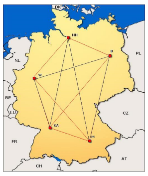
Let G=(V,Γ) be an undirected graph with |V| = n vertices.
Question (Decision): Does G contain a Hamiltonian circuit, i.e., a path $\langle(v_1,v_2),\ldots,(v_{n-1},v_n)\rangle$ of vertices, such that $(v_n,v_1)\in\Gamma$ and $(v_i,v_{i+1})\in\Gamma\ \forall i=1,\ldots,n-1$?

The Hamiltonian circuit problem (HC) has been proven NP-complete, cf. Garey, Johnson (1979), page 56.

The problem plays an important role for OR, because the decision version of the Traveling Salesperson Problem can be transformed to (HC).

Application of HC: Traveling Salesperson Problem

- 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, of all cities in C with minimal length!



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

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

- 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
      - ….

References for 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. 1st edition, W.H. Freeman and Company, San Francisco, 1979. (**ISBN-10:** 0-7167-1045-5)
- Nahmias, S.: Production and Operations Analysis. 6th edition, Irwin, Chicago et al., 2008. (**ISBN-10:** 0-0712-6370-5)
- Pinedo, M.L.: Scheduling: Theory, Algorithms and Systems. 4th edition, Prentice Hall, New Jersey, 2012. (**ISBN-10:** 1-4614-1986-7)
- 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., 2009. (**ISBN-10:** 0-4704-4404-5)