Production Analysis
KFK – Christian Almeder
Organization – Textbooks

Textbooks


• **Planning and Scheduling in Manufacturing and Services** by Pinedo, Springer, 2005
Organization – Course Content

- Introduction to production systems
- Aggregate Planning
- Capacity and Material Planning
- Lotsizing
- Scheduling
0. The Production Paradigm: Evolution of Production Systems

Ancient Systems
- basic planning, organizations and control
- specialization of labor

Feudal Systems
- hierarchical system (delegation)
- land and labor as production input

European System
- double entry bookkeeping, cost accounting

Industrial Revolution: specialization, mass markets, mass production

American System
- interchangeable parts
- steam power
- assembly lines
0.1 The Competitive Environment

Status Quo of the American (and European) System (late 80s):
- production driven system
- cost efficient production as the main goal
- high quality standardized goods
- Market is taken as given

Change towards a market-driven system
- more sophisticated consumers
- short product life cycles
- product variety increases
- global competition and heterogeneous markets
0.2 Production Systems

Input → Output
manufacturing firms
service companies

flow process in two parts:
  physical material
  information

coordination also with suppliers and distributors: supply chain management: recent emphasis on bi-directional information flow
0.2 Production Systems

Production Management

Supplier → Raw Material Inventory → Production Floor → Finished Goods Inventory → Customer

Inventory Management

Production System

Production Floor

Work-in-process

Purchasing

Forecasting

Long-range capacity planning
Production planning
Short-range requirements (material capacity)
Scheduling

Cost Estimation and Quality Control

Production Management
1. Advanced Planning [Fleischmann, Meyr, Wagner, 2008]

1.1 What is planning?

• along a supply chain hundreds and thousands of individual decisions have to be made and coordinated.

• decisions are of different importance
  – which job has to be scheduled next on a respective machine
  – open or close a factory
  → the more important a decision is the better it has to be prepared.
  – preparation is the job of planning.
Planning can be divided into the following phases

• recognition and analysis of a decision problem
• definition of objectives
• forecasting of future developments
• identification and evaluation of feasible activities (solutions) and
• selection of good solutions

Supply chains are very complex. → abstraction from reality

Model

the art of model building – represent the reality as simple as possible but as detailed as necessary
Models

• **Forecasting and simulation models**
  – to predict future developments
  – explain relationships between input and output of complex systems
  – selection of good solutions is ignored

• **Optimization models → additional objective function**
  – objective function has to be minimized or maximized

• **The validity of a plan is restricted to a predefined planning horizon.**
Planning horizon

**Long-term planning:** strategic decisions → they concern the design and structure of a supply chain and have long term effects

**Mid-term planning** (also known as tactical decisions): rough quantities and times for the flows and resources in the given supply chain. The planning horizon ranges from 6 to 24 months→ considering seasonal developments e.g. of the demand.

**Short-term planning** (also known as operational decisions): lowest planning level – all activities as detailed instructions for immediate execution and control. Short-term planning models require the highest degree of detail and accuracy. Planning horizon is between a few days and three months. Short-term planning is restricted by the decisions on structure and quantitative scope from the upper levels. Nevertheless it is an important factor of the production system or the supply chain → concerning lead time, delays, customer service and other strategic issues.
Difficulties in planning!

Objectives: finding the appropriate objective function
- conflicting objectives
  - high service level
  - low costs
→ multi-objective decision situation

→ set a minimum/maximum satisfaction level for each objective except for one that will be optimized

→ pricing all objectives monetarily by revenues or costs and optimize an aggregated objective functions (weights?)
  → customer service in monetary units is difficult.
→ Advanced Planning System (e.g. SAP) supports all these procedures in principle.
Difficulties in planning!

large number of alternatives, e.g.
• order sizes,
• starting times of jobs,
• sequence of several jobs on a machine
  – enumeration is impossible
  – heuristics and metaheuristics
• hardest difficulty is dealing with uncertainty
  – planning anticipates future activities and is based on data about future developments. Data may be estimated by forecast models
  – forecast errors occur
  – Rolling horizon approach +
  – Dynamic reoptimization +
Characteristics of APS

- Integral planning
- True optimization
  - heuristics
  - exact approaches
- A hierarchical planning system
Supply Chain Planning Matrix

Procurement → Production → Distribution → Sales

Strategic Network Planning

Master Planning

Material Requirements Planning

Production Planning

Distribution Planning

Scheduling

Transport Planning

Demand Planning

Demand Fulfilment & ATP

Long-term

Mid-term

Short-term
Supply Chain Planning Matrix

procurement → production → distribution → sales

long-term
× materials program
× supplier selection
× cooperations

mid-term
× personnel planning
× material requ. planning
× contracts

short-term
× personnel planning
× ordering materials

flow of goods

information flows

× plant location
× production system

× physical distribution structure
× product program
× strategic sales planning

× master production scheduling
× capacity planning

× distribution planning

× mid-term sales planning

× lot-sizing
× machine scheduling
× shop floor control

× warehouse replenishment
× transport planning

× short-term sales planning
Aggregate Planning

Example:

- one product (plastic case)
- two injection molding machines, 550 parts/hour
- one worker, 55 parts/hour
- steady sales 80,000 cases/month
- 4 weeks/month, 5 days/week, 8h/day
- how many workers?

In real life constant demand is rare
- change demand
- produce a constant rate anyway
- vary production
Aggregate Planning

Influencing demand
- do not satisfy demand
- shift demand from peak periods to nonpeak periods
- produce several products with peak demand in different period

Planning Production
- Production plan: how much and when to make each product
- rolling planning horizon
- long range plan
- intermediate-range plan
  - units of measurements are aggregates
  - product family
  - plant department
  - changes in workforce, additional machines, subcontracting, overtime,...

Short-term plan

Aspects of Aggregate Planning
- Capacity
- Aggregate Units
- Costs
Aggregate Planning

Aspects of Aggregate Planning

Capacity: how much a production system can make

Aggregate Units: products, workers,...

Costs

production costs (economic costs!)
inventory costs (holding and shortage)
capacity change costs
Aggregate Planning

Spreadsheet Methods

Zero Inventory Plan

Precision Transfer, Inc. Produces more than 300 different precision gears (the aggregation unit is a gear!).

Last year (=260 working days) Precision made 41.383 gears of various kinds with an average of 40 workers.

41.383 gears per year

\[
40 \times 260 \text{ worker-days/year} = 3,98 \rightarrow 4 \text{ gears/ worker-day}
\]

<table>
<thead>
<tr>
<th>Month</th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand</td>
<td>2760</td>
<td>3320</td>
<td>3970</td>
<td>3540</td>
<td>3180</td>
<td>2900</td>
<td>19,670</td>
</tr>
</tbody>
</table>

Aggregate demand forecast for precision gear:
Aggregate Planning

holding costs: $5 per gear per month
backlog costs: $15 per gear per month
hiring costs: $450 per worker
lay-off costs: $600 per worker
wages: $15 per hour (all workers are paid for 8 hours per day)
there are currently 35 workers at Precision
currently no inventory

Production plan?
Aggregate Planning

Zero Inventory Plan

produce exactly amount needed per period
adapt workforce
Aggregate Planning

Production Management
Aggregate Planning

Level Work Force Plan

backorders allowed
constant numbers of workers
demand over the planning horizon
gears a worker can produce over the horizon

19670/(4x129) = 38.12 \rightarrow 39 \text{ workers are always needed}
Aggregate Planning

Inventory: January: $3276 - 2760 = 516$

February: $516 + 3120 - 3320$

March: $316 + 3588 - 3670 = -66!$ - Backorders: $66 \times \$15$

= $990
Aggregate Planning

no backorders are allowed

\[
\text{workers} = \frac{\text{cumulative demand}}{\text{cumulative days} \times \frac{\text{units}}{\text{workers/day}}}
\]

January: \(\frac{2760}{(21 \times 4)} = 32.86\) -> 33 workers
February: \(\frac{(2760+3320)}{[(21+20) \times 4]} = 37.07\) -> 38 workers.
March: \(\frac{10050}{(64 \times 4)} \rightarrow 40\) workers
April: \(\frac{13590}{(85 \times 4)} \rightarrow 40\) workers
May: \(\frac{16770}{(107 \times 4)} \rightarrow 40\) workers
June: \(\frac{19670}{(129 \times 4)} \rightarrow 39\) workers
Aggregate Planning

Example Mixed Plan

The number of workers used is an educated guess based on the zero inventory and level work force plans!
## Spreadsheet Methods Summary

<table>
<thead>
<tr>
<th></th>
<th>Zero-Inv.</th>
<th>Level/BO</th>
<th>Level/No BO</th>
<th>Mixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hiring cost</td>
<td>4950</td>
<td>1800</td>
<td>2250</td>
<td>3150</td>
</tr>
<tr>
<td>Lay-off cost</td>
<td>7800</td>
<td>0</td>
<td>0</td>
<td>4200</td>
</tr>
<tr>
<td>Labor cost</td>
<td>59856</td>
<td>603720</td>
<td>619200</td>
<td>593520</td>
</tr>
<tr>
<td>Holding cost</td>
<td>0</td>
<td>4160</td>
<td>6350</td>
<td>3890</td>
</tr>
<tr>
<td>BO cost</td>
<td>0</td>
<td>7110</td>
<td>0</td>
<td>990</td>
</tr>
<tr>
<td>Total cost</td>
<td>611310</td>
<td>616790</td>
<td>627800</td>
<td>605180</td>
</tr>
<tr>
<td>Workers</td>
<td>33</td>
<td>39</td>
<td>40</td>
<td>35</td>
</tr>
</tbody>
</table>
Aggregate Planning

Linear Programming Approaches to Aggregate Planning

Parameters:

T... Planning horizon length

t ... Index of periods, t=1,2,..., T

\(D_t\) … forecasted number of units demanded in period t

\(n_t\) … number of units that can be made by one worker in period t

\(C_t^p\) … cost to produce one unit in period t

\(C_t^w\) … cost of one worker in period t
Aggregate Planning

\[ C^H_t \] … cost to hire one worker in period \( t \)

\[ C^L_t \] … cost to lay off one worker in period \( t \)

\[ C^I_t \] … cost to hold one unit in inventory in period \( t \)

\[ C^B_t \] … cost to backorder one unit in period \( t \)
Aggregate Planning

Decision Variables:

- $P_t$ ... number of units produced in period $t$
- $W_t$ ... number of workers available in period $t$
- $H_t$ ... number of workers hired in period $t$
- $L_t$ ... number of workers laid off in period $t$
- $I_t$ ... number of units held in inventory in period $t$
- $B_t$ ... number of units backordered in period $t$
Aggregate Planning
Constraints: work, Capacity, force, material

\[ P_t \leq n_t W_t \quad t = 1, 2, \ldots, T \]

\[ W_t = W_{t-1} + H_t - L_t \quad t = 1, 2, \ldots, T \]

net inventory this period = net inventory last period +
production this period - demand this period

\[ I_t - B_t = I_{t-1} - B_{t-1} + P_t - D_t \]

Costs

\[ \sum_{t=1}^{T} (C_t^P P_t + C_t^W W_t + C_t^H H_t + C_t^L L_t + C_t^I I_t + C_t^B B_t) \]
Aggregate Planning

Example: Precision Transfer

Planning horizon: 6 months $T = 6$

Costs do not vary over time $C_t^P = 0$

$d_t$: days in month $t$

$C_t^W = 120d_t$

$C_t^H = 450$

$C_t^L = 600$

$C_t^I = 5$

We assume that no backorders are allowed!

no production costs and no backorder costs are included!

Demand

<table>
<thead>
<tr>
<th></th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2760</td>
<td>3320</td>
<td>3970</td>
<td>3540</td>
<td>3180</td>
<td>2900</td>
<td>19,670</td>
</tr>
</tbody>
</table>

Production Management
Linear Program Model for Precision Transfer

Minimize

\[ 2520W_1 + 2400W_2 + 2760W_3 + 2520W_4 + 2640W_5 + 2640W_6 \]
\[ + 450(H_1 + H_2 + H_3 + H_4 + H_5 + H_6) \]
\[ + 600(L_1 + L_2 + L_3 + L_4 + L_5 + L_6) \]
\[ + 5(I_1 + I_2 + I_3 + I_4 + I_5 + I_6) \]

subject to

(Production-capacity constraints)

\[ P_1 \leq 84W_1, \quad P_2 \leq 80W_2, \quad P_3 \leq 92W_3, \quad P_4 \leq 84W_4, \quad P_5 \leq 88W_5, \quad P_6 \leq 88W_6, \]

(Work-force constraints)

\[ W_1 = 35 + H_1 - L_1, \quad W_2 = W_1 + H_2 - L_2, \quad W_3 = W_2 + H_3 - L_3, \]
\[ W_4 = W_3 + H_4 - L_4, \quad W_5 = W_4 + H_5 - L_5, \quad W_6 = W_5 + H_6 - L_6, \]

(Inventory-balance constraints)

\[ I_1 = P_1 - 2760, \quad I_2 = I_1 + P_2 - 3320, \quad I_3 = I_2 + P_3 - 3970, \]
\[ I_4 = I_3 + P_4 - 3540, \quad I_5 = I_4 + P_5 - 3180, \quad I_6 = I_5 + P_6 - 2900 (\leq 0), \]

(Non-negativity constraints)

\[ P_1, P_2, P_3, P_4, P_5, P_6, W_1, W_2, W_3, W_4, W_5, W_6, H_1, H_2, H_3, H_4, H_5, H_6, \]
\[ L_1, L_2, L_3, L_4, L_5, L_6, I_1, I_2, I_3, I_4, I_5, I_6 \geq 0 \]
# Aggregate Planning

**LP solution (total cost = $600 191,60)**

<table>
<thead>
<tr>
<th></th>
<th>Production</th>
<th>Inventory</th>
<th>Hired</th>
<th>Laid off</th>
<th>Workers</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>2940,00</td>
<td>180,00</td>
<td>0,00</td>
<td>0,00</td>
<td>35,00</td>
</tr>
<tr>
<td>February</td>
<td>3232,86</td>
<td>92,86</td>
<td>5,41</td>
<td>0,00</td>
<td>40,41</td>
</tr>
<tr>
<td>March</td>
<td>3877,14</td>
<td>0,00</td>
<td>1,73</td>
<td>0,00</td>
<td>42,14</td>
</tr>
<tr>
<td>April</td>
<td>3540,00</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>42,14</td>
</tr>
<tr>
<td>May</td>
<td>3180,00</td>
<td>0,00</td>
<td>0,00</td>
<td>6,01</td>
<td>36,14</td>
</tr>
<tr>
<td>June</td>
<td>2900,00</td>
<td>0,00</td>
<td>0,00</td>
<td>3,18</td>
<td>32,95</td>
</tr>
</tbody>
</table>
# Aggregate Planning

## Rounding LP solution

<table>
<thead>
<tr>
<th></th>
<th>January</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Days</td>
<td>21</td>
<td>20</td>
<td>23</td>
<td>21</td>
<td>22</td>
<td>22</td>
<td>129</td>
</tr>
<tr>
<td>Units/Worker</td>
<td>84</td>
<td>80</td>
<td>92</td>
<td>84</td>
<td>88</td>
<td>88</td>
<td>516</td>
</tr>
<tr>
<td>Demand</td>
<td>2760</td>
<td>3320</td>
<td>3970</td>
<td>3540</td>
<td>3180</td>
<td>2900</td>
<td>19670</td>
</tr>
<tr>
<td>Workers</td>
<td>35</td>
<td>41</td>
<td>42</td>
<td>42</td>
<td>36</td>
<td>33</td>
<td>229</td>
</tr>
<tr>
<td>Capacity</td>
<td>2940</td>
<td>3280</td>
<td>3864</td>
<td>3528</td>
<td>3168</td>
<td>2904</td>
<td>19684</td>
</tr>
<tr>
<td>Capacity - Demand</td>
<td>180</td>
<td>-40</td>
<td>-106</td>
<td>-12</td>
<td>-12</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>Cumulative Difference</td>
<td>180</td>
<td>140</td>
<td>34</td>
<td>22</td>
<td>10</td>
<td>14</td>
<td>400</td>
</tr>
<tr>
<td>Produced</td>
<td>2930</td>
<td>3280</td>
<td>3864</td>
<td>3528</td>
<td>3168</td>
<td>2900</td>
<td>19670</td>
</tr>
<tr>
<td>Net inventory</td>
<td>170</td>
<td>130</td>
<td>24</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>336</td>
</tr>
<tr>
<td>Hired</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Laid Off</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>Costs</td>
<td>89050</td>
<td>101750</td>
<td>116490</td>
<td>105900</td>
<td>98640</td>
<td>88920</td>
<td>600750</td>
</tr>
</tbody>
</table>

Production Management
Aggregate Planning

Practical Issues
100,000 variables and 40,000 constraints
LP/MIP Solvers: CPLEX, XPRESS-MP, ...

Extensions
Bounds
\[ I_t \leq I_t^U \]
\[ I_t^L \leq I_t \leq I_t^U \]
\[ L_t \leq 0.05W_t \]

Training
\[ W_t = W_{t-1} + H_{t-1} - L_t \]
Aggregate Planning

Transportation Models

supply points: periods, initial inventory

demand points: periods, excess demand, final inventory

\( n_t W_t \) = capacity during period \( t \)

\( D_t \) = forecasted number of units demanded in period \( t \)

\( C_t^P \) = the cost to produce one unit in period \( t \)

\( C_t^I \) = the cost to hold one unit in inventory in period \( t \)
Aggregate Planning

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>capacity $n_t W_t$</td>
<td>350</td>
<td>300</td>
<td>350</td>
</tr>
<tr>
<td>demand</td>
<td>200</td>
<td>300</td>
<td>400</td>
</tr>
<tr>
<td>production costs</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>holding costs</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

initial inventory: 50
final inventory: 75
## Aggregate Planning

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Ending Inventory</th>
<th>Excess Capacity</th>
<th>Available Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Beginning inventory</strong></td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Period 1</strong></td>
<td>150</td>
<td>10</td>
<td>12</td>
<td>50</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td><strong>Demand</strong></td>
<td>200</td>
<td>300</td>
<td>400</td>
<td>75</td>
<td>75</td>
<td>1050</td>
</tr>
<tr>
<td><strong>Period 2</strong></td>
<td>-</td>
<td>300</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td><strong>Period 3</strong></td>
<td>-</td>
<td>-</td>
<td>350</td>
<td>12</td>
<td>14</td>
<td>0</td>
</tr>
</tbody>
</table>
Aggregate Planning

Extension:

<table>
<thead>
<tr>
<th></th>
<th>t 1</th>
<th>t 2</th>
<th>t 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>capacity $n_tW_t$</td>
<td>350</td>
<td>350</td>
<td>300</td>
</tr>
<tr>
<td>demand</td>
<td>400</td>
<td>300</td>
<td>400</td>
</tr>
<tr>
<td>production costs</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>holding costs</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**overtime**: overtime capacity is 90, 90 and 75 in period 1, 2 and 3; overtime costs are $16, $18 and $20 for the three periods respectively;

**backorders**: units can be backordered at a cost of $5 per unit-month; production in period 2 can be used to satisfy demand in period 1
## Aggregate Planning

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Ending</th>
<th>Excess</th>
<th>Available</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>invent</td>
<td></td>
<td></td>
<td>inventory</td>
<td></td>
<td>capacity</td>
</tr>
<tr>
<td><strong>Beginning</strong></td>
<td>inventory</td>
<td></td>
<td>invent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Period 1</strong></td>
<td></td>
<td>invent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regular time</td>
<td>350</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>Overtime</td>
<td>50</td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>40</td>
</tr>
<tr>
<td><strong>Period 2</strong></td>
<td></td>
<td>invent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regular time</td>
<td>16</td>
<td>11</td>
<td>13</td>
<td>75</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Overtime</td>
<td>23</td>
<td>18</td>
<td>20</td>
<td>90</td>
<td>22</td>
<td>90</td>
</tr>
<tr>
<td><strong>Period 3</strong></td>
<td></td>
<td>invent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regular time</td>
<td>22</td>
<td>17</td>
<td>12</td>
<td>12</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>Overtime</td>
<td>30</td>
<td>25</td>
<td>20</td>
<td>75</td>
<td>22</td>
<td>75</td>
</tr>
<tr>
<td><strong>Demand</strong></td>
<td>400</td>
<td>300</td>
<td>400</td>
<td>75</td>
<td>130</td>
<td>1305</td>
</tr>
</tbody>
</table>

*Production Management*
Transportation Problem: Model and LP-Formulation

**Situation:**
location of customer and warehouses are fixed, the warehouses have fixed capacities

e**example:**
company with 3 factories and 4 customers
transportation costs per item from factory i to customer j
demand quantity equals production quantity

<table>
<thead>
<tr>
<th>Sources (Factories)</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>10</td>
<td>5</td>
<td>6</td>
<td>11</td>
<td>25</td>
</tr>
<tr>
<td>F2</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>25</td>
</tr>
<tr>
<td>F3</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>Demand</td>
<td>15</td>
<td>20</td>
<td>30</td>
<td>35</td>
<td>100</td>
</tr>
</tbody>
</table>

Production Management
General formulation

m producers with the supply $s_i$, $i = 1, \ldots, m$

n customers with demand $d_j$, $j = 1, \ldots, n$

transportation costs $c_{ij}$ per item from $i$ to $j$, $i = 1, \ldots, m$; $j = 1, \ldots, n$

LP-Formulierung:

: transported quantity $x_{ij}$ von $i$ nach $j$

transportation costs


\[ K = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \rightarrow \min \]

supply

\[ s_i = \sum_{j=1}^{n} x_{ij} \quad i = 1, \ldots, m \]

demand

\[ d_j = \sum_{i=1}^{m} x_{ij} \quad j = 1, \ldots, n \]

non-negativity

\[ x_{ij} \geq 0 \quad i = 1, \ldots, m; j = 1, \ldots, n \]

\[ \sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j = M \]
example above

\[ K = \left(10x_{11} + 5x_{12} + 6x_{13} + 11x_{14}\right) + \left(x_{21} + 2x_{22} + 7x_{23} + 4x_{24}\right) + \left(9x_{31} + x_{32} + 4x_{33} + 8x_{34}\right) \rightarrow \text{min} \]

Supply constraints

\[
\begin{align*}
X_{11} + x_{12} + x_{13} + x_{14} &= 25 \quad (i=1) \\
x_{21} + x_{22} + x_{23} + x_{24} &= 25 \quad (i=2) \\
x_{31} + x_{32} + x_{33} + x_{34} &= 50 \quad (i=3) 
\end{align*}
\]

Demand constraints

\[
\begin{align*}
x_{11} + x_{21} + x_{31} &= 15 \quad (j=1) \\
x_{12} + x_{22} + x_{32} &= 20 \quad (j=2) \\
x_{13} + x_{23} + x_{33} &= 30 \quad (j=3) \\
x_{14} + x_{24} + x_{34} &= 35 \quad (j=4)
\end{align*}
\]

Nonnegativity:

\[ x_{ij} \geq 0 \quad \text{für} \quad i = 1, \ldots, 3; \quad j = 1, \ldots, 4 \]
Northwest Corner Rule

Starting algorithm – base solution

1.) Start with the following table and fill it starting with the upper left corner (northwest corner) and proceed to the lower right corner.

2.) Select the maximum value, such that the remaining resource of the column or row is used; if the resource of the row is used go down, if the resource of the column is used go right.

3.) If there is only one column or row left → choose all free \( x_{ij} \) of that row or column as base variable (BV) with the maximum possible values → otherwise proceed with 2.)

Result:

feasible solution (production = demand)

exactly \( m + n - 1 \) base variables \( x_{ij} \)

the remaining \( m \times n - (m+n-1) \) variables must be set to 0 (NBV)
**Example:** Starting solution for above problem:

\[
\begin{array}{cccc|c}
  i \backslash j & 1 & 2 & 3 & 4 & s_i \\
  \hline
  1 & 15 & 10 & \text{-} & \text{-} & 25 \\
  2 & \text{-} & 10 & 15 & \text{-} & 25 \\
  3 & \text{-} & \text{-} & 15 & 35 & 50 \\
  d_j & 15 & 20 & 30 & 35 & 100 \\
\end{array}
\]

more consecutive right shifts are possible:

\[
\begin{array}{cccc|c}
  i \backslash j & 1 & 2 & 3 & 4 & s_i \\
  \hline
  1 & 15 & 10 & 5 & \text{-} & 30 \\
  2 & \text{-} & \text{-} & 20 & \text{-} & 20 \\
  3 & \text{-} & \text{-} & \text{-} & 25 & 35 \\
  d_j & 15 & 10 & 35 & 25 & 85 \\
\end{array}
\]

Production Management
Degeneration is possible (one or more base variable are 0). DON'T FORGET THEM!

Here we might delete the row and the column. We can only delete one (chose randomly).

<table>
<thead>
<tr>
<th>i\j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>s_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>15</td>
<td>0</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>30</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>d_j</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>20</td>
<td>80</td>
</tr>
</tbody>
</table>

Advantage: fast and simple

Disadvantage: cost factors are ignored; usual bad starting solution
Vogel’s Approximation

1.) Starting with the same table as for the NW-Corner-Rule. No row or column is delete.

2.) In each non-deleted row or column compute the difference between the smallest and the second smallest cost factor $c_{ij}$ not deleted yet.

3.) Chose the row or column with the biggest difference, select the smallest $c_{ij}$ and increase the according $x_{ij}$ as much as possible.

4.) If the resource of the column is used, delete the column,

   OR

   if the resource of the row is used, delete the row.

5.) If there is only one column or row left → choose all free xij of that row or column as base variable (BV) with the maximum possible values → otherwise procede with 2.)
Example: Starting solution for above problem:

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>$s_i$</th>
<th>opportunity cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>10</td>
<td>/</td>
<td>5</td>
<td>/</td>
<td>6</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td>15</td>
<td>2</td>
<td>/</td>
<td>7</td>
<td>/</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>9</td>
<td>/</td>
<td>1</td>
<td>20</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>d</td>
<td>j</td>
<td></td>
<td>15</td>
<td>20</td>
<td></td>
<td>5</td>
<td>20</td>
</tr>
</tbody>
</table>

Vogel's approximation → regret-based method → estimating the future profit or cost → decisions are made to avoid future costs (regret).
1.) Starting with the same table as for the NW-Corner-Rule. No row or column is delete. .

2.) Starting from left select the first available column.

3.) In this column select the smallest available $c_{ij}$ and increase the according $x_{ij}$ as much as possible.

4.) If the resource of the column is used, delete the column, OR

   if the resource of the row is used, delete the row.

5.) If there is only one column or row left → choose all free xij of that row or column as base variable (BV) with the maixmum possible values → otherwise proceede with 2.)
Example: Starting solution for above problem:

<table>
<thead>
<tr>
<th>i\j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>s_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td></td>
<td></td>
<td>10</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>20</td>
<td>30</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>d_j</td>
<td>15</td>
<td>20</td>
<td>30</td>
<td>35</td>
<td>100</td>
</tr>
</tbody>
</table>

Attention:

Column minimum method is a greedy method
only the costs of a single column are considered
delivers optimal solutions for level-workforce plans
MODI, stepping stone

Similar to the simplex method, but less memory intensive

Start with a base solution gained by any heuristic method

Iteration for the transportation simplex method:

Build the $m \times n$-table like for the heuristic method, but denote the costs $c_{ij}$ in the upper left corner of each cell and the value of the base variables in the middle of the cells.

<table>
<thead>
<tr>
<th>i\j</th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>n</th>
<th>$s_i$</th>
<th>$u_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$c_{11}$</td>
<td>$c_{12}$</td>
<td>...</td>
<td>$c_{1n}$</td>
<td>$s_1$</td>
<td>$u_1$</td>
</tr>
<tr>
<td>2</td>
<td>$c_{21}$</td>
<td>$c_{22}$</td>
<td>...</td>
<td>$c_{2n}$</td>
<td>$s_2$</td>
<td>$u_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>m</td>
<td>$c_{m1}$</td>
<td>$c_{m2}$</td>
<td>...</td>
<td>$c_{mn}$</td>
<td>$s_m$</td>
<td>$u_m$</td>
</tr>
<tr>
<td>$d_j$</td>
<td>$d_1$</td>
<td>$d_2$</td>
<td>...</td>
<td>$d_n$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_j$</td>
<td>$v_1$</td>
<td>$v_2$</td>
<td>...</td>
<td>$v_n$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For the current base solution \( u_i \) and \( v_j \) can be computed as follows:

\[
c_{ij} = u_i + v_j \text{ if } x_{ij} \text{ is a BV}
\]

The values are denoted in the last column and last row of the extended table. \( u_i \) and \( v_j \) are not well-defined. Therefore one of these dual variables will be set to 0. It simplifies the calculations if the column/row with the most BVs is selected.

For all NBVs compute the coefficient for the objective function \( c_{ij} - u_i - v_j \) write it to the table. The most negative coefficient determines the new BV. The solution is optimal if all coefficients are non-negative.

Increase the new BV and follow the chain reaction of the other BVs. Note, that the sum of the rows and columns of the BVs must not change. The BV, which reaches first 0 is skipped (becomes a NBV). [stepping stone]

Compute the new base solution, i.e. perform the chain reaction and proceed with the next step.
Example: Starting solution for above problem:

Using the solution of the NW-Corner Rule b. Compute the $u_i$ and $v_j$ according to step 1 ($u_1=0$). Compute the coefficients of the NBVs (step 2).

<table>
<thead>
<tr>
<th>i\j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>s_i</th>
<th>u_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>5</td>
<td>6</td>
<td>-4</td>
<td>11</td>
<td>-3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-6</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>-7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
<td>-3</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
<td>-6</td>
</tr>
<tr>
<td>d_j</td>
<td>15</td>
<td>20</td>
<td>30</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v_j</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Total costs of this solution: $10 \times 15 + 5 \times 10 + 2 \times 10 + 7 \times 15 + 4 \times 15 + 8 \times 35 = 665$.

To check for mistakes ensure that after each iterations the objective of the primal and the dual problem are equal:

$$K = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij} = \sum_{i=1}^{m} u_{i}s_{i} + \sum_{j=1}^{n} v_{j}d_{j}$$

$$K = 25 \times 0 + 25 \times (-3) + 50 \times (-6) + 15 \times 10 + 20 \times 5 + 30 \times 10 + 35 \times 14 = 665$$

Chose the most negative coefficient $c_{ij} - u_{i} - v_{j}; -7$ bei $x_{24}$

$\Rightarrow$ new BV $x_{24}$.

Chain reaction: Increase the value of the new BV by $\varepsilon$ and observe the changes of the other BVs.
Chain reaction:

new BV \( x_{24} = 0 \) → increase by \( \varepsilon \). Other BVs are increased/decreased by \( + \varepsilon \) or \( - \varepsilon \).

If \( x_{24} \) is increased by \( \varepsilon \), \( x_{23} \) and \( x_{34} \) must decrease by \( \varepsilon \), and \( x_{33} \) must increase by \( \varepsilon \). For \( \varepsilon = 15 \) \( x_{23} \) reaches 0 → BV \( x_{23} \) is removed.

\[
K = 665 - 7 \times \varepsilon
\]
\[
= 665 - 7 \times 15 = 560
\]

<table>
<thead>
<tr>
<th>( i ) ( j )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>( s_i )</th>
<th>( u_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>5</td>
<td>6</td>
<td>-4</td>
<td>11</td>
<td>-3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-6</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>-7</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>( d_i )</td>
<td>15</td>
<td>20</td>
<td>30</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( v_j )</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
new $BV_{24} = 15 \rightarrow$ chain reaction

$x_{33} = 15 + 15 = 30$

$x_{23}$ is no $BV$; all other $BVs$ remain unchanged.

Next iteration

<table>
<thead>
<tr>
<th>$i,j$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>$s_i$</th>
<th>$u_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td><strong>15- $\varepsilon$</strong></td>
<td><strong>10+ $\varepsilon$</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-6</td>
<td>7</td>
<td>7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>$\varepsilon$</strong></td>
<td><strong>10- $\varepsilon$</strong></td>
<td></td>
<td></td>
<td><strong>15</strong></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>-2</td>
<td>1</td>
<td>-5</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>30</strong></td>
<td><strong>20</strong></td>
</tr>
<tr>
<td>$d_j$</td>
<td>15</td>
<td>20</td>
<td>30</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_j$</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$K = 560 - 6 \cdot \varepsilon$

$= 560 - 6 \cdot 10 = 500$
next iteration

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>s_i</th>
<th>u_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>10</td>
<td>5</td>
<td>6</td>
<td>-3</td>
<td>11</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>0</td>
<td>25</td>
<td>-9</td>
<td>25</td>
<td>-5</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>-9</td>
<td>25</td>
<td>-9</td>
<td>25</td>
<td>-5</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>-5</td>
<td>50</td>
<td>-5</td>
<td>50</td>
<td>-5</td>
</tr>
<tr>
<td>d_j</td>
<td>15</td>
<td>20</td>
<td>30</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v_j</td>
<td>10</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

K = 500 − 3 * \( \varepsilon \)

= 500 − 3*5 = 485
### Production Management

Next iteration

<table>
<thead>
<tr>
<th>i\j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>s_i</th>
<th>u_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20-(\varepsilon)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ K = 485 - 2 \times \varepsilon \]

\[ = 485 - 2 \times 20 = 445 \]
All NBVs have non-negative coefficients → optimale solution

base variables:
- $x_{13} = 25$
- $x_{21} = 15$
- $x_{24} = 10$
- $x_{32} = 20$
- $x_{33} = 5$
- $x_{34} = 25$

Total cost: $K = 445$
Aggregate Planning

Disaggregating Plans
aggregate units are not actually produced, so the plan should consider individual products

disaggregation
master production schedule

Questions:
In which order should individual products be produced?
e.g.: shortest run-out time $R_i = I_i / D_i$

How much of each product should be produced?
e.g.: balance run-out time
Aggregate Planning

Advanced Production Planning Models

Multiple Products

same notation as before

add subscript i for product i

Objective function

$$\min \sum_{t=1}^{T} \left( C_t^W W_t + C_t^H H_t + C_t^L L_t + \sum_{i=1}^{N} C_{it}^P P_{it} + C_{it}^I I_{it} \right)$$
Aggregate Planning

subject to

\[
\sum_{i=1}^{N} \left( \frac{1}{n_{it}} \right) P_{it} \leq W_t \quad t = 1, 2, \ldots, T
\]

\[
W_t = W_{t-1} + H_t - L_t \quad t=1,2,\ldots,T
\]

\[
I_{it} = I_{it-1} + P_{it} - D_{it} \quad t=1,2,\ldots,T; i=1,2,\ldots,N
\]

\[
P_{it}, W_t, H_t, L_t, I_{it} \geq 0 \quad t=1,2,\ldots,T; i=1,2,\ldots,N
\]
Aggregate Planning

Computational Effort:

10 products, 12 periods: 276 variables, 144 constraints

100 products, 12 periods: 2436 variables, 1224 constraints
Aggregate Planning

Example: Carolina Hardwood Product Mix

Carolina Hardwood produces 3 types of dining tables;
There are currently 50 workers employed who can be hired and laid off at any time;
Initial inventory is 100 units for table 1, 120 units for table 2 and 80 units for table 3;

<table>
<thead>
<tr>
<th></th>
<th>t</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>costs of hiring</td>
<td>420</td>
<td>410</td>
<td>420</td>
<td>405</td>
<td></td>
</tr>
<tr>
<td>costs of lay off</td>
<td>800</td>
<td>790</td>
<td>790</td>
<td>800</td>
<td></td>
</tr>
<tr>
<td>costs per worker</td>
<td>600</td>
<td>620</td>
<td>620</td>
<td>610</td>
<td></td>
</tr>
</tbody>
</table>
Aggregate Planning

The number of units that can be made by one worker per period:

<table>
<thead>
<tr>
<th>t</th>
<th>Table 1</th>
<th>Table 2</th>
<th>Table 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>300</td>
<td>260</td>
</tr>
<tr>
<td>2</td>
<td>220</td>
<td>310</td>
<td>255</td>
</tr>
<tr>
<td>3</td>
<td>210</td>
<td>300</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
<td>290</td>
<td>265</td>
</tr>
</tbody>
</table>

Forecasted demand, unit cost and holding cost per unit are:

<table>
<thead>
<tr>
<th></th>
<th>Demand</th>
<th>Unit costs</th>
<th>Holding costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>Table 1</td>
<td>Table 2</td>
<td>Table 3</td>
</tr>
<tr>
<td>1</td>
<td>3500</td>
<td>5400</td>
<td>4500</td>
</tr>
<tr>
<td>2</td>
<td>3100</td>
<td>5000</td>
<td>4200</td>
</tr>
<tr>
<td>3</td>
<td>3000</td>
<td>5100</td>
<td>4100</td>
</tr>
</tbody>
</table>
Aggregate Planning

Minimize

\[ 600W_1 + 620W_2 + 620W_3 + 610W_4 + 420H_1 + 410H_2 + 420H_3 + 405H_4 + 800L_1 + 790L_2 + 790L_3 + 800L_4 \]
\[ + 120P_{11} + 150P_{21} + 200P_{31} + 125P_{12} + 150P_{22} + 210P_{32} + 120P_{13} + 145P_{23} + 205P_{33} + 125P_{14} + 148P_{24} + 205P_{34} \]
\[ + 10I_{11} + 12I_{21} + 12I_{31} + 9I_{12} + 11I_{22} + 12I_{32} + 10I_{13} + 12I_{23} + 11I_{33} + 10I_{14} + 11I_{24} + 11I_{34} \]

subject to

\[ \frac{P_{11}}{200} + \frac{P_{21}}{300} + \frac{P_{31}}{260} \leq W_1, \quad \frac{P_{12}}{220} + \frac{P_{22}}{310} + \frac{P_{32}}{255} \leq W_2, \]
\[ \frac{P_{13}}{210} + \frac{P_{23}}{300} + \frac{P_{33}}{250} \leq W_3, \quad \frac{P_{14}}{200} + \frac{P_{24}}{290} + \frac{P_{34}}{265} \leq W_4, \]

\[ W_1 = 50 + H_1 - L_1, \quad W_2 = W_1 + H_2 - L_2, \quad W_3 = W_2 + H_3 - L_3, \quad W_4 = W_3 + H_4 - L_4, \]

\[ I_{11} = 100 + P_{11} - 3500, \quad I_{21} = 120 + P_{21} - 5400, \quad I_{31} = 80 + P_{31} - 4500, \]
\[ I_{12} = I_{11} + P_{12} - 3100, \quad I_{22} = I_{21} + P_{22} - 5000, \quad I_{32} = I_{31} + P_{32} - 4200, \]
\[ I_{13} = I_{12} + P_{13} - 3000, \quad I_{23} = I_{22} + P_{23} - 5100, \quad I_{33} = I_{32} + P_{33} - 4100, \]
\[ I_{14} = I_{13} + P_{14} - 3400, \quad I_{24} = I_{23} + P_{24} - 5500, \quad I_{34} = I_{33} + P_{34} - 4600, \]

\[ P_{it}, I_{it}, W_t, H_t, L_t, I_{it} \geq 0 \]
Aggregate Planning

Multiple Products and Processes

\[ T = \text{horizon length, in periods} \]
\[ N = \text{number of products} \]
\[ K = \text{number of resource types} \]
\[ t = \text{index of periods}, \ t = 1, 2, \ldots, T \]
\[ i = \text{index of products}, \ i = 1, 2, \ldots, N \]
\[ k = \text{index of resource types}, \ k = 1, 2, \ldots, K \]
\[ D_{it} = \text{forecasted number of units demanded for product } i \text{ in period } t \]
\[ m_i = \text{number of different processes available to make product } i \]
\[ A_{kt} = \text{amount of resource } k \text{ available in period } t \]
\[ a_{ijk} = \text{amount of resource } k \text{ required by one unit of product } i \text{ if produced by process } j \]
\[ C_{ijt}^P = \text{cost to produce one unit of product } i \text{ using process } j \text{ in period } t \]
\[ C_{it}^I = \text{cost to hold one unit of product } i \text{ in inventory for period } t \]

The decision variables are

\[ P_{ijt} = \text{number of units of product } i \text{ produced by process } j \text{ in period } t \]
\[ I_{it} = \text{number of units of product } i \text{ held in inventory at the end of period } t \]
Aggregate Planning

The linear programming formulation is

\[
\text{Minimize } \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{m_i} (C_{ijt}^P P_{ijt} + C_{it}^I I_{it})
\]

subject to

\[
\sum_{i=1}^{N} \sum_{j=1}^{m_i} a_{ijk} P_{ijt} \leq A_{kt} \quad t = 1, 2, \ldots, T; \quad k = 1, 2, \ldots, K
\]

\[
I_{it} = I_{it-1} + \sum_{j=1}^{m_i} P_{ijt} - D_{it} \quad t = 1, 2, \ldots, T; \quad i = 1, 2, \ldots, N
\]

\[
P_{ijt}, I_{it} \geq 0 \quad t = 1, 2, \ldots, T; \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, m_i
\]
Aggregate Planning

Example: Cactus Cycles process plan
CC produces 2 types of bicycles, street and road;
Estimated demand and current inventory:

<table>
<thead>
<tr>
<th>t</th>
<th>initial inventory</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>street b.</td>
<td>100</td>
<td>1000</td>
<td>1050</td>
<td>1100</td>
</tr>
<tr>
<td>road b.</td>
<td>50</td>
<td>500</td>
<td>600</td>
<td>550</td>
</tr>
</tbody>
</table>

available capacity(hours) and holding costs per bike:

<table>
<thead>
<tr>
<th>t</th>
<th>Capacity(hours)</th>
<th>Holding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Machine</td>
<td>Worker</td>
</tr>
<tr>
<td>1</td>
<td>8600</td>
<td>17000</td>
</tr>
<tr>
<td>2</td>
<td>8500</td>
<td>16600</td>
</tr>
<tr>
<td>3</td>
<td>8800</td>
<td>17200</td>
</tr>
</tbody>
</table>
Aggregate Planning

process costs (process1, process2) and resource requirement per unit:

<table>
<thead>
<tr>
<th>t</th>
<th>Process1</th>
<th>Process2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Street</td>
<td>Road</td>
</tr>
<tr>
<td>1</td>
<td>72</td>
<td>85</td>
</tr>
<tr>
<td>2</td>
<td>74</td>
<td>88</td>
</tr>
<tr>
<td>3</td>
<td>75</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>Machine hours required</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Worker hours required</td>
<td>10</td>
</tr>
</tbody>
</table>
Aggregate Planning

Minimize

\[ 72P_{111} + 80P_{121} + 85P_{211} + 90P_{221} \]
\[ + 74P_{112} + 78P_{122} + 88P_{212} + 95P_{222} \]
\[ + 75P_{113} + 78P_{123} + 84P_{213} + 92P_{223} \]
\[ + 5I_{11} + 6I_{12} + 5I_{13} + 6I_{21} + 7I_{22} + 7I_{23} \]

subject to

\[ 5P_{111} + 4P_{121} + 8P_{211} + 6P_{221} \leq 8600, \]
\[ 5P_{112} + 4P_{122} + 8P_{212} + 6P_{222} \leq 8500, \]
\[ 5P_{113} + 4P_{123} + 8P_{213} + 6P_{223} \leq 8800, \]
\[ 10P_{111} + 8P_{121} + 12P_{211} + 9P_{221} \leq 17000, \]
\[ 10P_{112} + 8P_{122} + 12P_{212} + 9P_{222} \leq 16600, \]
\[ 10P_{113} + 8P_{123} + 12P_{213} + 9P_{223} \leq 17200, \]

\[ I_{11} = 100 + P_{111} + P_{121} - 1000, \quad I_{21} = 50 + P_{211} + P_{221} - 500, \]
\[ I_{12} = I_{11} + P_{112} + P_{122} - 1050, \quad I_{22} = I_{21} + P_{212} + P_{222} - 600, \]
\[ I_{13} = I_{12} + P_{113} + P_{123} - 1100, \quad I_{23} = I_{22} + P_{213} + P_{223} - 550, \]

\[ P_{ijt}, I_{it} \geq 0 \quad t = 1, 2, 3; \quad i = 1, 2; \quad j = 1, 2 \]
Aggregate Planning

solution: Objective Function value = $368,756.25

<table>
<thead>
<tr>
<th></th>
<th>Street Bicycle</th>
<th></th>
<th>Road Bicycle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Process</td>
<td>Inventory</td>
<td>Process</td>
</tr>
<tr>
<td>t</td>
<td>t 1</td>
<td>t 2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>900</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1050</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1100</td>
<td>0</td>
</tr>
</tbody>
</table>
Aggregate Planning

Work to do:

Examples: 5.7, 5.8abcdef, 5.9abcd, 5.10abcd, 5.16abcd, 5.21, 5.22, 5.29, 5.30

Replace capacity columns of table in problem 5.29 with Month Machine Worker

<table>
<thead>
<tr>
<th>Month</th>
<th>Machine</th>
<th>Worker</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1350</td>
<td>19000</td>
</tr>
<tr>
<td>2</td>
<td>1270</td>
<td>19000</td>
</tr>
<tr>
<td>3</td>
<td>1350</td>
<td>19500</td>
</tr>
</tbody>
</table>

Minicase BF SWING II
Production, Capacity and Material Planning

Production plan
quantities of final product, subassemblies, parts needed at distinct points in time

To generate the Production plan we need:
end-product demand forecasts
Master production schedule

Master production schedule (MPS)
delivery plan for the manufacturing organization
exact amounts and delivery timings for each end product
accounts for manufacturing constraints and final goods inventory
Production, Capacity and Material Planning

Based on the MPS:

rough-cut capacity planning

Material requirements planning
  determines material requirements and timings for each phase of production
  *detailed capacity planning*
Production, Capacity and Material Planning

End-Item Demand Estimate →
- Master Production Schedule (MPS)
  →
  - Rough-Cut Capacity
  →
  - Detailed Capacity Planning
  →
  - Material Requirements Planning (MRP)
    →
    - Material Plan
    →
    - Purchasing Plan
      →
      - Shop Orders
      →
      - Shop Floor Control

Updates →

Production Management
Master Production Scheduling

Aggregate plan

demand estimates for individual end-items

demand estimates vs. MPS

inventory

capacity constraints

availability of material

production lead time

...

Market environments

make-to-stock (MTS)

make-to-order (MTO)

assemble-to-order (ATO)
Master Production Scheduling

**MTS**
- produces in batches
- minimizes customer delivery times at the expense of holding finished-goods inventory
- MPS is performed at the end-item level
- production starts before demand is known precisely
- small number of end-items, large number of raw-material items

**MTO**
- no finished-goods inventory
- customer orders are backlogged
- MPS is order driven, consists of firm delivery dates
Master Production Scheduling

ATO

large number of end-items are assembled from a relatively small set of standard subassemblies, or modules
automobile industry
MPS governs production of modules (forecast driven)
Final Assembly Schedule (FAS) at the end-item level (order driven)
2 lead times, for consumer orders only FAS lead time relevant
Master Production Scheduling

MPS - SIBUL manufactures phones

three desktop models A, B, C

one wall telephone D

MPS is equal to the demand forecast for each model

<table>
<thead>
<tr>
<th>WEEKLY MPS (= FORECAST)</th>
<th>Jan</th>
<th>Feb</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Week</td>
<td>Week</td>
</tr>
<tr>
<td>Product</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Model B</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Model C</td>
<td>1500</td>
<td>1500</td>
</tr>
<tr>
<td>Model D</td>
<td>600</td>
<td>600</td>
</tr>
<tr>
<td>weekly total</td>
<td>3100</td>
<td>3000</td>
</tr>
<tr>
<td>monthly total</td>
<td>12200</td>
<td></td>
</tr>
</tbody>
</table>
Master Production Scheduling

MPS Planning - Example

MPS plan for model A of the previous example:

Make-to-stock environment

No safety-stock for end-items

\[ I_t = I_{t-1} + Q_t - \max\{F_t, O_t\} \]

- \( I_t \): end-item inventory at the end of week \( t \)
- \( Q_t \): manufactured quantity to be completed in week \( t \)
- \( F_t \): forecast for week \( t \)
- \( O_t \): customer orders to be delivered in week \( t \)

<table>
<thead>
<tr>
<th>INITIAL DATA Model A</th>
<th>Jan</th>
<th>Feb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Inventory =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1600</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>forecast ( F_t )</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>orders ( O_t )</td>
<td>1200</td>
<td>800</td>
</tr>
</tbody>
</table>
Master Production Scheduling

Batch production: batch size = 2500

\[ I_t = \max\{0, I_{t-1}\} - \max\{F_t, O_t\} \]

\[ Q_t = \begin{cases} 
0, & \text{if } I_t > 0 \\
2500, & \text{otherwise}
\end{cases} \]

\[ I_1 = \max\{0, 1600\} - \max\{1000, 1200\} = 400 \geq 0 \]

\[ I_2 = \max\{0, 400\} - \max\{1000, 800\} = -600 < 0 \Rightarrow Q2 = 2500 \]

\[ I_2 = 2500 + 400 - \max\{1000, 800\} = 1900, \text{ etc.} \]

<table>
<thead>
<tr>
<th>MPS</th>
<th>Jan</th>
<th>Feb</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Week</td>
<td>Week</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Current Inventory = 1600</td>
<td></td>
<td></td>
</tr>
<tr>
<td>orders Ot</td>
<td>1200</td>
<td>800</td>
</tr>
<tr>
<td>Inventory It</td>
<td>1600</td>
<td>400</td>
</tr>
<tr>
<td>MPS Qt</td>
<td><strong>2500</strong></td>
<td><strong>2500</strong></td>
</tr>
<tr>
<td>ATP</td>
<td>400</td>
<td>1400</td>
</tr>
</tbody>
</table>

Production Management
Master Production Scheduling

Available to Promise (ATP)

\[
\begin{align*}
\text{ATP}_1 &= 1600 + 0 - 1200 = 400 \\
\text{ATP}_2 &= 2500 -(800 + 300) = 1400, \text{ etc.}
\end{align*}
\]

Whenever a new order comes in, ATP must be updated

<table>
<thead>
<tr>
<th>MPS</th>
<th>Lot-for-Lot production</th>
<th>Jan</th>
<th>Feb</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Current Inventory =</td>
<td>Week</td>
<td>Week</td>
</tr>
<tr>
<td></td>
<td>1600</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>orders Ot</td>
<td>1200</td>
<td>800</td>
<td>300</td>
</tr>
<tr>
<td>Inventory It</td>
<td>1600</td>
<td>400</td>
<td>0</td>
</tr>
<tr>
<td>MPS Qt</td>
<td>0</td>
<td>600</td>
<td>1000</td>
</tr>
<tr>
<td>ATP</td>
<td>400</td>
<td>0</td>
<td>700</td>
</tr>
</tbody>
</table>
Master Production Scheduling

MPS Modeling

differs between MTS-ATO and MTO

find final assembly lot sizes

additional complexity because of joint capacity constraints

cannot be solved for each product independently
Master Production Scheduling

Make-To-Stock-Modeling

\begin{align*}
Q_{it} &= \text{production quantity of product } i \text{ in period } t \\
I_{it} &= \text{Inventory of product } i \text{ at end of period } t \\
D_{it} &= \text{demand (requirements) for product } i \text{ in time period } t \\
a_i &= \text{production hours per unit of product } i \\
h_i &= \text{inventory holding cost per unit of product } i \text{ per time period} \\
A_i &= \text{set-up cost for product } i \\
G_t &= \text{production hours available in period } t \\
y_{it} &= 1, \text{if set-up for product } i \text{ occurs in period } t (Q_{it} > 0)
\end{align*}
Master Production Scheduling

**Make-To-Stock-Modeling**

\[
\min \sum_{i=1}^{n} \sum_{t=1}^{T} \left( A_i y_{it} + h_i I_{it} \right)
\]

\[
I_{i,t-1} + Q_{it} - I_{it} = D_{it} \quad \text{for all (i,t)}
\]

\[
\sum_{i=1}^{n} a_i Q_{it} \leq G_t \quad \text{for all t}
\]

\[
Q_{it} - y_{it} \sum_{k=1}^{T} D_{ik} \leq 0 \quad \text{for all (i,t)}
\]

\[
Q_{it} \geq 0; I_{it} \geq 0; y_{it} \in \{0,1\}
\]
Assemble-To-Order Modeling

two master schedules

MPS: forecast-driven

FAS: order driven

overage costs

holding costs for modules and assembled products

shortage costs

final product assembly based on available modules

no explicit but implicit shortage costs for modules

final products: lost sales, backorders
**Master Production Scheduling**

$m$ module types and $n$ product types

$Q_{kt} = \text{quantity of module } k \text{ produced in period } t$

$g_{kj} = \text{number of modules of type } k \text{ required to assemble order } j$

**Decision Variables:**

$I_{kt} = \text{inventory of module } k \text{ at the end of period } t$

$y_{jt} = 1, \text{ if order } j \text{ is assembled and delivered in period } t; 0, \text{ otherwise}$

$h_k = \text{holding cost}$

$\pi_{jt} = \text{penalty costs, if order } j \text{ is satisfied in period } t \text{ and order } j \text{ is due in period } t' \ (t' < t); \text{ holding costs if } t' > t$
Master Production Scheduling

Assemble-To-Order Modeling

\[
\min \sum_{k=1}^{m} \sum_{t=1}^{L} h_{k} I_{kt} + \sum_{j=1}^{n} \sum_{t=1}^{L} \pi_{jt} y_{jt}
\]

subject to

\[
I_{kt} = I_{k,t-1} + Q_{kt} - \sum_{j=1}^{n} g_{kj} y_{jt} \quad \text{for all (k, t)}
\]

\[
\sum_{j=1}^{n} a_{j} y_{jt} \leq G_{t} \quad \text{for all } t
\]

\[
\sum_{t=1}^{L} y_{jt} = 1 \quad \text{for all } j
\]

\[
I_{kt} \geq 0; \quad y_{jt} \in \{0,1\} \quad \text{for all (j, k, t)}
\]
Master Production Scheduling

Capacity Planning

- Bottleneck in production facilities
- Rough-Cut Capacity Planning (RCCP) at MPS level feasibility
- Detailed capacity planning (CRP) at MRP level
- Both RCCP and CRP are only providing information
# Master Production Scheduling

## MPS:

<table>
<thead>
<tr>
<th>Product</th>
<th>January</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Week</td>
</tr>
<tr>
<td>A</td>
<td>1000</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>1500</td>
</tr>
<tr>
<td>D</td>
<td>600</td>
</tr>
</tbody>
</table>

## Bill of capacity (min)

<table>
<thead>
<tr>
<th>Product</th>
<th>Assembly</th>
<th>Inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>24</td>
<td>2.5</td>
</tr>
<tr>
<td>C</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>25</td>
<td>2.4</td>
</tr>
</tbody>
</table>

## Capacity requires (hr)

<table>
<thead>
<tr>
<th>Week</th>
<th>Assembly</th>
<th>Inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1133</td>
<td>1083</td>
</tr>
<tr>
<td>2</td>
<td>1333!!</td>
<td>1333!!</td>
</tr>
<tr>
<td>3</td>
<td>883</td>
<td>83</td>
</tr>
<tr>
<td>4</td>
<td>1200</td>
<td>110</td>
</tr>
</tbody>
</table>

- Assembly: \(1000 \times 20 + 1500 \times 22 + 600 \times 25 = 68000\) min = 1133.33 hr
- Inspection: \(1000 \times 2 + 1500 \times 2 + 600 \times 2.4 = 6440\) min = 107.33 hr etc.

Available capacity per week is 1200 hr for the assembly work center and 110 hours for the inspection station;
Master Production Scheduling

Capacity Modeling

heuristic approach for finite-capacity-planning
based on input/output analysis
relationship between capacity and lead time

\( G \) = work center capacity
\( R_t \) = work released to the center in period \( t \)
\( Q_t \) = production (output) from the work center in period \( t \)
\( W_t \) = work in process in period \( t \)
\( U_t \) = queue at the work center measured at the beginning of period \( t \), prior to the release of work
\( L_t \) = lead time at the work center in period \( t \)
Master Production Scheduling

\[ Q_t = \min\{G, U_{t-1} + R_t \} \]
\[ U_t = U_{t-1} + R_t - Q_t \]
\[ W_t = U_{t-1} + R_t = U_t + Q_t \]
\[ L_t = \frac{W_t}{G} \]

Lead time is not constant assumptions:
constant production rate
any order released in this period is completed in this period
Master Production Scheduling

Example

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>G (hr/week)</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Rt (hours)</td>
<td>20</td>
<td>30</td>
<td>60</td>
<td>20</td>
<td>40</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Qt (hours)</td>
<td>30</td>
<td>30</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>Ut (hours)</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>8</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>Wt (hours)</td>
<td>30</td>
<td>30</td>
<td>60</td>
<td>44</td>
<td>48</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>Lt (weeks)</td>
<td>0.83</td>
<td>0.83</td>
<td>1.67</td>
<td>1.22</td>
<td>1.33</td>
<td>1.44</td>
<td></td>
</tr>
</tbody>
</table>
Material Requirements Planning

Inputs

- master production schedule
- inventory status record
- bill of material (BOM)

Outputs

- planned order releases
  - purchase orders (supply lead time)
  - work orders (manufacturing lead time)
Material Requirements Planning

Legend:
S/A = subassembly
PP = purchased part
MP = manufactured part
RM = raw material

part #
quantity
Material Requirements Planning

MRP Process

goal is to find net requirements (trigger purchase and work orders)

explosion

  Example:
  MPS, 100 end items
  yields gross requirements

netting

  Net requirements = Gross requirements - on hand inventory - quantity on order
  done at each level prior to further explosion

offsetting

  the timing of order release is determined

lotsizing

  batch size is determined
Material Requirements Planning

Example 7-6
Material Requirements Planning

PART 11 (gross requirements given)

net requirements?
Planned order release?

Net requ.(week 2) = 600 – (1600 + 700) = -1700 => Net requ.(week2) = 0
Net requ.(week 3) = 1000 – (1700 + 200) = -900 => Net requ.(week3) = 0
Net requ.(week 4) = 1000 – 900 = 100 etc.

<table>
<thead>
<tr>
<th></th>
<th>week</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>current</td>
</tr>
<tr>
<td>gross</td>
<td></td>
</tr>
<tr>
<td>requirements</td>
<td></td>
</tr>
<tr>
<td>scheduled</td>
<td></td>
</tr>
<tr>
<td>receipts</td>
<td></td>
</tr>
<tr>
<td>projected</td>
<td></td>
</tr>
<tr>
<td>inventory</td>
<td></td>
</tr>
<tr>
<td>balance</td>
<td></td>
</tr>
<tr>
<td>net</td>
<td></td>
</tr>
<tr>
<td>requirements</td>
<td></td>
</tr>
<tr>
<td>planned</td>
<td></td>
</tr>
<tr>
<td>receipts</td>
<td></td>
</tr>
<tr>
<td>planned</td>
<td></td>
</tr>
<tr>
<td>order</td>
<td></td>
</tr>
<tr>
<td>release</td>
<td></td>
</tr>
</tbody>
</table>
### Material Requirements Planning

**Assumptions:**
- Lot size: 3000
- Lead time: 2 weeks

<table>
<thead>
<tr>
<th></th>
<th>current</th>
<th>week 1</th>
<th>week 2</th>
<th>week 3</th>
<th>week 4</th>
<th>week 5</th>
<th>week 6</th>
<th>week 7</th>
<th>week 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>gross requirements</td>
<td></td>
<td>600</td>
<td>1000</td>
<td>1000</td>
<td>2000</td>
<td>2000</td>
<td>2000</td>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>scheduled receipts</td>
<td></td>
<td>400</td>
<td>700</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>projected inventory balance</td>
<td>1200</td>
<td>1600</td>
<td>1700</td>
<td>900</td>
<td>2900</td>
<td>900</td>
<td>1900</td>
<td>2900</td>
<td>900</td>
</tr>
<tr>
<td>planned receipts</td>
<td></td>
<td></td>
<td></td>
<td>3000</td>
<td>3000</td>
<td>3000</td>
<td>3000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>planned order release</td>
<td></td>
<td>3000</td>
<td>3000</td>
<td>3000</td>
<td>3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Material Requirements Planning

Multilevel explosion

<table>
<thead>
<tr>
<th>part number</th>
<th>description</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>base assembly</td>
<td>1</td>
</tr>
<tr>
<td>121</td>
<td>housing S/A</td>
<td>1</td>
</tr>
<tr>
<td>123</td>
<td>rubber pad</td>
<td>4</td>
</tr>
<tr>
<td>1211</td>
<td>key pad</td>
<td>1</td>
</tr>
</tbody>
</table>

lead time is one week
lot for lot for parts 121, 123, 1211
part 12: fixed lot size of 3000
<table>
<thead>
<tr>
<th>Part 12</th>
<th>current</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduled receipts</td>
<td></td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>projected inventory balance</td>
<td></td>
<td>800</td>
<td>1200</td>
<td>1000</td>
<td>400</td>
<td>2400</td>
<td>400</td>
<td>1400</td>
<td>2400</td>
</tr>
<tr>
<td>net requirements</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>planned receipts</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3000</td>
<td>0</td>
<td>3000</td>
<td>3000</td>
<td>0</td>
</tr>
<tr>
<td>planned order release</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3000</td>
<td>0</td>
<td>3000</td>
<td>3000</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part 121</th>
<th>current</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>gross requirements</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3000</td>
<td>0</td>
<td>3000</td>
<td>3000</td>
<td>0</td>
</tr>
<tr>
<td>scheduled receipts</td>
<td></td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>projected inventory balance</td>
<td></td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>net requirements</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>planned receipts</td>
<td></td>
<td>0</td>
<td>0</td>
<td>2500</td>
<td>0</td>
<td>3000</td>
<td>3000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>planned order release</td>
<td></td>
<td>0</td>
<td>0</td>
<td>2500</td>
<td>0</td>
<td>3000</td>
<td>3000</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part 123</th>
<th>current</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>gross requirements</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12000</td>
<td>0</td>
<td>12000</td>
<td>12000</td>
<td>0</td>
</tr>
<tr>
<td>scheduled receipts</td>
<td></td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>projected inventory balance</td>
<td></td>
<td>15000</td>
<td>15000</td>
<td>25000</td>
<td>13000</td>
<td>13000</td>
<td>10000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>net requirements</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>planned receipts</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>planned order release</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part 1211</th>
<th>current</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>gross requirements</td>
<td></td>
<td>0</td>
<td>0</td>
<td>2500</td>
<td>0</td>
<td>3000</td>
<td>3000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>scheduled receipts</td>
<td></td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>projected inventory balance</td>
<td></td>
<td>1200</td>
<td>2700</td>
<td>200</td>
<td>200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>net requirements</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>planned receipts</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2800</td>
<td>3000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>planned order release</td>
<td></td>
<td>0</td>
<td>0</td>
<td>2800</td>
<td>3000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Material Requirements Planning

**MRP Updating Methods**

MRP systems operate in a dynamic environment.

Regeneration method: the entire plan is recalculated.

Net change method: recalculates requirements only for those items affected by change.

<table>
<thead>
<tr>
<th>Product</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>350</td>
<td>-</td>
<td>-</td>
<td>350</td>
<td>500</td>
<td>-</td>
<td>200</td>
<td>150</td>
</tr>
<tr>
<td>C</td>
<td>1000</td>
<td>-</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>-</td>
<td>800</td>
<td>1000</td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>300</td>
<td>200</td>
<td>-</td>
<td>-</td>
<td>300</td>
<td>200</td>
<td>-</td>
</tr>
</tbody>
</table>

**Net Change for February**

<table>
<thead>
<tr>
<th>Product</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>300</td>
<td>-100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>150</td>
<td>200</td>
<td>-200</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>200</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>200</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Updated MPS for February**

<table>
<thead>
<tr>
<th>Product</th>
<th>Week</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td>350</td>
<td>-</td>
<td>-</td>
<td>350</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>1000</td>
<td>-</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>-</td>
<td>300</td>
<td>200</td>
<td>-</td>
</tr>
</tbody>
</table>
Material Requirements Planning

Additional Netting procedures

implosion:
  opposite of explosion
  finds common item
combining requirements:
  process of obtaining the gross requirements of a common item
pegging:
  identify the item’s end product
  useful when item shortages occur
Material Requirements Planning
Lot Sizing in MRP

minimize set-up and holding costs

can be formulated as MIP

a variety of heuristic approaches are available

simplest approach: use independent demand procedures (e.g. EOQ) at every level
Material Requirements Planning

MIP Formulation

Indices:

\[ i = 1 \ldots P \]

label of each item in BOM (assumed that all labels are sorted with respect to the production level starting from the end-items)

\[ t = 1 \ldots T \]

period \( t \)

\[ m = 1 \ldots M \]

resource \( m \)
Material Requirements Planning

MIP Formulation

Parameters:

- $\Gamma(i)$: set of immediate successors of item $i$
- $\Gamma^{-1}(i)$: set of immediate predecessors of item $i$
- $s_i$: setup cost for item $i$
- $c_{ij}$: quantity of item $i$ required to produce item $j$
- $h_i$: holding cost for one unit of item $i$
- $a_{mi}$: capacity needed on resource $m$ for one unit of item $i$
- $b_{mi}$: capacity needed on resource $m$ for the setup process of item $i$
- $L_{mt}$: available capacity of resource $m$ in period $t$
- $oc_m$: overtime cost of resource $m$
- $G$: large number, but as small as possible (e.g. sum of demands)
- $D_{it}$: external demand of item $i$ in period $t$
Material Requirements Planning

Decision variables:

- \( x_{it} \) deliverd quantity of item \( i \) in period \( t \)
- \( I_{it} \) inventory level of item \( i \) at the end of period \( t \)
- \( O_{mt} \) overtime hours required for machine \( m \) in period \( t \)
- \( y_{it} \) binary variable indicating if item \( i \) is produced in period \( t \) (=1) or not (=0)

Equations:

\[
\min \sum_{i=1}^{P} \sum_{t=1}^{T} (s_i y_{it} + h_i I_{it}) + \sum_{t=1}^{T} \sum_{m=1}^{M} o c_m O_{mt}
\]

\[
I_{i,t} = I_{i,t-1} + x_{i,t} - \sum_{j \in \Gamma(i)} c_{ij} x_{jt} - D_{it} \quad \forall i, t
\]

\[
x_{it} - G y_{it} \leq 0 \quad \forall i, t
\]

\[
\sum_{i=1}^{P} (a_{mi} x_{it} + b_{mi} y_{it}) \leq L_{mt} + O_{mt} \quad \forall m, t
\]

\[
x_{it}, I_{it}, O_{mt} \geq 0, \quad y_{it} \in \{0,1\} \quad \forall i, m, t
\]
Production oriented decomposition without cost adaptation (Erzeugnisorientierte Dekomposition ohne Kostenanpassung)

The simplest method, widely used in practical applications and implemented in PPS-systems, ignores the cost effects concerning a lot sizing decision for a product to the predecessor products. The basic procedure (not to consider these effects) is the following:

Start with the end-item and lotsize with single product heuristic or wagner-whithin method. (In the general case the different levels are considered in an ordered way starting with the end-item.

Plan the immediate predecessor products – the demand for these predecessor products results from the lotsizing decision of the successor products. – when one level is lotsized go to the next level to lotsize the products.
Example: \( N = 2 \) Products, \( T = 4 \) periods, \( a_{12} = 1 \), demand, setup cost and inventory holding cost:

<table>
<thead>
<tr>
<th>product</th>
<th>( t = 1 )</th>
<th>( t = 2 )</th>
<th>( t = 3 )</th>
<th>( t = 4 )</th>
<th>( S_i )</th>
<th>( h_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i = 1 )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>120</td>
<td>10</td>
</tr>
<tr>
<td>( i = 2 )</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>100</td>
<td>11</td>
</tr>
</tbody>
</table>

Planning of product oriented decomposition without cost adaptation:

The end item \( i = 2 \) is lotsized. We receive the following Silver-Meal solution:

\( t = 1: 100/1 < [100 + 11 \times 10]/2 = 105 \) d.h. \( q_{21} = 10 \), no lot generation \( \rightarrow q_{22} = 10, q_{23} = 10, q_{24} = 10 \).

We have the demand of the predecessor product:

<table>
<thead>
<tr>
<th>Produkt</th>
<th>( t = 1 )</th>
<th>( t = 2 )</th>
<th>( t = 3 )</th>
<th>( t = 4 )</th>
<th>( S_i )</th>
<th>( h_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i = 1 )</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>120</td>
<td>10</td>
</tr>
</tbody>
</table>
Using Silver-Meal we calculate the following lots:

\[ t = 1: 120/1 > [120 + 10 \times 10]/2 = 110, \text{ but } 110 < [120 + 10 \times 10 + 10 \times 2 \times 10]/3 = 140 \] 
\[ \text{d.h. Lots: } q_{11} = 10 + 10, q_{12} = 0. \]

\[ t = 3: 120/1 > 110, \text{ d.h. Lots: } q_{13} = 10 + 10, q_{14} = 0. \]

The total costs are 840:

| product 2: | setup, is 400 |
| product 1: | 2 \times setup, 2 \times holding costs, 240 + 200 = 440 |

To compare: **Lot generation at the end-item:**

\[ q_{21} = 20, q_{22} = 0, q_{23} = 20, q_{24} = 0 \]

these results in demands of the predecessor product 1:

<table>
<thead>
<tr>
<th>product</th>
<th>( t = 1 )</th>
<th>( t = 2 )</th>
<th>( t = 3 )</th>
<th>( t = 4 )</th>
<th>( S_i )</th>
<th>( h_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i = 1 )</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>120</td>
<td>10</td>
</tr>
</tbody>
</table>
Applying Silver-Meal we receive the following lots:

\[ t = 1: \frac{120}{1} > \frac{120 + 0}{2} = 60, \text{ aber } 60 < \frac{120 + 0 + 10 \times 2 \times 20}{3} = 173.3 \]

d.h. Lot generation: \( q_{11} = 20, q_{12} = 0 \).

\[ t = 3: \text{the same lot generation: } q_{13} = 20, q_{14} = 0. \]

The total costs are the following 660:

Product 2: \( 2 \times \text{setup, } 2 \times \text{holding costs, is } 200 + 220 = 420 \)

Product 1: \( 2 \times \text{setup, is } 240 \)

*The solution of the previous section can be improved by more than 20%*

The lotsizing decision of the end item should consider, that also the costs of the predecessor products are influenced. This leads to the idea of cost adaptation (Kostenanpassung). The systematic change (increase) of inventory holding costs and setup costs of the successor products should consider the costs of the predecessor products (in lotsizing the successor product).
Assumption: converging product structure (every product (except the end-items have a clear defined predecessor) There are different approaches with the following idea:

By calculating the modified costs fixed primary demands are assumed.
Bei Ermittlung der modifizierten Kosten wird von konstanten Primärbedarfsmengen ausgegangen.

→ calculate multiplicators \( \pi_i \), indicates how often a lot of the successor product \( n(i) \). By nested strategies \( \pi_i \geq 1 \)

On the basis of \( \pi_i \) the inventory holding costs and setup costs of successor products are modified.
variant 1: ELSP with converging product structure

the following multiplicators are calculated

the setup costs are corrected:

the inventory holding cost \( h_j \) will not be modified.

\[
\pi_i = \sqrt{\frac{S_i h_{n(i)}}{S_{n(i)} h_i}}
\]

\[
\hat{S}_j = S_j + \sum_{i \in V(j)} S_i / \pi_i
\]

for the example above:

\[
\pi_1 = \sqrt{\frac{120 \times 11}{100 \times 10}} = 1,15 \quad \rightarrow \quad \hat{S}_2 = 100 + 120/1,15 = 204,35
\]

Silver-Meal for end item 2:

\( q_{21} = 20, \ q_{22} = 0, \ q_{23} = 20, \ q_{24} = 0, \) denn

\[
\frac{204,35}{1} > \frac{[204,35 + 11\times10]}{2} = 157,18 < \frac{[204,35 + 330]}{3} = 178,12
\]
Method of Afentakis

There is a number of heuristic methods which can be categorized in the following:

*product oriented decomposition*: independent one product models are considered, which are connected through cost adaptation

*period oriented decomposition*: all products are considered simultaneously and the planning horizon is step-by-step extended.

A typical method for period oriented decomposition is the method of Afentakis (1987). Here step by step for $t=1,2,...,T$ an approximate optimal solution $Q(t)$ for the planning horizon $[1,t]$ is calculated.
Afentakis II

We assume, that only for the end-item N a primary demand $d_{Nt}$ exists.

initial solution

$$Q(1) = \begin{pmatrix} q_{11} \\ \vdots \\ q_{N1} \end{pmatrix} = \begin{pmatrix} v_{1N}d_{N1} \\ \vdots \\ v_{NN}d_{N1} \end{pmatrix}$$

step $t-1 \rightarrow t$:

Initial situation:  

$$Q(t-1) = \begin{pmatrix} q_{1}(t-1) \\ \vdots \\ q_{N}(t-1) \end{pmatrix} \text{ wobei } q_i(t-1) = (q_{i1}, \ldots, q_{i,t-1})$$

Moreover $\tau_{i,t-1}$ is the last production period of product i, the last period with positive lotsize.
Now politic $Q(t)$, for all $i$ will be calculated.

All production periods are preserved. The demand of product $i$ of the period $t$ will be either fulfilled by increasing the production amount $\tau_{i,t-1}$ or set up a new lot of product $i$ in one of the periods $\tau_{i,t-1} + 1, \ldots, t$. There are $t + 1 - \tau_{i,t-1}$ potential periods possible.

The scenario has to be nested, we only set up a lot for $i$, when also for all the direct (and also the indirect) successor a lot is set up. $x_{it} = 1 \implies x_{n(i),t} = 1$. These feature is in the optimal scenario fulfilled, therefore it makes sense to use this feature also in the heuristic solution construction process.

→ out of all the scenarios the cheapest scenario is selected.
Beispiel: $T = 3$, $N = 3$. End-item 3 und predecessor product 1 and 2 where $a_{13} = a_{23} = 1$ und $a_{ij} = 0$ sonst.

Set up cost $S_1 = 8$, $S_2 = 10$, $S_3 = 5$.

inventory holding cost $h_3 = 3$, $h_1 = h_2 = 1$ (systemwide costs $H_1 = H_2 = H_3 = 1$).

primary demand for end-item 3: $d_{31} = 5$, $d_{32} = 9$, $d_{33} = 8$.

begin and end of the planning horizon inventory balance = 0.

Initial solution $t=1$: every product is produced in $t=1$.

$$X(1) = \begin{pmatrix} x_{11} \\ x_{21} \\ x_{31} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \text{also} \quad Q(1) = \begin{pmatrix} q_{11} \\ q_{21} \\ q_{31} \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \\ 5 \end{pmatrix} \quad \text{with costs } 8 + 10 + 5 = 23$$
Iteration $t = 1$: there exist 5 potential scenarios – the non-nested scenarios are not considered.

$$X (2) = \begin{pmatrix} 1 & x_{12} \\ 1 & x_{22} \\ 1 & x_{32} \end{pmatrix}$$

Lösung:

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Kosten:

23 + 9 + 8 + 5 = 46

23 + 9 + 10 + 5 = 48

23 + 8 + 10 + 5 = 48

= 50

= 46

= 48

= 48

= 48
Iteration $t = 2$: there exist 8 potential scenarios

Lösung:

\[
\begin{bmatrix}
1 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 0 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

Kosten:

\[45 + 8(1+2+1) = 77\]

\[45 + 8(1+2) + 5 = 74\]

\[45 + 8 \times 2 + 8 + 5 = 74\]

\[45 + 8 + 10 + 5 = 68\]

Lösung:

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 0 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 0 \\
1 & 1 & 0 \\
\end{bmatrix}
\]

Kosten:

\[45 + 8 + 10 + 5 = 68\]

\[46 + 8(1+1) + 5 = 67\]

\[46 + 8 + 8 + 5 = 67\]

\[46 + 8(1+1+1) = 70\]
Aftenakis VII

Approximate optimal solution [1, ..., 3]:

\[
X(3) = \begin{pmatrix}
1 & 1 & 0 \\
1 & 1 & 0 \\
1 & 1 & 1 \\
\end{pmatrix}
\]
oder

\[
X(3) = \begin{pmatrix}
1 & 1 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1 \\
\end{pmatrix}
\]

The lotsizing decisions are the following:

\[
Q(3) = \begin{pmatrix}
5 & 17 & 0 \\
5 & 17 & 0 \\
5 & 9 & 8 \\
\end{pmatrix}
\]
oder

\[
Q(3) = \begin{pmatrix}
5 & 9 & 8 \\
5 & 17 & 0 \\
5 & 9 & 8 \\
\end{pmatrix}
\]
LP-Models for multiperiod dynamic models without capacity restrictions

LP-Modell with „normal“ inventory levels

\( i \) ... Index for the predecessor products  \( (i = 1,\ldots,N-1) \)

\( N \) ... *Index for the end-item*

\( t \) ... Index for the periods  \( (t = 1,\ldots,T) \)

\( h_i \) ... inventory holding costs for product \( i \)

\( S_i \) ... setup costs for product \( i \)

\( d_{it} \) ... demand on \( i \) in period \( t \)  \( \text{(primary demand)} \)

\( q_{it} \) ... lotsize of product \( i \) in period \( t \)

\( y_{it} \) ... inventory level of product \( i \) end of period \( t \)

\( N(i) \) ... set of the direct successors of product \( i \)
LP-Modell with „normal“ inventory levels II

\[ a_{ij} \ldots \text{direct demand coefficient, d.h. amount of product } i, \text{ which is in 1 unit of product } j \text{ integrated (number in arc } i \rightarrow j \text{ in Gozintographen)} \]

Furthermore \[ x_{it} = \begin{cases} 1 & \text{falls } q_{it} > 0 \\ 0 & \text{falls } q_{it} = 0 \end{cases} \]

a binary variable, indicating if the product is lotsized.

Assumption: the production in period \( t \) is for the demand in period \( t \). no loss of sales or backorders are allowed. the complete demand has to be satisfied \( \rightarrow \) therefore the production quantity is fixed therefore the constant variable production costs can be not considered.

Example: \( N = 3 \)

1 item end-item 3 consists of 1 part predecessor product 1 and 2 parts of predecessor product 2. In predecessor product 1 is one additional unit of predecessor product 2.

\[ N(1) = \{3\} \]
\[ N(2) = \{1, 3\} \]
\[ N(3) = \{\} \]
**LP-Model**

\[
C = \sum_{t=1}^{T} \sum_{i=1}^{N} \left[ h_i y_{it} + S_i x_{it} \right] \rightarrow \text{min!}
\]

**Cost**

- Inventory balance equation:
  \[
y_{Nt} = y_{N,t-1} + q_{Nt} - d_{Nt} \quad \text{für alle } t = 1, \ldots, T
\]
  \[
y_{it} = y_{i,t-1} + q_{it} - d_{it} - \sum_{j \in N(i)} a_{ij} q_{jt} \quad \text{für alle } i = 1, \ldots, N - 1; \ t = 1, \ldots, T
\]
  \[
y_{i0} = y_{iT} = 0 \quad \text{für alle } i = 1, \ldots, N
\]

**Setup costs:**

\[
q_{it} \leq M \cdot x_{it} \quad \text{für alle } i = 1, \ldots, N; \ t = 1, \ldots, T
\]

wobei \( M \) is a large number.

**Non-negativity:**

\[
q_{it} \geq 0; y_{it} \geq 0 \quad \text{für alle } i = 1, \ldots, N; \ t = 1, \ldots, T
\]

**Binary variable:**

\[
x_{it} \in \{0,1\} \quad \text{für alle } i = 1, \ldots, N; \ t = 1, \ldots, T
\]
the system wide inventory level is used:

\[ Y_{it} = y_{it} + \sum_{j \in N^*(i)} v_{ij} y_{jt} \]

...system wide inventory level of product i on the end of period t, all products i which are already integrated in some other products (successor product) or on stock.

\[ v_{ij} \] ... Nesting-coefficient of product i wrt to product j, d.h. amount of product i which is directly or indirectly in 1 unit of product j integrated.

\[ N^*(i) \] ... set of all successors (also the indirect ones)

Recalculation from \( Y_{it} \) to \( y_{it} \) via

\[ y_{it} = Y_{it} - \sum_{j \in N(i)} a_{ij} Y_{jt} \]
analog definiert man:

$$ H_i = h_i - \sum_{k \in V(i)} a_{ki} h_k $$

... system wide inventory holding cost for product $i$, 

\[ V(i) \] ... set of all direct predecessors of $i$

**Obiges Beispiel:**

$N^*(i) = N(i)$ hier z.B.: $a_{23} = 2$, $v_{23} = 2 + 1 = 3$

Also $Y_{2t} = y_{2t} + 1y_{1t} + 3y_{3t}$

$V(1) = \{2\}$, $V(3) = \{1, 2\}$

Wenn z.B. $h_1 = 2$, $h_2 = 1$, $h_3 = 6$,

dann $H_2 = 1$, $H_1 = 2 - 1 = 1$, $H_3 = 6 - 1 \times 2 - 2 \times 1 = 2$
\textbf{LP - Formulierung}

\textbf{cost:} \\
\[ C = \sum_{t=1}^{T} \sum_{i=1}^{N} \left[ H_i Y_{it} + S_i x_{it} \right] \rightarrow \min! \]

\textbf{inventory balance:} \\
\[ Y_{it} = Y_{i,t-1} + q_{it} - d_{it} - \sum_{j \in N^*(i)} v_{ij} d_{jt} \quad \text{für alle } i = 1, \ldots, N; \quad t = 1, \ldots, T \]
\[ Y_{i0} = Y_{iT} = 0 \quad \text{für alle } i = 1, \ldots, N \]

\textbf{no loss sales:} \\
\[ Y_{it} - \sum_{j \in N(i)} a_{ij} Y_{it} \geq 0 \]

\textbf{setup costs:} \\
\[ q_{it} \leq M \cdot x_{it} \quad \text{für alle } i = 1, \ldots, N; \quad t = 1, \ldots, T \]
\[ \text{wobei } M \text{ eine große Zahl ist.} \]

\textbf{non-negativity:} \\
\[ q_{it} \geq 0; Y_{it} \geq 0 \quad \text{für alle } i = 1, \ldots, N; \quad t = 1, \ldots, T \]

\textbf{binary variables:} \\
\[ x_{it} \in \{0,1\} \quad \text{für alle } i = 1, \ldots, N; \quad t = 1, \ldots, T \]
converging product structure
when every product (except the end-item) has exactly one successor (converging product structure) – the above formulas can be simplified.

the above formulation can be replaced

\[ y_{it} = y_{i,t-1} + q_{it} - d_{it} - \sum_{j \in N(i)} a_{ij} q_{jt} \quad \text{für alle } i = 1, \ldots, N - 1; \ t = 1, \ldots, T \]

\[ y_{it} = y_{i,t-1} + q_{it} - d_{it} - a_{i,n(i)} q_{n(i),t} \quad \text{für alle } i = 1, \ldots, N - 1; \ t = 1, \ldots, T \]

by, where \( n(i) \) ist the sole successor of \( i \), so \( N(i) = \{ n(i) \} \).

In the formulation with systemwide inventory we have the following simplification:

no loss of sales

\[ Y_{it} - a_{i,n(i)} Y_{i,n(i)} \geq 0 \]
in the context of cost adaptation the system wide concept is also used:

**Variante 4: systemwide costs $H_i$**

the values $\pi_i$ are calculated as follows:

$$\pi_i = \sqrt{\frac{S_i H_{n(i)}}{S_{n(i)} H_i}}$$

the costs will be corrected as follows:

$$\hat{S}_j = S_j + \sum_{i \in V(j)} \frac{\hat{S}_i}{\pi_i} \quad \text{und} \quad \hat{H}_j = H_j + \sum_{i \in V(j)} \frac{\hat{H}_i}{\pi_i}$$
some notes on capacity restrictions

In LP models capacity restrictions can be easily considered. Within the heuristics some problems occur.

single level problems clsp (capacitated lot sizing problem)
multi-level problems MLCLSP, multi level CLSP – heuristics ACO, Simulated annealing

→ Pitakaso et al.
Scheduling: The Role of Scheduling!

• Decision Making Process used on a regular basis in many manufacturing and service industries

• allocation of resources and tasks
• over given time periods
• and its goal to optimize one or more objectives

• resources
  – machines in a workshop
  – runways at an airport
  – crews at a construction site
  – processing units in a computing environment.
Scheduling: The Role of Scheduling!

- each task may have a certain priority level
- an earliest possible starting time
- a due date

many different objectives
- minimization of the completion time of the last task
- minimization of the number of tasks completed after their respective due dates.

Scheduling plays an important role
- in most manufacturing and production systems
- as well as information processing environments
- transportation and distribution
- other types of service industries (travelling repairmen).
An illustrative example – A Paper Bag Factory

A factory produces paper bags for cement, charcoal, dog food, ...

Basic raw material
- rolls of paper

Production Process
- Printing of the Logo
- Gluing of the side of the bag
- Sewing of one end of both ends bag

Each stage consists of a number of machines which are not necessarily identical
- the machines may differ in speed
- the number of colors they can print
- the size of the bag they can produce
An illustrative example – A Paper Bag Factory

Each production order indicates a given quantity of a specific bag that has to be produced and shipped
 • shipping date
 • due date
 • processing times are proportional to the size of the order

Late delivery implies a penalty in the form of loss of goodwill
the magnitude of the penalty depends on the importance of the order or the client
the tardiness of the delivery.

One objective is to minimize these penalties.

Machine is switched over for one type of bag to another type of bag a setup is required.

The length of the setup time on the machine depends on the similarities between two consecutive orders (the number of colors in common, the differences in bag size and so on). An important objective of the scheduling system is the minimization of the total time spent on setups.
Gate Assignment at an Airport

Airline terminal at a major airport
There are dozens of gates and hundreds of planes arriving and departing each day.

- gates are not identical
- planes are not identical

some of the gates are in locations with a lot of space where large planes can be accommodated easily.
other gates are in locations where it is difficult to bring in the planes, certain planes may actually have to be towed to their gates.
Gate Assignment at an Airport

Planes arrive and depart according to a certain schedule. Schedule is subject to a certain amount of randomness which may be weather related or caused by unforeseen events.

During the time that a plane occupies a gate the passengers have to be deplaned, the plane has to be serviced and the departing passengers have to be boarded.

The scheduled departure time can be viewed as a due date and the airlines performance is measured accordingly.

The scheduler has to assign planes to gates that the assignment is physically feasible while optimizing a number of objectives. The objectives include minimization of work for airline personnel and minimization of airplane delays. Gates are the ressources and handling and servicing of the planes are the tasks. Arrival is starting time of the task and departure is completion time.
Scheduling Topics

Single Machine Models
Advanced Single Machine Models
Parallel Machine Models
Flow Shop
Job Shop
Operations Scheduling

Buffer  \rightarrow  Soldering (Löten)  \rightarrow  Visual Inspection  \rightarrow  Special Stations  \rightarrow  Buffer

workforce
Operations Scheduling

Scheduling is
the process of organizing, choosing and timing resource usage
to carry out all the activities necessary to produce the desired
outputs at the desired times, while satisfying a large number of
time and relationship constraints among the activities and the
resources (Morton and Pentico, 1993).

Schedule specifies
the time each job starts and completes on each machine, as
well as any additional resources needed.

A Sequence is
a simple ordering of the jobs.
Determining a **best** sequence

32 jobs on a single machine

32! Possible sequences approx. $2.6 \times 10^{35}$

 suppose a computer could examine one billion sequences per second

 it would take $8.4 \times 10^{15}$ centuries

real life problems are much more complicated

Scheduling theory helps to

 classify the problems

 identify appropriate measures

 develop solution procedures
Algorithmic complexity

an efficient algorithm is one whose effort of any problem instance is bounded by a polynomial in the problem size, e.g. # of jobs
minimal spanning tree can be solved in at most $n^2$ iterations
n: number of edges
$O(n^2)$
if effort is exponential $O(2^n)$ the algorithm is not efficient
branch and bound algorithm for 0/1 variables
NP-hard problems: no exact algorithm in polynomial time is known.
e.g. Traveling salesman problem
Heuristics are usually polynomial algorithms tailored to the specific problem structure
Operations Scheduling

Graph showing the comparison of $n^2$ and $2^n$ for values of $n$ from 1 to 10.
In a Machine Scheduling problem \( n \) orders or jobs \((j=1,\ldots,n)\) have to be assigned to \( m \) machines. For each order or job \( j \) the following data are given:

- \( r_j \): release date of the order \( j \)
  - are all orders available at \( a_j=0 \) (known a priori) then we have a static problem otherwise a dynamic problem.

- \( d_j \): Due date of order \( j \)

- \( p_{ji} \): Processing time of order \( j \) on machine \( i \)

Are all the inputs known a priori, then we have a deterministic model, otherwise we have a stochastic model (stochastic release dates or stochastic processing times)
Sequences

An order \( j \) can be divided in \( g_j \) different tasks (Arbeitsgänge) \( A_{j1}, \ldots, A_{jg_j} \) divided, they have to be processed in a fixed ordering \( \rightarrow \) task ordering (Arbeitsgangfolge). The task ordering is given due to technological reasons.

There is for each task \( A_{jh} \) of an order \( j \) a specific machine required \( \mu_{jh} \). The temporal sequence of the tasks assigned to the different machines is defined as machine ordering (Maschinenfolge) \( \mu_j = (\mu_{j1}, \ldots, \mu_{jg_j}) \) af task \( j \). The machine ordering is defined through technological requirements.

The sequences, in which the different tasks of the orders have to be processed on one machine is defined as order sequence (Auftragsfolge) on machine \( i \). It can happen that different orders want to be processed at the same machine at the same time. The order sequence is not given \( \rightarrow \) This is the decision of the planning situation.

A temporal assignment of orders to machines is called (feasible) scheduling plan (Ablaufplan). A plan is feasible when all the sequence restrictions are fulfilled and all further restrictions are fulfilled.
each order consists of $g_j = 3$ tasks

**Example: static Jobshop-Problem with 3 machines and 3 orders:**
these orders have to be processed in the sequence $A_{j1}, A_{j2}, A_{j3}$

<table>
<thead>
<tr>
<th>order</th>
<th>machine $\mu_{j1}$</th>
<th>processing time $t_{ji}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1  3  3  2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3  3  2  3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3  4  1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>order</th>
<th>tasks $A_{jh}$</th>
<th>Machine number $\mu_{jh}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1  1  2  3</td>
<td>1  1  2  3</td>
</tr>
<tr>
<td>2</td>
<td>2  2  3  1</td>
<td>2  2  3  1</td>
</tr>
<tr>
<td>3</td>
<td>3  2  1  3</td>
<td>3  2  1  3</td>
</tr>
</tbody>
</table>

Order 2 is processed on M2, next on M3 and then on M1

the first task (on machine 2) requires 2 time units
The specification with respect to task sequence or machine sequence can be illustrated in the following machine sequence graph (Maschinenfolgegraph). Every node number is the machine number $\mu_{jh}$, for the task $h$ of the order $j$. 

Machine sequence graph (Maschinenfolgegraph): Angabe

- Every node represents the machine $i = \mu_{jh}$

Diagram:

- Arbeitsgang (task)
  - $h=1$: j=1 → 1 → 2 → 3
  - $h=2$: j=1 → 2 → 3 → 1
  - $h=3$: j=3 → 2 → 1 → 3

- Auftrag (order)
Determining the order sequence (Auftragsfolge) for each machine i, represents in which sequence the different tasks j=1,2,3, have to be processed.

Within the machine sequence graph the nodes with the same machine number i can be connected with additional arcs. This produces a path. The resulting graph is the machine sequence graph (Ablaufgraph).

The picture illustrates the machine sequence graph for the example above: The orders on machine 1 are processed in the sequence 1,3,2. On Machine 2 in the sequence 3,2,1 and on machine 3 in Sequence 2,1,3.
In Gantt-Charts the processing times are drawn at the abscissa (x-axis) the machines (machine oriented Gantt chart) or the orders (order oriented Gantt chart) are drawn at the ordinate (y-axis).

There exists machine oriented Gantt charts (mainly used) and order oriented Gantt charts.
Here all tasks under consideration of the sequence restrictions are scheduled. The green fields are the idle times of the machines (in the machine oriented gantt chart) and the waiting times of the machines (in the order oriented gantt chart).
Semiactive sequence plans have the property that there exists no task where the beginning can be shifted forward without violating the machine sequence or to change the order sequence.

**Beispiel: (machine oriented Gantt-chart)**
For every feasible sequence plan a semiactive sequence plan exists. This plan is easy to construct: all the task should be shifted left. The above sequence plan is semi-active but it is nevertheless very bad.

**Active Sequence Plans:**

no task can scheduled earlier without delaying the begin of another task

for producing an active sequence plan only the order sequence can be changed (not the task ordering/machine ordering).

each active plan is also semiactive
active sequence plans
(machine oriented Gantt chart)

*Obiges Beispiel:* change order sequence on machine 2

![Gantt chart illustration](image-url)
Framework and Notation (Pinedo)

**Processing time** \( (p_{ij}) \) The \( p_{ij} \) presents the processing time of job \( j \) on machine \( i \).

**Release date** \( (r_j) \) The release date \( r_j \) of job \( j \) may also be referred to as the ready state. The earliest time at which job \( j \) can start processing.

**Due date** \( (d_j) \) The due date \( d_j \) of job \( j \) represents the committed shipping or completion date (i.e., the date the job is promised to the customer). Completion of a job after its due date is allowed, but then a penalty is incurred.

**Weight** \( (w_j) \) The weight \( w_j \) of a job \( j \) is basically the priority factor.

A scheduling problem is described by a triplet \( \alpha|\beta|\gamma \)
α – field: describes the machine environment

Single machine (1)
Identical machines in parallel (Pm)
Machines in parallel with different speed (Qm)
Flow shop (Fm) – There are m machines in series. Each job has to be processed on each one of the m machines. All jobs have to follow the same route – the have to be processed first on machine 1 and then on machine 2,…

Job shop (Jm) – in a job shop each job has ist own predetermined route to follow.
β-field

Release dates ($r_j$) – the job can not start its processing before its release date.

Preemtions (prmp) – Preemtions imply that it is not necessary to keep a job on a machine, once started, until ist completion.

Precedence constraints (prec) one or more jobs may have to be completed before another job is allowed to start.

Sequence dependent setup times ($s_{jk}$) The sequence dependent setup time between processing job $j$ and job $k$. 
**γ-field**

**Common objectives**
- total flowtime
- total tardiness
- makespan
- maximum tardiness
- number of tardy jobs

if not all jobs are equally important weights should be introduced

minimizing total completion time is equivalent to minimizing total flowtime or minimizing total tardiness
Operations Scheduling

Scheduling Theory (Background)

Jobs are

- activities to be done
- processing time known
- in general continuously processed until finished (preemption not allowed)
- due date
- release date
- precedence constraints
- sequence dependent setup time
- processed by at most one-machine at the same time
Machines (resources)

single machine, parallel machines

flow shop:
  each job must be processed by each machine exactly once
  all jobs have the same routing
  a job cannot begin processing on the second machine until it
  has completed processing on the first

assembly line

job shop:
  each job may have a unique routing

open shops:
  job shops in which jobs have no specific routing
  re-manufacturing and repair
Operations Scheduling

Measures

profit, costs

it is difficult to relate a schedule to profit and cost

**regular measure** is a function of completion time

function only increases if at least one completion time in schedule increases

**Data:**

n = number of jobs to be processed
m = number of machines
p_{ik} = time to process job i on machine k
r_i = release date of job i
d_i = due date of job i
w_i = weight of job i relative to the other jobs
Operations Scheduling

\[ C_i = \text{the completion time} \]

\[ F_i = C_i - r_i, \text{ the flowtime} \]

\[ L_i = C_i - d_i, \text{ lateness of job } i \]

\[ T_i = \max\{0, L_i\}, \text{ tardiness of job } i \]

\[ E_i = \max\{0, -L_i\}, \text{ earliness of job } i \]

\[ \delta_i = 1, \text{ if job } i \text{ is tardy } (T_i > 0) \]

\[ \delta_i = 0, \text{ if job } i \text{ is on time } (T_i = 0) \]

\[ C_{\text{max}} = \max_{i=1,n}\{C_i\}, \text{ makespan} \]

\[ L_{\text{max}} = \max_{i=1,n}\{L_i\}, \text{ maximum lateness} \]

\[ T_{\text{max}} = \max_{i=1,n}\{T_i\}, \text{ maximum tardiness} \]
Operations Scheduling

Algorithms:

exact algorithms often based on (worst case scenario) enumeration (e.g. Branch and Bound, Dynamic Programming)

heuristic algorithm judged by quality (difference to the optimal solution) and efficacy (computational effort)

worst-case bounds are desirable to motivate use of a certain heuristic
Operations Scheduling

Consider the following four-job, three-machine job-shop scheduling problem:

<table>
<thead>
<tr>
<th>Job</th>
<th>Op.1</th>
<th>Op.2</th>
<th>Op.3</th>
<th>Release Date</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4/1</td>
<td>3/2</td>
<td>2/3</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>4/1</td>
<td>4/3</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>3/3</td>
<td>2/2</td>
<td>3/1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>3/2</td>
<td>3/3</td>
<td>1/1</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

Assume the following sequences:

- 2-1-4-3 on M1
- 2-4-3-1 on M2
- 3-4-2-1 on M3
Operations Scheduling

Gantt Chart (machine oriented)

Gantt Chart (job oriented)
Operations Scheduling

\[ C_1 = 14, C_2 = 11, C_3 = 13, C_4 = 10 \]

The makespan is

\[ C_{\text{max}} = \max \{C_1, C_2, C_3, C_4\} = \max \{14, 11, 13, 10\} = 14 \]

The total flowtime is

\[ \sum_i F_i = 14 + 11 + 13 + 10 = 48 \]
The lateness and the tardiness of a job:

\[ L_1 = 14 - 16 = -2 \]
\[ L_2 = 11 - 14 = -3 \]
\[ L_3 = 13 - 10 = 3 \]
\[ L_4 = 10 - 8 = 2 \]

The total lateness is

\[ \sum_i L_i = (-2) + (-3) + 3 + 2 = 0 \]

The total tardiness is

\[ \sum_i T_i = 0 + 0 + 3 + 2 = 5 \]

The maximum tardiness is

\[ T_{\text{max}} = \max\{0,0,3,2\} = 3 \]

Tardy jobs have \( \delta_i = 1 \), so

\[ \begin{align*}
T_1 &= 0 \Rightarrow \delta_1 = 0 \\
T_2 &= 0 \Rightarrow \delta_2 = 0 \\
T_3 &> 0 \Rightarrow \delta_3 = 1 \\
T_4 &> 0 \Rightarrow \delta_4 = 1
\end{align*} \]

\[ N_T = 2 \]
Operations Scheduling

Single Machine Scheduling
Minimizing Flowtime

Problem data

<table>
<thead>
<tr>
<th>Job $i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_i$</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Sequence: 1-2-3-4-5
Total Flowtime=?

\[ F = p_1 + (p_1 + p_2) + (p_1 + p_2 + p_3) + \ldots + (p_1 + p_2 + \ldots + p_n) \]

\[ F = np_1 + (n-1)p_2 + \ldots + p_n \]
Theorem. SPT sequencing minimizes total flowtime on a single machine with zero release times.

Proof. We assume an optimal schedule is not an SPT sequence.

\[ p_i > p_j \]

\[
\begin{align*}
TF(S) &= TF(B) + (t+p_i) + (t+p_i+p_j) + TF(A) \\
TF(S^{'}) &= TF(B) + (t+ p_j) + (t+ p_j +p_i) + TF(A) \\
TF(S)-TF(S^{'}) &= p_i - p_j > 0
\end{align*}
\]
SPT-rule ⇒ sequence: 2-4-3-1-5

\[ C_1 = 11 \]
\[ C_2 = 2 \]
\[ C_3 = 7 \]
\[ C_4 = 4 \]
\[ C_5 = 15 \]

Total flowtime = total completion time = 39

SPT rule also minimizes
- total waiting time
- mean # of jobs waiting (mean work in progress)
- total lateness

Why?
Operations Scheduling

Minimize weighted Flow-time: \[ \sum_{i=1}^{n} w_i F_i \]

weighted SPT (WSPT): order ratios \[ \frac{p_i}{w_i} \] (nondecreasing)

exact algorithm for weighted flow-time with zero release time (completion time)
Weighted Flowtime

WSPT scheduling

\[ w_1 = 1, w_2 = 4, w_3 = 3, w_4 = 1, w_5 = 3 \]

the processing-time-to-weight ratio gives: 4; 0.5; 1; 2; 1.33

the WSPT sequence is the following: 2-3-5-4-1

\[
C_1 = 15 \\
C_2 = 2 \\
C_3 = 5 \\
C_4 = 11 \\
C_5 = 9
\]

the value of weighted flowtime is

\[
\sum_{i=1}^{5} w_i F_i = 76
\]
Maximal Tardiness and Maximal Lateness

due date oriented measure
earliest due date sequence (EDD)

EDD minimizes
Maximal Tardiness and
Maximal Lateness

<table>
<thead>
<tr>
<th>Job i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due date</td>
<td>16</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Proc. Time</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

EDD-sequence: 5-3-4-2-1
Tardiness of the jobs is (0, 0, 2, 1, 0)
Operations Scheduling

Number of Tardy Jobs

Hodgson’s algorithm

Step 1. Compute the tardiness for each job in the EDD sequence. Set $N_T=0$, and let $k$ be the first position containing a tardy job. If no job is tardy go to step 4.

Step 2. Find the job with the largest processing time in positions 1 to $k$.

\[
\text{Let } p_{[j]} = \max_{i=1,k} p_{[i]} \text{ then } j^* = [j]
\]

Step 3. Remove job $j^*$ from the sequence, set $N_T=N_T+1$, and repeat Step 1.

Step 4. Place the removed $N_T$ jobs in any order at the end of the sequence.

This sequence minimizes the number of tardy jobs
Consider the previous example:
EDD-sequence: 5-3-4-2-1

Step 1: The tardiness is \((0, 0, 2, 1, 0)\) \(\Rightarrow\) Job 4 in the third position is the first tardy job;
Step 2: The processing times for jobs 5, 3 and 4 are 4, 3, 2, respectively; largest processing time for job 5
Step 3: Remove job 5, goto step 1
Step 1: EDD-sequence is 3-4-2-1; completion times \((3, 5, 7, 11)\) and tardiness \((0, 0, 0, 0)\) \(\Rightarrow\) Go to step 4
Step 4: schedule that minimizes the number of tardy jobs is 3-4-2-1-5 and has only one tardy job: Job 5
Minimize the weighted number of tardy jobs!

NP-hard Problem

Heuristic approach: processing-time-to-weight ratio (not exact!)

Consider the previous example with the following weights:

\[ w_1 = 1, w_2 = 4, w_3 = 3, w_4 = 1, w_5 = 3 \]

EDD-sequence was 5-3-4-2-1

**Step 1** first tardy job is job 4

**Step 2** the processing-time-weight-ratio for jobs 5, 3 and 4 are 4/3, 3/3 and 2/1

**Step 3** Remove job 4

**Step 1** EDD-sequence is 5-3-2-1 with no tardiness

**Step 4** new schedule 5-3-2-1-4 has one tardy job: job 4 with weight 1
Operations Scheduling

Minimize Flowtime with no tardy jobs

for all jobs to be on time, the last job must be on time

schedulable set of jobs contain all jobs with due dates greater than or equal to the sum of all processing times

Start from the end and choose the job with the largest proc time among the schedulable jobs, schedule this job last, remove from the list and continue

Optimal algorithm! (corresponding alg. For weighted flowtime is only heuristic)

Problem data

<table>
<thead>
<tr>
<th>Job i</th>
<th>p i</th>
<th>p 1</th>
<th>p 2</th>
<th>p 3</th>
<th>p 4</th>
<th>p 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>due date</td>
<td>16</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
Operations Scheduling

Step 1: Sum of the processing time is 15
   Job 1 has a due-date greater to 15 \(\Rightarrow\) schedule x-x-x-x-1

Step 2: Sum of the remaining processing-times is 11
   Job 5 has a larger processing time \(\Rightarrow\) schedule x-x-x-5-1

Step 3: remaining processing time is 7
   All remaining jobs have due dates at least that big
   \(\Rightarrow\) choose the one with the largest processing time \(\Rightarrow\) x-x-3-5-1

Step 4: Continue \(\Rightarrow\) 2-4-3-5-1
Minimizing total Tardiness

general single-machine tardiness problem is NP-hard

Heuristic approach for the weighted problem (Rachamadugu/Morton)

if all jobs are tardy, minimizing weighted tardiness is equivalent to minimizing weighted completion time, which is accomplished by the WSPT sequence.

Weight-to-processing-time ratio is used

Slack of job i, \( S_i = d_i - (p_i + t) \) where \( t \) is the current time
A job should not get full WTPTR “credit” if its slack is positive

$$S_i^+ = \max\{0, S_i\}$$

Average processing time of the jobs:

$$p_{av} = \frac{1}{n} \sum_{i=1}^{n} p_i$$

Ratio of the slack to the average processing time of jobs:

$$S_i^+/p_{av}$$

which is the number of average job lengths until job j is tardy

Weight of a job is discounted by an exponential function:

$$\exp(-S_i^+/\kappa p_{av})$$
Operations Scheduling

Define the priority of job i by

\[
\gamma_i = \left( \frac{w_i}{p_i} \right) e^{-\left[ \frac{S_i^+}{\kappa \cdot p_{av}} \right]}
\]

\( \kappa \) is a parameter of the heuristic to be chosen by the user (e.g. \( \kappa = 2 \))

Sequence jobs in descending order of priorities.
Operations Scheduling

Rachamadugu and Morton (1982) R&M Heuristics:

The owner of Pensacola Boat Construction has currently 10 boats to construct;

If PBC delivers a boat after the delivery date, a penalty proportional to both the value of the boat and the tardiness must be paid.

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(i)</td>
<td>8</td>
<td>12</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>7</td>
</tr>
<tr>
<td>w(i)</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>d(i)</td>
<td>26</td>
<td>28</td>
<td>32</td>
<td>35</td>
<td>38</td>
<td>48</td>
<td>50</td>
<td>51</td>
<td>53</td>
<td>64</td>
</tr>
</tbody>
</table>

How should PBC schedule the work to minimize the penalty paid?
Operations Scheduling

Penalty is weighted tardiness where weights measure the value of the boat.

\( \kappa = 2 \quad p_{av} = 9 \)

Calculate:

Job1:

\[
\gamma_1 = \left( \frac{W_1}{p_1} \right) e^{-\left[ S_1^+ / (\kappa \ p_{av}) \right]} = \left( \frac{4}{8} \right) e^{-\left[ (26 - 8) / (2 \times 9) \right]} = 0,5 e^{-1} = 0,18
\]
### Operations Scheduling

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p(i))</td>
<td>8</td>
<td>12</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>7</td>
</tr>
<tr>
<td>(w(i))</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>(d(i))</td>
<td>26</td>
<td>28</td>
<td>32</td>
<td>35</td>
<td>38</td>
<td>48</td>
<td>50</td>
<td>51</td>
<td>53</td>
<td>64</td>
</tr>
<tr>
<td>(\frac{w(i)}{p(i)})</td>
<td>0.5</td>
<td>0.08</td>
<td>1</td>
<td>0.5</td>
<td>0.33</td>
<td>0.36</td>
<td>0.56</td>
<td>0.82</td>
<td>0.62</td>
<td>0.14</td>
</tr>
<tr>
<td>(S^+(i))</td>
<td>18</td>
<td>16</td>
<td>26</td>
<td>25</td>
<td>35</td>
<td>37</td>
<td>41</td>
<td>40</td>
<td>40</td>
<td>57</td>
</tr>
<tr>
<td>(\frac{S^+(i)}{k_{p_{av}}})</td>
<td>1</td>
<td>0.89</td>
<td>1.44</td>
<td>1.39</td>
<td>1.94</td>
<td>2.06</td>
<td>2.28</td>
<td>2.22</td>
<td>2.22</td>
<td>3.17</td>
</tr>
<tr>
<td>Priority</td>
<td>0.18</td>
<td>0.03</td>
<td>0.24</td>
<td>0.12</td>
<td>0.05</td>
<td>0.05</td>
<td>0.06</td>
<td>0.09</td>
<td>0.07</td>
<td>0.01</td>
</tr>
</tbody>
</table>

### Jobs

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>gamma_i</td>
<td>0.24</td>
</tr>
<tr>
<td>(p_i)</td>
<td>6</td>
</tr>
<tr>
<td>C_i</td>
<td>6</td>
</tr>
<tr>
<td>d_i</td>
<td>32</td>
</tr>
<tr>
<td>T_i</td>
<td>0</td>
</tr>
<tr>
<td>w_i</td>
<td>6</td>
</tr>
<tr>
<td>w_i T_i</td>
<td>0</td>
</tr>
</tbody>
</table>
Minimizing Earliness and Tardiness with a Common Due-Date

\[ Z = \sum_{i=1}^{n} (E_i + T_i) \]

this is not a regular measure
assume common due date: \( d_j = D \)

Number jobs in LPT sequence:
choose \( j^* = \frac{n}{2} \) or \( \frac{n}{2} + 0.5 \)

if \( p_1 \geq p_2 \geq \cdots \geq p_n \) then the following sequence
is \( p_1 + p_3 + \cdots + p_{j^*} \leq D \)

optimal: \( 1 - 3 - 5 - 7 - \ldots - n - \ldots - 6 - 4 - 2 \)
Example: 10 Jobs with common due-date 80

<table>
<thead>
<tr>
<th>Jobs</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>proc Time</td>
<td>8</td>
<td>18</td>
<td>11</td>
<td>4</td>
<td>15</td>
<td>5</td>
<td>23</td>
<td>25</td>
<td>10</td>
<td>17</td>
</tr>
</tbody>
</table>
if $p_1 + p_3 + \cdots + p_{j^*} > D$ then apply a heuristic (by Sundararaghavan & Ahmed, 1984)

**Step 0:** Set $B = D; A = \sum_{i=1}^{n} p_i - D; k = b = 1; a = n$, use the LPT sequence

**Step 1:** If $B > A$: assign job $k$ to position $b$

$b := b + 1$

$B := B - p_k$

else assign job $k$ to position $a$

$a := a - 1$

$A := A - p_k$

**Step 2:** $k := k - 1$; if $k \leq n$ go to step 1.
Problems with non-zero release time

Non-zero release times typically makes scheduling problems much harder, e.g. SPT does in general not minimize total flowtime.

**Heuristic Approach:**
At each time \( t \) determine the set of **schedulable jobs**: jobs that have been released but not yet processed.

Choose from the schedulable jobs according to some rule (e.g. SPT for minimizing flowtime)
## Operations Scheduling

### Preemption allowed:

<table>
<thead>
<tr>
<th>j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>12</td>
<td>2</td>
<td>0</td>
<td>11</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>p</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

- **t=0**: 3
- **t=2**: 1
- **t=3**: C
- **t=4**: 2
- **t=6**: C
- **t=9**: C
- **t=10**: 2
- **t=11**: 1
- **t=12**: 6
- **t=18**: C
- **t=24**: C
Operations Scheduling

Minimizing makespan with non-zero release time and tails
Given n jobs with release times $r_i$, processing times $P_i$, and tails $n_i$

Schrage Heuristics:
Start at $t=0$
1. Determine schedulable jobs
2. If there are schedulable jobs select the job $j^*$ among them with the largest tails, otherwise $t=t+1$ goto 1.
3. Schedule $j^*$ at $t$
4. If all jobs have been scheduled stop, otherwise set $t = t + P_{j^*}$, goto 1.
Schrage Heuristics Example: 6 jobs with release times and tails

<table>
<thead>
<tr>
<th>j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_j</td>
<td>12</td>
<td>2</td>
<td>0</td>
<td>11</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>p_j</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>n_j</td>
<td>21</td>
<td>9</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

Minimize makespan!
Denote by $SJ$ the set of schedulable jobs and by $S$ the scheduled sequence

Step 1. $t = 0$, $SJ = \{3\}$, $S = <3>$, $t = 3$, $C_{\text{max}} = 5$

Step 2. $t=3$, $SJ = \{2\}$, $S = <3-2>$, $t = 7$, $C_{\text{ma}} = 16$

Step 3. $t = 9$, $SJ = \{5\}$, $S = <3-2-5>$, $t = 11$, $C_{\text{ma}} = 18$

Step 4. $t=11$, $SJ = \{4, 6\}$, $S = <3-2-5-6>$, $t = 13$, $C_{\text{ma}} = 23$

Step 5. $t=13$, $SJ = \{1, 4\}$, $S = <3-2-5-6-1>$, $t = 21$, $C_{\text{ma}} = 42$

Step 6. $T=21$ $SJ = \{4\}$, $S = <3-2-5-6-1-4>$, $t = 27$, $C_{\text{ma}} = 42$

Schrage heuristic is in general not optimal, e.g. B&B model can be used as an exact algorithm
Minimizing Set-Up Times

sequence-dependent set-up times

the time to change from one product to another may be significant and may depend on the previous part produced

\[ p_{ij} = \text{time to process job } j \text{ if it immediately follows job } i \]

Examples:

- electronics industry
- paint shops
- injection molding

minimizes makespan

problem is equivalent to the traveling salesman problem (TSP), which is NP-hard.
SST(=shortest set-up time) heuristic

A metal products manufacturer has contracted to ship metal braces each day for four customers. Each brace requires a different set-up on the rolling mill:

<table>
<thead>
<tr>
<th>Job</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>∞</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>∞</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>6</td>
<td>∞</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>4</td>
<td>∞*</td>
<td>∞</td>
</tr>
</tbody>
</table>

*Job C cannot follow job D, because of quality problems

SST-heuristic:

**Step 1** starting arbitrarily by choosing one Job: A

**Step 2** B has the smallest set-up time following A; ⇒ A-B

**Step 3** C has the smallest set-up time of all the remaining jobs following B; ⇒ A-B-C

**Step 4** D is the last remaining job; ⇒ A-B-C-D-A with a makespan of 3 + 4 + 2 + 5 = 14
A regret based Algorithm

makespan must be at least as big as the n smallest elements

reduced matrix

row reduction

column reduction

sum of reduced costs = lower bound for TSP

find reduced matrix!

<table>
<thead>
<tr>
<th>Job</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>∞</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>∞</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>6</td>
<td>∞</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>4</td>
<td>∞*</td>
<td>∞</td>
</tr>
</tbody>
</table>
The reduced matrix has a zero in every row and column what happens if we do not choose j to follow i 
regret: lower bound on not choosing j to follow i

<table>
<thead>
<tr>
<th>Job</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>∞</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>∞</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>5</td>
<td>∞</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>∞*</td>
<td>∞</td>
</tr>
</tbody>
</table>
Regret heuristic

Find the cycle sequence that minimizes the set-up time.

Set-up times

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>18</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>∞</td>
<td>9</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>18</td>
<td>∞</td>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>∞</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>1</td>
<td>13</td>
<td>17</td>
<td>∞</td>
</tr>
</tbody>
</table>

Solution: TSP model – regret heuristic

Step 0  C(max) = 0 and L = 1
Step 1  Reduce the matrix:

Reduced matrix

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>∞</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>9</td>
<td>∞</td>
<td>4</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>∞</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>0</td>
<td>12</td>
<td>16</td>
<td>∞</td>
<td>1</td>
</tr>
</tbody>
</table>

19
**Operations Scheduling**

**Step 2** Calculate the regret

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>15</td>
<td>0(0)</td>
<td>0(4)</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>∞</td>
<td>4</td>
<td>5</td>
<td>0(5)</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0(9)</td>
<td>9</td>
<td>∞</td>
<td>4</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>0(1)</td>
<td>∞</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>0(17)</td>
<td>12</td>
<td>16</td>
<td>∞</td>
<td>19</td>
</tr>
</tbody>
</table>

**Step 3** Choose the largest regret : 17

**Step 4** Assign a job pair: Job 2 immediately follows job 5 (5-2)

L = 1+1;
We prohibit 2-5
Step 1 Reduce the matrix

\[ C_{\text{max}} = 19 + 4 + 1 = 24 \]

Reduced Matrix

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>∞</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>∞</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

Step 2 Calculate the regret

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>0(0)</td>
<td>0(1)</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0(1)</td>
<td>1</td>
<td>∞</td>
</tr>
<tr>
<td>3</td>
<td>0(9)</td>
<td>∞</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0(0)</td>
<td>∞</td>
<td>0(2)</td>
</tr>
</tbody>
</table>

Step 3 Choose the largest regret: 9
Step 4 Assign a job pair: 3-1
Prohibit 1-3

Step 1 Reduce the matrix: not possible

Matrix

<table>
<thead>
<tr>
<th>Job</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>∞</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>∞</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>
Step 2 Calculate regret

<table>
<thead>
<tr>
<th>Job</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\infty$</td>
<td>0(3)</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0(1)</td>
<td>1</td>
<td>$\infty$</td>
</tr>
<tr>
<td>4</td>
<td>0(0)</td>
<td>$\infty$</td>
<td>0(2)</td>
</tr>
</tbody>
</table>

Step 3 Choose the largest regret: 3

Step 4 Assign job pair: 1-4; partial sequence: 5-2, 3-1-4
Prohibit 4-1 and 4-3 (to keep 3-1-4-3 from being chosen)

Final Matrix

<table>
<thead>
<tr>
<th>Job</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>$\infty$</td>
</tr>
<tr>
<td>4</td>
<td>$\infty$</td>
<td>0</td>
</tr>
</tbody>
</table>

choose 2-3 and 4-5
-> sequence 3-1-4-5-2
the total set-up time is 24
Branch and Bound Algorithm

1. Using the regret heuristic construct a (sub-)tree where each node represents the decision to let \( j \) follow \( i \) (\( i \rightarrow j \)) or to prohibit that \( j \) follows \( i \) (\( i \not\rightarrow j \)).

2. For each node a lower bound for the makespan is inferred from the regret heuristic.

3. Once a solution is obtained from the regret heuristic this is an upper bound for the optimal makespan. All nodes where the lower bound is above that level are pruned.

4. If all but one final node are pruned (or no non-pruned node can be further branched) this final node gives the optimal solution.

5. If 4. does not hold start again with 1. at one of nodes which are not pruned and can still be branched.
Operations Scheduling

Branch and Bound Algorithm

All final nodes can be pruned:

opt. Solution has been found!
Single-Machine Search Methods

- Neighborhood Search
- Simulated Annealing
- Ant System
- Tabu Search

... 

**Neighborhood Search**

seed

Neighborhood

any heuristic can be used to produce an initial sequence
adjacent pairwise interchange (API):
  n-1 neighbors
  1-2-3-4-5-6-7-8-9
  1-2-3-4-6-5-7-8-9
Pairwise interchange (PI):
  n(n-1)/2 neighbors
  1-2-8-4-5-6-7-3-9
Insertion (INS)
  (n-1)^2 neighbors
  1-2-3-7-4-5-6-8-9
Evaluation function
Update function
Neighborhood search

Consider the following single-machine tardiness problem; Use the EDD sequence as the initial seed with an API neighborhood;

Data for neighborhood search

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing time</td>
<td>10</td>
<td>3</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Due-date</td>
<td>15</td>
<td>16</td>
<td>24</td>
<td>30</td>
<td>35</td>
<td>37</td>
</tr>
</tbody>
</table>

Step 1 Construct the EDD sequence and evaluate its total tardiness

Set $i = 1$ and $j = 2$

The EDD sequence $S^*: 1-2-3-4-5-6$; tardiness-vector (0, 0, 5, 7, 6, 14)
Operations Scheduling

Step 2 Swap the jobs in the $i$-th and $j$-th position in $S^*$; the sequence is $S'$ with tardiness $T'$. If $T' < T$, go to step 4.

Step 3 $j = j + 1$: If $j > n$: go to step 5. Otherwise, $i = j - 1$ and go to step 2.

Step 4 Replace $S^*$ with $S'$; $i = 1, j = 2$; go to step 2.

Step 5 Stop; $S^*$ is a local optimal sequence.
### Operations Scheduling

**Neighborhood search solution**

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Schedule</th>
<th>Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>i j</td>
<td>1 2 3 4 5 6</td>
<td>32</td>
</tr>
<tr>
<td>1 2</td>
<td>2 1 3 4 5 6</td>
<td>32</td>
</tr>
<tr>
<td>2 3</td>
<td>1 3 2 4 5 6</td>
<td>42</td>
</tr>
<tr>
<td>3 4</td>
<td>1 2 4 3 5 6</td>
<td>33</td>
</tr>
<tr>
<td>4 5</td>
<td>1 2 3 5 4 6</td>
<td>30</td>
</tr>
</tbody>
</table>

1 2 2 1 3 5 4 6 30
2 3 1 3 2 5 4 6 40
3 5 1 2 5 3 4 6 34
5 4 1 2 3 4 5 6 32
4 6 1 2 3 5 6 4 32

---

**Production Management**
Operations Scheduling

Single machine results

Flowtime - SPT (E)
Lateness - SPT (E)
Weighted Flowtime - WSPT (E)
Maximal Tardiness (Lateness) - EDD (E)
Nb. Of tardy jobs - Hodgson (E)
weighted nb. Of tardy jobs - modified Hodgson (H)
No jobs tardy/flowtime - modified SPT (E)
Tardiness - R&M (H)
weighted Tardiness - R&M (H)
makespan with non-zero release time and tails - Schrage (H)
Sequence dependent - SST (H), regret (H), B&B (E)
Parallel Machines

Scheduling decisions:

which machine processes the job
in what order

List Schedule

to create a schedule, assign the job on the list to the machine with the smallest amount of work assigned.

**Step 0.** Let $H_i=0$, $i=1,2,...,m$ be the assigned workload on machine $i$, $L=([1],[2],...,[n])$ the ordered list sequence, $C_j=0$, $j=1,2,...,n$, and $k=1$

**Step 1.** Let $j^*=L_k$ and $H_{i^*} = \min_{i=1,m}\{H_i\}$; Assign job $j^*$ to be processed on machine $i^*$, $C_{j^*}=H_{i^*}+p_{j^*}, H_{i^*}=H_{i^*}+p_{j^*}$

**Step 2.** Set $k=k+1$, if $k>n$, stop. Otherwise go to step 1.
Minimizing flowtime on parallel processors

Consider a facility with 3 identical machines and 15 jobs that need to be done as soon as possible; Processing times (after SPT):

<table>
<thead>
<tr>
<th>Job</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>15</td>
<td>19</td>
</tr>
</tbody>
</table>

Optimal schedule:

<table>
<thead>
<tr>
<th>Machine 1</th>
<th>Machine 2</th>
<th>Machine 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>j</td>
<td>p(j)</td>
<td>C(j)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>30</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>46</td>
</tr>
</tbody>
</table>

Total flowtime = 372
# Operations Scheduling

## Minimize the makespan

Use a longest processing time (LPT) first list; Assign the next job on the list to the machine with the least total processing time assigned.

### Optimal schedule:

<table>
<thead>
<tr>
<th>Machine 1</th>
<th>Machine 2</th>
<th>Machine 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>j</td>
<td>p(j)</td>
<td>C(j)</td>
</tr>
<tr>
<td>15</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>42</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>52</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>53</td>
</tr>
</tbody>
</table>

![Diagram showing the optimal schedule for machine 1, machine 2, and machine 3.](image)
Operations Scheduling

Flow shops

all jobs are processed in the same order

two machine makespan model: Johnson’s Algorithm

Bound on makespan:

\[
C_{\text{max}}^* \geq \max \left\{ \left( \min_{i=1}^{n} p_{i2} + \sum_{i=1}^{n} p_{i1} \right), \left( \min_{i=1}^{n} p_{i1} + \sum_{i=1}^{n} p_{i2} \right) \right\}
\]

Formulate Johnson’s Algorithm

For 2-machine Flow shops the optimal schedule is a **Permutation Schedule**, i.e. the job sequence is the same on every machine
Operations Scheduling

Makespan with more than two machines

Johnson’s algorithm will work in special cases, e.g. three machine problem where the second machine is dominated:

\[ p_{i2} \leq \max(\min p_{i1}, \min p_{i3}) \]

Formulate an artificial two machine problem with

\[ p'_{i1} = p_{i1} + p_{i2} \quad \text{and} \quad p'_{i2} = p_{i2} + p_{i3} \]

and solve it using the Johnson algorithm gives the optimal solution for the three machine problem
Heuristics for the m-machine problem

Cambell, Dudek and Smith (1970)

convert a m-machine problem into a two machine problem

how?

\[ p'_{i1} = \sum_{j=1}^{k} p_{ij} \quad \text{and} \quad p'_{i2} = \sum_{j=l}^{m} p_{ij} \]

Start with: \( k=1 \) and \( l=m \); then \( k=2 \) and \( l=m-1 \); until: \( k=m-1 \) and \( l=2 \)

m-1 schedules are generated

Use the best of these m-1 schedules
Operations Scheduling

Flow-shop heuristics

Processing data:

<table>
<thead>
<tr>
<th>Job</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>13</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>9</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>17</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>3</td>
<td>5</td>
<td>16</td>
</tr>
</tbody>
</table>

Use the CDS heuristic to solve this problem.

(1) i.) Use the Johnson’s algorithm only for M1 and M4:

<table>
<thead>
<tr>
<th>Job</th>
<th>M1</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>16</td>
</tr>
</tbody>
</table>

1-2-5-4-3, C_{max} = 88

Next combine M1 and M2 to pseudomachine 1 and M3 and M4 to pseudomachine 2.

<table>
<thead>
<tr>
<th>Job</th>
<th>PM1</th>
<th>PM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job</th>
<th>[j]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

5-2-3-1-4, C_{max} = 85

Finally combine M1, M2 and M3 to pseudomachine 1 and M2, M3 and M4 to pseudomachine 2.

<table>
<thead>
<tr>
<th>Job</th>
<th>PM1</th>
<th>PM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>48</td>
</tr>
<tr>
<td>3</td>
<td>39</td>
<td>26</td>
</tr>
<tr>
<td>4</td>
<td>31</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>24</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job</th>
<th>[j]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

5-1-2-3-4, C_{max} = 85
Operations Scheduling

Gantt Chart for the CDS schedule
Operations Scheduling

Gupta – Heuristic

Gupta (1972)

exact for 2-machine problem and 3-machine problem, where the 2nd machine is dominated

\[ e_i = \begin{cases} 1 & \text{if } p_{i1} < p_{im} \\ -1 & \text{if } p_{i1} \geq p_{im} \end{cases} \]

\[ s_i = \frac{e_i}{\min_{k=1,\ldots,m-1} \{p_{i,k} + p_{i,k+1}\}} \]

Sorting jobs with nonincreasing \( s_i \)

\((s_{[1]} \geq s_{[2]} \geq \ldots \geq s_{[n]})\)

<table>
<thead>
<tr>
<th>Job</th>
<th>p1+p2</th>
<th>p2+p3</th>
<th>p3+p4</th>
<th>min</th>
<th>ei</th>
<th>si</th>
<th>[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>19</td>
<td>8</td>
<td>8</td>
<td>1</td>
<td>0.12</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>30</td>
<td>36</td>
<td>22</td>
<td>1</td>
<td>0.05</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>22</td>
<td>17</td>
<td>17</td>
<td>-1</td>
<td>-0.06</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
<td>19</td>
<td>8</td>
<td>8</td>
<td>-1</td>
<td>-0.12</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>8</td>
<td>21</td>
<td>8</td>
<td>1</td>
<td>0.12</td>
<td>2</td>
</tr>
</tbody>
</table>
Operations Scheduling

Branch and Bound Approaches

machine based bounds

job based bounds

three machines

$H_j =$ current completion time of the last job scheduled on machine $j$

$U =$ set of unscheduled jobs

makespan on machine 1 must be at least:

$$C^*_{\text{max}} \geq H_1 + \sum_{i \in U} p_{i1} + \min_{i \in U} \{p_{i2} + p_{i3}\}$$

machine 2:

$$C^*_{\text{max}} \geq \max \left\{ \left( H_1 + \min_{i \in U} \{p_{i1}\} \right), H_2 \right\} + \sum_{i \in U} p_{i2} + \min_{i \in U} \{p_{i3}\}$$
Operations Scheduling

Machine 3:

\[
C^*_\text{max} \geq \max \left\{ \left( H_1 + \min_{i \in U} \{ p_{i1} + p_{i2} \} \right), \left[ H_2 + \min_{i \in U} \{ p_{i2} \} \right], H_3 \right\} + \sum_{i \in U} p_{i3}
\]

job oriented bounds:

\[
C_{\text{max}} \geq H_1 + \max_{i \in U} \left\{ \sum_{j=1}^{m} p_{ij} + \sum_{k \in U, k \neq i} \min \{ p_{k1}, p_{k3} \} \right\}
\]
Operations Scheduling

B&B algorithm for minimizing makespan in multi-machine Flow Shops

1. Create an initial incumbent solution, e.g. CDS heuristic upper bound
2. Starting at t=0 with a root node; branch the tree by generating a node for each schedulable jobs.
3. In each node calculate the lower bounds and prune the node if at least one exceeds the upper bound.
4. If a non-pruned final node exists at the lowest level take the corresponding solution as new incumbent, update the upper bound and do the corresponding pruning.
5. If all final nodes are pruned current incumbent is the optimal solution, otherwise branch at the node with the lowest lower bound and goto 3.
### Operations Scheduling

#### Makespan permutation schedule for a three-machine flow-shop

Processing data:

<table>
<thead>
<tr>
<th>Machine j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>17</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>12</td>
<td>9</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>18</td>
<td>13</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Solution:

Start with CDS algorithm: sequence: 1-2-3-4-5, $C_{\text{max}} = 65$

Initial lower bound:

**M1:** $C_{\text{max}}^* \geq H_1 + (p_{11} + p_{21} + p_{31} + p_{41} + p_{51})$

$$+ \min\{ p_{12} + p_{13}, p_{22} + p_{23}, p_{32} + p_{33}, p_{42} + p_{43}, p_{52} + p_{53} \}$$

$$= 0 + (1 + 10 + 17 + 12 + 11) + \min\{19, 30, 22, 19, 8\} = 51 + 8 = 59$$

**M2:** $C_{\text{max}}^* \geq \max\{[ H_1 + \min\{p_{11},p_{21},p_{31},p_{41},p_{51}\}],H_2\}$

$$+ ( p_{12} + p_{22} + p_{32} + p_{42} + p_{52} ) + \min \{ p_{13} , p_{23} , p_{33} , p_{43} , p_{53} \}$$

$$= \max\{[0 + \min\{1, 10, 17, 12, 11\}], 0\}$$

$$+ (13 + 12 + 9 + 17 + 3) + \min\{6, 18, 13, 2, 5\}$$

$$= 1 + 54 + 2 = 57$$
Operations Scheduling

M3: $C_{\text{max}}^* \geq \max\{H_1 + \min\{p_{11} + p_{12}, p_{21} + p_{22}, p_{31} + p_{32}, p_{41} + p_{42}, p_{51} + p_{52}\}\},$

$[H_2 + \min\{p_{12}, p_{22}, p_{32}, p_{42}, p_{52}\}], H_3\} + (p_{13} + p_{23} + p_{33} + p_{43} + p_{53})$

$= \max\{[0 + \min\{14, 22, 26, 29, 14\}],$

$[0 + \min\{13, 12, 9, 17, 3\}], 0\} + (6 + 18 + 13 + 2 + 5)$

$= \max\{14, 3, 0\} + 44 = 58$

Job-based bounds are the following:

$$C_{\text{max}} \geq H_1 + \sum_{j=1}^{3} p_{1j} + \sum_{k \in \{2, 3, 4, 5\}} \min\{p_{k1}, p_{k3}\}$$

J1: $C_{\text{max}}^* \geq H_1 + (p_{11} + p_{12} + p_{13})$

$+ (\min\{p_{21}, p_{23}\} + \min\{p_{31}, p_{33}\} + \min\{p_{41}, p_{43}\} + \min\{p_{51}, p_{53}\})$

$= 0 + (1 + 13 + 6) + (\min\{10, 18\} + \min\{17, 13\} + \min\{12, 2\} + \min\{11, 5\})$

$= 0 + 20 + (10 + 13 + 2 + 5) = 50$

Similarly, we have

J2: $C_{\text{max}}^* \geq 61$, J3: $C_{\text{max}}^* \geq 57$, J4: $C_{\text{max}}^* \geq 60$, J5: $C_{\text{max}}^* \geq 45$

LB: 61, UB: 65
UB = 65 (Gupta)
LB = 61 J2

Job 1
First

Job 2
Second

Job 3
Third

Job 4
Fourth

Job 5
Fifth

Solution (=LB): 65
1\textsuperscript{st} level: J2 at first place: \(H_1 = 10, H_2 = 22, H_3 = 40\)
\[U = \{1, 3, 4, 5\}\]
M1: \(C_{\text{max}}^* \geq 59\)
M2: \(C_{\text{max}}^* \geq 66\), which is greater than the upper bound; thus we fathom the node;
J3, J4 and J5 at first place: we can fathom all of them;

2\textsuperscript{nd} level: Consider Job 3: \(H_1 = 18, H_2 = 27, H_3 = 40, U = \{2, 4, 5\}\)
\[M1: C_{\text{max}}^* \geq 59\]
\[M2: C_{\text{max}}^* \geq 62\]
\[M3: C_{\text{max}}^* \geq 65\], so we fathom the job; only job 2 remains unfathomed;

3\textsuperscript{rd} level: Job 3: \(H_1 = 28, H_2 = 37, H_3 = 57, U = \{4, 5\}\)
\[M1: C_{\text{max}}^* \geq 59\]
\[M2: C_{\text{max}}^* \geq 61\]
\[M3: C_{\text{max}}^* \geq 64\]
Machine-bounds did not fathom the node; so we have to calculate job-based bounds:
\[J4: C_{\text{max}}^* \geq 64\]
\[J5: C_{\text{max}}^* \geq 49\]
\[\Rightarrow\] best bound = 64; thus create nodes for J4 and J5

4\textsuperscript{th} level: nodes J4 and J5 of level 3 will be fathomed; thus the algorithm is complete:
1-2-3-4-5 with a makespan of 65;
Operations Scheduling

Job Shops
- different routings for different jobs
- precedence constraints
- \((n!)^m\) possible schedules
Operations Scheduling

Two machine job shops

Jackson (1956)

minimize makespan

Machine A: \{AB\}, \{A\}, \{BA\}
Machine B: \{BA\}, \{B\}, \{A,B\}

Jackson’s algorithm

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route</td>
<td>BA</td>
<td>AB</td>
<td>BA</td>
<td>B</td>
<td>A</td>
<td>AB</td>
<td>B</td>
<td>BA</td>
<td>BA</td>
<td>AB</td>
</tr>
<tr>
<td>p(i)1</td>
<td>3</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>8</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>p(i)2</td>
<td>8</td>
<td>10</td>
<td>13</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Find a schedule that would finish all jobs as soon as possible!

Solution:
\{A\} = \{5\}, \{B\} = \{4,7\}, \{AB\} = \{2, 6, 10\} and \{BA\} = \{1, 3, 8, 9\}
Johnson’s algorithm for \{AB\}:

<table>
<thead>
<tr>
<th>Job</th>
<th>2</th>
<th>10</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(i)1</td>
<td>1</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>p(i)2</td>
<td>10</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

Johnson’s algorithm (reversed) for \{BA\}:

<table>
<thead>
<tr>
<th>Job</th>
<th>9</th>
<th>3</th>
<th>8</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(i)1</td>
<td>13</td>
<td>11</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>p(i)2</td>
<td>6</td>
<td>13</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

sequence for A: 2-10-6-5-9-3-8-1
sequence for B: 9-3-8-1-4-7-2-10-6

makespan: 67
Operations Scheduling

Dispatching

job shop scheduling

dispatching rules

Basic idea:

schedule an operation of a job as soon as possible
if more than one job is waiting to be processed by the same machine schedule the one with best priority

Define:

$A = \text{set of idle machines}$

$J_k = \text{the index of the last job scheduled on machine } k$

$U_k = \text{the set of jobs that can be processed on machine } k$

$H_k = \text{the completion time of the job currently processed on machine } k$

$u_{it} = \text{the priority of job } i \text{ at time } t$
Operations Scheduling

Step 0. Initialize: $t=0; H_k=0, k=1,2,...,m$;
$A=\{1,2,...,m\}; U_k=\{i|\text{operation 1 of } i \text{ is on machine } k, i=1,2,...,n\}; s_{ij}=c_{ij}=0$. Go to step 4.

Step 1. Increment $t$; Let
$$t = \min_{k=1,m;k \notin A} H_k, \text{ and } K = \{k \mid H_k = t\}$$

Step 2. Find the job or jobs that complete at time $t$ and the machine released. Set $A = A \cup K$.

Step 3. Determine the jobs ready to be scheduled on each machine;
Let $U_k=\{i|\text{job } i \text{ uses machine } k \text{ and all operations of job } i \text{ before machine } k \text{ are completed}\}, k=1,2,...,m$.
If $U_k=0$ for $k=1,2,...,m$, Stop.
If $U_k=0$ for $k \in A$, go to Step 1.
Operations Scheduling

Step 4. For each idle machine try to schedule a job; for each \( k \in A \) with \( U_k \neq 0 \),

let \( i^* \) be the job with the best priority: 
\[
 u_{i^*t} = \min_{i \in U_k} u_{it}
\]

Schedule job \( i^* \) on machine \( k \)

Set \( J_k = i, s_{i^*k} = t, c_{i^*k} = t + p_{i^*j(k)}, H_k = c_{i^*k} \)

Remove the scheduled job from \( U_k \)

\[
 U_k \leftarrow U_k - \{i\}
\]

and the machine from \( A \)

\[
 A \leftarrow A - \{k\}
\]

Go to Step 1
Operations Scheduling

Many priority measures possible:

- SPT
- FCFS
- MWKR (most work remaining)
- EDD
- EDD/OP
- SLACK, SLACK/OP

Critical ratio: slack/remaining time

...
Quick Closures: job-shop dispatch heuristic

Quick Closure has four machines in the shop: (1) brake, (2) emboss, (3) drill, (4) mill; The shop has currently orders for six different parts, which use all the four machines, but in a different order.
Processing time:

<table>
<thead>
<tr>
<th>Operation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>6/1</td>
<td>8/2</td>
<td>13/3</td>
<td>5/4</td>
</tr>
<tr>
<td>2</td>
<td>4/1</td>
<td>1/2</td>
<td>4/3</td>
<td>3/4</td>
</tr>
<tr>
<td>3</td>
<td>3/4</td>
<td>8/2</td>
<td>6/1</td>
<td>4/3</td>
</tr>
<tr>
<td>4</td>
<td>5/2</td>
<td>10/1</td>
<td>15/3</td>
<td>4/4</td>
</tr>
<tr>
<td>5</td>
<td>3/1</td>
<td>4/2</td>
<td>6/4</td>
<td>4/3</td>
</tr>
<tr>
<td>6</td>
<td>4/3</td>
<td>2/1</td>
<td>4/2</td>
<td>5/4</td>
</tr>
</tbody>
</table>

Finish all six parts as soon as possible!

Solution: We use a dispatch procedure with MWKR as the priority.
Operations Scheduling

Step 1 \[ t = 0, \ H_1 = H_2 = H_3 = H_4 = 0, \ A = \{1, 2, 3, 4\}, \ U_1 = \{1, 2, 5\}, \ U_2 = \{4\}, \ U_3 = \{6\}, \ U_4 = \{3\}; \ s_{ij} = c_{ij} = 0, \ i = 1, 2, 3, 4, 5, 6; \text{ and } j = 1, 2, 3, 4; \text{ Go to step 4 } \]

Step 4 \[ u_{10} = -(6+8+13+5) = -32, \ u_{20} = -12, \ u_{50} = -17; \text{ thus } s_{11} = 0, \ c_{11} = 0 + 6 = 6, \ H_1 = 6. \]
Remove job 1 from \( U_1, \ U_1 = \{2, 5\} \) and machine 1 from \( A, \ A = \{2, 3, 4\}. \)
Set \( k = 2; \) there is only one job in \( U_2 \) so we schedule it on machine 2; \( i^* = 4, \ s_{41} = 0, \ c_{41} = 5, \ H_2 = 5, \ U_2 = \{\}, \text{ and } A = \{3, 4\}. \)
We schedule \( J_6 \) and \( J_3 \) on \( M_3 \) and \( M_4 \) (tab: \( t = 0 \) row). Go to step 1.

Step 1 \[ t = \min_{k=1,m:k \in A} H_k = \min\{6, 5, 4, 3\} = 3, \text{ and } K = \{k \setminus H_k = 3\} = \{4\}; \ H_k \text{ min is bold in the table; } \]

Step 2 \( J_3 \) completes at time 3 on \( M_4 \), so \( i^3 = \{i \setminus J_k = i, \ k \in K\} = \{3\}, \ K = \{4\}, \) and \( A = \{} U \{4\} = \{4\}, \text{ (tab: } t = 3 \text{ row) } \)

Step 3 \( U_1 = \{2, 5\}, \ U_2 = \{3\}, \ U_3 = U_4 = \{\}; \text{ Since no jobs are waiting for } M_4, \text{ no jobs can be scheduled to start at time 3; go to step 1 etc. } \)
# Operations Scheduling

<table>
<thead>
<tr>
<th>$t$</th>
<th>$it$</th>
<th>$K$</th>
<th>$A$</th>
<th>$U1$</th>
<th>$U2$</th>
<th>$U3$</th>
<th>$U4$</th>
<th>$H1$</th>
<th>$H2$</th>
<th>$H3$</th>
<th>$H4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>$1,2,3,4$</td>
<td>$1,2,5$</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2,5</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>3</td>
<td>3,4</td>
<td>2,5,6</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>$2,3,4$</td>
<td>$2,4,5,6$</td>
<td>3</td>
<td>6</td>
<td>13</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>$1,3,4$</td>
<td>$2,4,5,6$</td>
<td>1</td>
<td>16</td>
<td>13</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>2</td>
<td>$2,3,4$</td>
<td>$2,3,5,6$</td>
<td>1</td>
<td>16</td>
<td>21</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>1</td>
<td>$1,3,4$</td>
<td>$2,3,5,6$</td>
<td>4</td>
<td>19</td>
<td>21</td>
<td>31</td>
<td>4</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>5</td>
<td>1</td>
<td>$1,4$</td>
<td>$2,3,6$</td>
<td>5</td>
<td>23</td>
<td>21</td>
<td>31</td>
<td>4</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>2</td>
<td>2,4</td>
<td>3,6</td>
<td>5</td>
<td>1</td>
<td>23</td>
<td>25</td>
<td>31</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>1</td>
<td>1,4</td>
<td>3,6</td>
<td>2</td>
<td>1</td>
<td>25</td>
<td>25</td>
<td>31</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>25</td>
<td>6,5</td>
<td>1,2</td>
<td>$1,2,4$</td>
<td>3</td>
<td>2,6</td>
<td>1</td>
<td>5</td>
<td>31</td>
<td>29</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>29</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>31</td>
<td>30</td>
<td>31</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1,2</td>
<td>6</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>31</td>
<td>3,4</td>
<td>1,3</td>
<td>$1,2,3$</td>
<td>1,2,3</td>
<td>4,6</td>
<td>44</td>
<td>36</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>36</td>
<td>6</td>
<td>4</td>
<td>$1,2,4$</td>
<td>2,3,5</td>
<td>4</td>
<td>44</td>
<td>40</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
<td>4</td>
<td>$1,2,4$</td>
<td>2,3,5</td>
<td>4</td>
<td>44</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>44</td>
<td>1</td>
<td>3</td>
<td>$1,2,3,4$</td>
<td>$2,3,5$</td>
<td>1</td>
<td>48</td>
<td>49</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>48</td>
<td>2</td>
<td>3</td>
<td>$1,2,3$</td>
<td>3,5</td>
<td>2</td>
<td>52</td>
<td>49</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>49</td>
<td>1</td>
<td>4</td>
<td>$1,2,4$</td>
<td>5</td>
<td>2</td>
<td>52</td>
<td>52</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>52</td>
<td>3,2</td>
<td>3,4</td>
<td>$1,2,3,4$</td>
<td>5</td>
<td>56</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>56</td>
<td>5</td>
<td>3</td>
<td>$1,2,3,4$</td>
<td>5</td>
<td>56</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>
Operations Scheduling

M1
M2
M3
M4

0 10 20 30 40 50
Problems with two orders (Akers)

We consider Flow Shop and Job Shop Problems with 2 orders (Objectives minimization of cycle time):

Example: [aus Domschke, Scholl und Voß (1993)] static flow shop with four machines, machine ordering $\mu_1 = \mu_2 = (1, 2, 3, 4)$ and the following processing times:

<table>
<thead>
<tr>
<th>$j$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{1j}$</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>$t_{2j}$</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

The problem can be represented in a two-dimensional coordinate system one axis corresponds to one of the two orders. The point of origin $Q=(0,0)$ represents the time 0 (release time).
Problems with two orders II

\[ S_j = \Sigma t_{ji} \] denotes the earliest possible completion time of the order \( j \), when order \( j \) is produced without preemption - the earliest completion time starting from time 0 – without preemption. The points \( Q \) and \( S=(S1,S2) \) describe a rectangle (an operation rectangle).

The intervall \([0, S_1] \) can be divided in \( m \) disjunctive intervalls. These intervals depend on the machine sequence \( \mu_1 \) of the first order, the machine sequence of the first oder is the following \( \mu_{11},...,\mu_{1m} \). The length of each interval is the processing time \( t_{1j} \) of each task \( j \).

The same is valid for the second task and the y-axis can also be divided between \([0, S_2] \) in disjunctive intervals.

For each machine \( i \) a rectangle is defined which is denoted as conflict rectangle (Konfliktfeld). In the following example the conflict fields are drawn for the above example in yellow.

For the minimal cycle time \( Z^* \) we can calculate \( Z = \max \{S_1, S_2\} \) as lower bound and \( S_1+S_2 \) as trivial upper bound. In our example \( Z = 8 \) and upper bound = 16.
The method of Akers finds a shortest path in the operations field from the Origin Q und the point S under the following conditions:

The route consists of
- horizontal
- vertical
- diagonal sections.

The yello conflict fields are not crossed.
Method of Akers II

Under diagonal sections we understand section with a slope 1, this represents a parallel processing of both orders on different machines. Horizontal sections denote the processing of order 1 and vertical sections denote the processing of order 2.

The length of a path from Q to S results that each movement to the right or to the left means that one time unit is consumed.

⇒ different ways are possible (in our example 3). The two ways of the length $Z=11$ are permutation plans (order 1 is always processed before order 2 or order 2 is always processed before order 1 on the different machines). The optimal plan with $Z=10$ is not a so called permutation plan, as an overtaking of the orders take place⇒what is also visible in the Gantt-chart.
Method of Akers

It is necessary, that all the different paths between Q and S are checked systematically. Therefore a directed graph $G = (V, e, c)$ is constructed. The node set $V$ includes the start node $Q$ and the termination node $S$ and for each machine the north-west corner point and the south-east corner point of the conflict field. The set of arcs $E$, the costs $c$ for each arc, and the shortest distance from $Q$ to $S$ are calculated simultaneously.
Method of Akers IV

Starting from each node \( p=(p_1,p_2) \) with a (current) shortest distance \( d_p \) from \( Q \) to \( p \), diagonal moves in direction \( S \) are performed \( \rightarrow \) as long as the border of the operation rectangle is touched or a conflict rectangle is touched.

We need a solution to cross the conflict rectangle \( i \) an arc beginning from \( p \) to the north-west corner \( q \) and an arc to the south-east corner \( r \). The costs are the maximum of the \( x \)- or \( y \)-distance.

\[
c(p, q) = q_2 - p_2
\]
\[
c(p, r) = r_1 - p_1
\]
Example: Job Shop-Problem $[J5 \mid n = 2 \mid Z]$. The processing times and the machine ordering are given as follows.

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{1i}$</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>$t_{2i}$</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>h</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_{1i}$</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>$\mu_{2i}$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

The sums of the processing times are the following $S = (17, 15)$.

The conflict fields of the machines concerning the job ordering are drawn. When we have Job Shop problems the conflict fields are not drawn „diagonally“.
Method of Akers VII

The shortest path algorithm of Dijkstra provides the following result:

Solution:
Method of Akers VIII

Order

2
i=1 | i = 2 | i = 3 | i = 4 | i = 5
1 | i=4 | i = 1 | i = 3 | i = 2 | i = 5

Zeit

2 4 7 10 13 15 18 21

Machine

5 | j = 1 | j = 2
4 | j = 1 | D | j = 2 | F
3 | B | j = 1 | j = 2
2 | j = 2 | j = 1
1 | j = 2 | j = 1

Zeit

2 4 7 10 13 15 18 21
The method of Akers can also be applied when release dates are used \( a_j \neq 0 \) (move conflicts towards North-East). It can also be used when other objective functions are used.
Shortest route
(Problem des kürzesten Weges)

Calculation of the shortest path within a network starting from Source (Quelle) to Sink (Senke) T.
Basis algorithm for many other logistical problems
many different exact methods.
Tree-algorithms: calculate the shortest path from one node (source) to all other nodes (and construct a kind of tree).

classical tree algorithms are based on the idea of dynamic programming. These algorithms mark in each iteration step one further node.

Methods, which generate shortest paths to all nodes.
Dijkstra Algorithms

\[ d_{ij} = \text{length of the direct arc from node } i \text{ to } j \text{ (if such an arc exists, } d_{ij} = \infty \text{ otherwise} \]

**Initialization: \( n = 0 \):**

All nodes are temporarily marked, and the shortest path from the source \( D[i] \) with the direct distance \( d_{0i} \) are calculated. [Only node 0 is finally marked].

If there exist no direct arc from \( O \) to \( i \), then \( D[i] = \infty \). The temporarily direct predecessor node is the source: \( V[i] = 0 \)
Iterationsschritt $n$

1. mark the temporarily marked node, which has the minimum distance $D[i]$. This is the $n$-th next node from the origin $O$. The shortest distance is $D[i]$ and the predecessor is $V[i]$.

2. Check all temporarily marked nodes $j$, which are reachable from $i$ (via a direct arc). When $D[i] + d_{ij} < D[j]$ then the new path via $i$ to $j$ is shorter than the so far known best path to $j$. Update $D[j] = D[i] + d_{ij}$ and $V[j] = i$.

3. When $T$ is finally marked, (or when all nodes are finally marked). The algorithm is terminated.
### Dijkstra's Method

#### Lösung: O – A – B – D – T

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0, O</td>
<td>2, O</td>
<td>5, O</td>
<td>4, O</td>
<td>$\infty$, O</td>
<td>$\infty$, O</td>
<td>$\infty$, O</td>
</tr>
<tr>
<td>1</td>
<td>2, O</td>
<td>2, O</td>
<td>5, O</td>
<td>4, O</td>
<td>$\infty$, O</td>
<td>$\infty$, O</td>
<td>$\infty$, O</td>
</tr>
<tr>
<td>2</td>
<td>4, A</td>
<td>4, O</td>
<td>9, A</td>
<td>$\infty$, O</td>
<td>$\infty$, O</td>
<td>$\infty$, O</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4, O</td>
<td>8, B</td>
<td>7, B</td>
<td>$\infty$, O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8, B</td>
<td>8, B</td>
<td>7, B</td>
<td>$\infty$, O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8, B</td>
<td>8, B</td>
<td>7, B</td>
<td>$\infty$, O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8, B</td>
<td>8, B</td>
<td>14, E</td>
<td>$\infty$, O</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Production Management
Solution

Two shortest paths with length 13:

Production Management
Method of Bellman

Select out of all marked (visited) nodes, the not visited neighbors with the shortest distance.

Mark the node in the next iteration step with the shortest distance from the source.
Verfahren von **Bellmann II**

<table>
<thead>
<tr>
<th>Iteration n</th>
<th>Vergebene Knoten, die unmittelbar mit einem noch nicht Vergebenen zu verbinden sind</th>
<th>Kürzeste Verbindung zu einem noch nicht vergebenen Knoten</th>
<th>Gesamte Entfernung zum Ursprung</th>
<th>n-ter nächster Knoten $i$</th>
<th>Kürzeste Entfernung $D[i]$</th>
<th>Unmittelbarer Vorgänger $V[i]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O</td>
<td>A</td>
<td>2</td>
<td>A</td>
<td>2</td>
<td>O</td>
</tr>
<tr>
<td>2</td>
<td>O</td>
<td>C</td>
<td>4</td>
<td>C</td>
<td>4</td>
<td>O</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>B</td>
<td>4</td>
<td>B</td>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>D</td>
<td>9</td>
<td>A</td>
<td>9</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>E</td>
<td>7</td>
<td>E</td>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>E</td>
<td>8</td>
<td>E</td>
<td>8</td>
<td>E</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>D</td>
<td>9</td>
<td>D</td>
<td>8</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>D</td>
<td>8</td>
<td>D</td>
<td>8</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>D</td>
<td>8</td>
<td>E</td>
<td>8</td>
<td>E</td>
</tr>
<tr>
<td>6</td>
<td>D</td>
<td>T</td>
<td>13</td>
<td>T</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>T</td>
<td>14</td>
<td>E</td>
<td>14</td>
<td>T</td>
</tr>
</tbody>
</table>

Production Management
Verfahren von Bellmann III

1. Lösung: O - A - B - D - T
2. Lösung: O - A - B - E - D - T
Operations Scheduling

Shifting Bottleneck Procedure
heuristic to minimize makespan for multiple machine job shops

Main idea:

1. for each job on each machine calculate the minimal amount of time needed before and after the processing of this job generates minimal makespan problem with release times and tails
2. for each machine solve this problem for each machine (e.g. Schrage heuristic) and determine the machine with the maximal makespan (bottleneck machine)
3. Fix the found sequence on the bottleneck machine, update release times and tails on the remaining machines and repeat 2. for the remaining machines until schedules for all machines have been determined
Shifting Bottleneck Procedure Example:
3 machines (M1, M2, M3), 3 jobs (1,2,3)

Job routings:
1: M1-M2-M3
2: M2-M3-M1
3: M2-M1-M3

Processing times:

<table>
<thead>
<tr>
<th>i</th>
<th>p_{ik}</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Machine-Flow-Graph:

Job 1

Job 2

Job 3

s
Operations Scheduling

Problems with release times and tails for each machine:

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>r</td>
<td>0</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>p</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>n</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Operations Scheduling

Schrage heuristic gives the following solutions for the three machines:

Machine 2 is bottleneck with $C_2 = 11$

Fix sequence on machine 2
Update release times and tails on M1 and M3:

**M1:**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_j$</td>
<td>0</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>$p_j$</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$n_j$</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**M3:**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_j$</td>
<td>9</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>$p_j$</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>$n_j$</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
Schrage heuristic for M1, M3:

Both machines could be considered the bottleneck with $C=12$, fix sequence on M1.
Operations Scheduling

Updated machine-flow-graph:

update release time and tails and apply Schrage to M3. This gives with $C_{\max} = 12$
Operations Scheduling

Work to do: 8.3abcde, 8.4, 8.5, 8.6, 8.10, 8.14, 8.16, 8.18 (with the following due dates: 42, 50, 12, 63, 23, 34, 36, 42, 54, 32) 8.30ab, 8.32abc, 8.36ab, 8.43, 8.44, 8.49ab, 8.51ab, 8.56, 8.57 (apply shifting bottleneck procedure)

Minicase: Ilana Designs
Metaheuristics for Operations Scheduling
Relevante Literatur


Construction Heuristic
Improvement Heuristic (Local Search)
Metaheuristic
Metaheuristic
Metaheuristic
Metaheuristic?

A heuristic usually exploits the available problem structure
Positiv: the exploitation of problem specific knowledge helps
Negativ: a heuristic which is developed for one problem type X is often not suitable for another problem type Y.

The solution quality provided by a heuristic is very often not suitable to provide a reasonable solution for the problem at hand.

How can neighborhoods of a local optimum reasonably checked.

→ Metaheuristics help to improve this situation – they can avoid the negative aspects of heuristics.
[MH]: Historical development

Prototyp: Exchange Operators (*move, etc.*)

Prinzip:

- start with a feasible solution.
- apply different local modifications, as long as there is an improvement in the objective function value (monotone evolution of the solution quality)
[MH]: Historical development

Classical improvement methods have two properties:

The solution quality will not deteriorate – es wird nicht mit schlechterwerdenden Lösungen gearbeitet.

The solution remains always feasible – infeasible solutions are not allowed.
[MH]: Historical development

Local improvement methods stop when a local optimum will be found (wrt to allowed modifications)

Solution quality (and computation time) depends on the complexity ("richness") of the allowed modifications in every iteration.
[MH]: More historical background

1983: Kirkpatrick, Gelatt and Vecchi publish their famous paper in *Nature* on *simulated annealing*.
A probabilistic local search algorithm capable of overcoming local optima and with convergence properties.
Renewed interest for the development of new types of heuristics (*metaheuristics*).
Metaheuristics

Concept introduced by Glover (1986)
Generic heuristic solution approaches designed to control and guide specific problem-oriented heuristics
Generally inspired from analogies with natural processes
Rapid development over the last 15 years
[MH]: Well known Metaheuristics

Nature inspired methods
  Simulated Annealing (SA)

Evolutionary Algorithms (EA)
  Genetic Algorithms (GA)
  Memetic Algorithms (MA)

Neural Nets (NN)

Ant Colony Optimization (ACO)

Methods based on Local Search
  Tabu Search (TS)
  Variable Neighborhood Search (VNS)
[MH]: Other Metaheuristics

Adaptive Memory Procedures (AMP)

Threshold Acceptance methods (TA)

Greedy Randomized Adaptive Search Procedure (GRASP)

Scatter Search (SS)

...
[MH]: Components

Metaheuristics have three characteristics:

Lokale Search: classical improvement steps
But: **deteriorating and infeasible solutions are allowed.**

By using classical heuristics it can happen that we get trapped in a local optimum.

Deteriorating solutions are allowed, to find a better local Optimum (hopefully the global optimum)

Lösungsrecombination:
Many solutions are generated and recombined.
Learning: successful actions will be repeated.
[MH]: Variable Neighborhood Search

Mladenovic and Hansen (1997)
Another local search method.
In its simplest version (*Var. Neighborhood Descent*), it involves alternating between different neighborhood structures, whenever a local optimum is encountered. More sophisticated versions include rules for escaping local optima more effectively.
[MH]: Basic VNS

Initialization:
  set of neighborhood structures
  initial solution
  stopping condition

Repeat until stopping condition is reached
  Shaking
    (set of neighborhood structures $N_k$, for $k = 1, \ldots, k_{\text{max}}$)
  Local Search
  Move or not
[MH]: Basic VNS (2)

Shaking
  generate $x_1$ at random from the $k^{th}$ neighborhood of $x$:
  $x_1 \in N_k(x)$

Local Search
  apply local search to $x_1 \rightarrow$ local optimum $x_2$

Move or not
  if $x_2$ is better than $x$, move to new optimum:
  $x \leftarrow x_2$ and $k \leftarrow 1$
  otherwise $k \leftarrow k + 1$
[VNS]: Lokale Suche
[VNS]: Shaking – k=1
[VNS]: Shaking – k=2
[VNS]: Shaking – k=3
[VNS]: Move or not? – deteriorating solutions are allowed!
[MH]: Ant Colony Optimization

Colorni, Dorigo and Maniezzo (1991)
Inspired from an analogy with the way colonies of ants forage for food.
Ants foraging for food deposit *pheromone* on the path between the food source they have found and their nest.
Over time, the quantity of pheromone depends on the quality of food, the distance and the number of ants traveling over a path.
Ant Colony Optimization

Analogy:
- value of a solution $\leftrightarrow$ quality of food
- distance $\leftrightarrow$ cost of individual elements
  leading to a good solution
- artificial pheromone is added to modify the cost of individual elements
  leading to good solutions

Artificial ants repeatedly construct solutions using the costs of the individual elements modified by pheromone levels which are dynamically adjusted.
Real Ants

Ants:
- social insects
- self organized collective behavior
- emergence of complex dynamics

Interesting phenomena:
- division of labor
- task allocation
- cementary organization
Real Ants (contd.)

More interesting phenomena:
  - cooperative transport

Ant Foraging

Ant Foraging:
  - Many species (e.g. Lasius niger) show trail laying – trail following behavior
Real Ants (contd.)

Trail laying - Trail following:

simple local rules

ants lay aromatic essence called pheromone where they walk for orientation purposes

ants follow trails according to intensity \(\Rightarrow\) reinforcement of 'trails' with high pheromone concentration

pheromone on unused trails evaporates
Real Ants (contd.)

Trail laying - Trail following:
complex global dynamics
⇒ e.g. The binary bridges experiment:
Ant System

introduced by Dorigo, Maniezzo & Colorni

population based meta-heuristic

each ant in the population constructs a solution to the optimization problem

after all ants are done, a memory is updated (artificial pheromone)

solution construction and memory update are repeated until a prespecified stopping criterion is met
Ant System (contd.)

Solution construction

a number of decisions are taken probabilistically

e.g.: TSP

first application

Solution construction

mechanism:

Nearest Neighbor heuristic
Ant Colony Optimization – ACO
Ant Colony Optimization – ACO

visibility
pheromone
trail

\[ p_{hi} = \tau_{hi} \cdot \eta_{hi} \]

\[ p_{hj} = \tau_{hj} \cdot \eta_{hj} \]
Ant System (contd.)

Solution construction

Decision making is based on both, a constructive heuristic rule (a local quality criterion $s$) and an adaptive memory (a global quality criterion $\tau$)

\[
P_{ij} = \begin{cases} 
\frac{[s_{ij}]^\beta \cdot [\tau_{ij}]^\alpha}{\sum_{h\in\Omega_i} [s_{ih}]^\beta \cdot [\tau_{ih}]^\alpha} & \text{if } h \in \Omega_i \\
0 & \text{otherwise}
\end{cases}
\]

where $\Omega_i$ is the set of feasible alternatives
AS - Roulette wheel selection

<table>
<thead>
<tr>
<th></th>
<th>probability</th>
<th>cum. Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_{12}</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>p_{13}</td>
<td>0.02</td>
<td>0.22</td>
</tr>
<tr>
<td>p_{14}</td>
<td>0.44</td>
<td>0.66</td>
</tr>
<tr>
<td>p_{15}</td>
<td>0.12</td>
<td>0.78</td>
</tr>
</tbody>
</table>

Randomly generated number = 0.21

Move from 1 to 3 is selected
Ant System (contd.)

Pheromone update

all edges are updated at the end of each iteration

\[ \tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \sum_{m \in M} \Delta \tau_{ij}^m, \]

where

is the amount of reinforcement an element receives

\[ \Delta \tau_{ij}^m = \frac{1}{L^m}, \text{ if } (i, j) \in T^m \]
\[ 0, \text{ otherwise} \]
Rank based Ant System

distinguish between good and poor solutions
increase exploitation

⇒ modified (rank based) pheromone update
Rank based Ant System (contd.)

Pheromone update

\[ \tau_{ij} := \rho^* \tau_{ij} + \sum_{s=1}^{k-1} (k - s) \Delta \tau_{ij}^s + k\Delta \tau_{ij}^* \]

\( \Delta \tau_{ij}^s = 1 / L^s \)

\( \Delta \tau_{ij}^* = 1 / L^* \)

\( k \) ... number of elitists

\( L^* \), \( L^s \) ... objective value global (*)-, s-best solution
Project Scheduling [Pinedo, 2005, Hans/Hurink]

Project definition:

A complex and large scale one-of-a-kind product or service, made up by a number of component activities (jobs), that entails a considerable financial effort and must be time-phased, i.e. scheduled, according to specified precedence and resource requirements (Hax and Candea, 1984)
Project properties

Project goals: quality, time, costs, customer satisfaction
Network of activities/jobs
Limited resource capacity
Project life-cycle:
  Order acceptance
  Engineering and process planning
  Material and resource scheduling
  Project execution
  Evaluation & service
Project examples

Construction
Production
Management
Research
Maintenance
Installation, implementation
Hierarchical planning

Strategic
- Strategic resource planning

Tactical
- Rough-cut process planning
- Rough-cut capacity planning

Tactical/operational
- Engineering & process planning
- Project scheduling

Operational
- Detailed scheduling

Production Management
Project scheduling topics

Project representation (precedence graph)
Critical Path Method (CPM)
Resource-constrained project scheduling
  standard problem
  methods
  extensions
Project representation / precedence graph

Rules for “job on node” networks:
- network contains no **directed cycles**
- **event numbering**
- network contains no **redundant arcs**
Project representation example 1

<table>
<thead>
<tr>
<th>Job</th>
<th>Duration</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4,5</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4,5</td>
</tr>
</tbody>
</table>

“job on node”-representation:

```
1 2 4 6
2 3 1
3 1 1
4 2
5 2 3
6 1 4,5
7 3 4,5
```

“job on arc”-representation:
Project representation example 2

"job on node"-representation:

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1,2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2,3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

"job on arc"-representation:
Project scheduling

Without resource constraints relatively easy (EK and VK)

With resource constraints very complex:

when jobs share resources with limited availability, these jobs cannot be processed simultaneously \( \Rightarrow \) draw disjunctive arcs

Example 4.6.1:

<table>
<thead>
<tr>
<th>Jobs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(j)</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>R(1,j)</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>R(2,j)</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>
Project scheduling without resource constraints: critical path method (CPM)

Critical job: job without slack

\[
\begin{align*}
S_j' &= \text{earliest possible starting time of job } j \\
C_j' &= \text{earliest possible completion time of job } j \\
C_j'' &= \text{latest possible completion time of job } j \\
\text{slack}_j &= C_j'' - p_j - S_j'
\end{align*}
\]

Critical path = chain of critical jobs

Forward procedure
Backward procedure
Project scheduling without resource constraints: critical path method (CPM)

Critical path method initialization:
- determine earliest starting time for all jobs
- determine latest completion time for all jobs
- determine which jobs have no slack

2 CPM solution methods:
- Forward procedure
- Backward procedure
Critical path method

Forward CPM procedure

STEP1: For each job that has no predecessors:

\[ S'_j = 0 \]
\[ C'_j = p_j \]

STEP2: compute for each job \( j \):

\[ S'_j = \max_{k \to j} C'_k \]
\[ C'_j = S'_j + p_j \]

STEP3: \( C_{\text{max}} = \max_j C'_j \)

Backward CPM procedure

STEP1: For each job that has no successors:

\[ C''_j = C_{\text{max}} \]
\[ S''_j = C_{\text{max}} - p_j \]

STEP2: compute for each job \( j \):

\[ C''_j = \min_{j \to \text{all } k} S''_k \]

STEP3: Verify that:

\[ 0 = \min_j S''_j \]
\[ S''_j = C''_j - p_j \]
Critical path method example

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1,2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2,3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Source (project start)

Sink (project start)
Critical path method example (cont.)

<table>
<thead>
<tr>
<th>Job</th>
<th>$p(j)$</th>
<th>Predecessors</th>
<th>$S'$</th>
<th>$C''$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1,2</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2,3</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Critical job: $C'' = C' = S' + p$

Legend:
- $j$: job
- $S'$: start time
- $C''$: completion time

Production Management
Suppose jobs require a resource:

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>Predecessors</th>
<th>S'</th>
<th>C''</th>
<th>R(1,j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1,2</td>
<td>3</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2,3</td>
<td>3</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>

resource requirements
Suppose:

\[ R_1 = 4 \]

Cmax increases by 2
Resource-Constrained Project Scheduling Problem (RCPSP)

- $n$ jobs $j=1,...,n$
- $N$ resources $i=1,...,N$
- $R_k$: availability of resource $k$
- $p_j$: duration of job $j$
- $R_{kj}$: requirement of resource $k$ for job $j$
- $P_j$: (immediate) predecessors of job $j$
Goal: minimize makespan:

Restrictions:
- no job may start before $T=0$
- precedence relations
- finite resource capacity

\[ C_{\text{max}} = \max_j C'_j \]
RCPSP example

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>S'</th>
<th>C''</th>
<th>R(1,j)</th>
<th>R(2,j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1,2</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2,3</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

R₁ = 4

R₂ = 2
Disjunctive arcs

Suppose $R_1 = 4$. These jobs cannot be performed simultaneously:

Job 1 & 3
Job 3 & 6
Job 4 & 5
Job 5 & 6
Priority-rule-based scheduling

Generation scheme
  serial
  parallel

Priority rule
  latest finish time
  minimum slack

Sampling procedure
Serial scheduling method

Each stage represents a job $\Rightarrow n$ stages
completed set of jobs: scheduled jobs
decision set: jobs of which all predecessors have been scheduled
remaining set: other jobs

procedure:
1. Start with an empty schedule
2. Select job from decision set with highest priority, and schedule it as early as possible
3. Repeat step 2 if the decision set is not empty
Serial scheduling method example (1)

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>R(1,j)</th>
<th>v(j) (priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

$R_1 = 2$

Decision set
Serial scheduling method example (2)

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>R(1,j)</th>
<th>v(j) (priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ R_1 = 2 \]

Decision set

\[ \begin{array}{c}
1 \\
2 \\
3 \\
\end{array} \]
Serial scheduling method example (3)

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>R(1,j)</th>
<th>v(j) (priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

$R_1 = 2$

Decision set
Serial scheduling method example (4)

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>R(1,j)</th>
<th>v(j) (priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ R_1 = 2 \]
Parallel scheduling method

At most n stages
Each stage n represents:
1. partial schedule
2. schedule time $t_n$
3. four disjoint sets of jobs:
   - completed set: scheduled jobs, completed at $t_n$
   - active set: scheduled jobs, not completed yet
   - decision set: all unscheduled jobs, that could be scheduled
   - remaining set: all unscheduled jobs, that cannot be scheduled
Parallel scheduling method

procedure:
1. Start with an empty schedule.
2. Let T be the first time in which an unscheduled job may start. Let D be the collection of jobs that may be started on T, and of which all predecessors are scheduled
3. Select the job from D with the highest priority, and schedule it from time T
4. Repeat step 2 if there remain jobs to be scheduled
Parallel scheduling method example (1)

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>R(1,j)</th>
<th>v(j) (priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Decision set

$T = 0$

$R_1 = 2$
Parallel scheduling method example (2)

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>R(1,j)</th>
<th>v(j) (priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Decision set

$T = 0$

$R_1 = 2$

Production Management
Parallel scheduling method example (3)

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>R(1,j)</th>
<th>v(j) (priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Decision set

$T = 3$

$R_1 = 2$
Parallel scheduling method example (4)

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>R(1,j)</th>
<th>v(j) (priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ R_1 = 2 \]
Priority-rule-based scheduling

**Generation scheme**
- serial
- parallel

**Priority rule**
- latest finish time
- minimum slack

**Sampling procedure**
Priority-rule-based scheduling: priority rules

Latest finish time (LFT) priority rule:

$$v_j = -C'_j$$

Minimum slack (MS) priority rule

$$v_j = -(C'_j - p_j - S_j^{'*})$$

current earliest starting time
**MS priority rule example (1)**

**serial scheduling scheme**

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>R(1,j)</th>
<th>S'(j)</th>
<th>C''(j)</th>
<th>v(j) (priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>-2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
v_j = -(C'_j - p_j - S'_j^*)
\]

\[R_1 = 2\]
MS priority rule example (2)

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>R(1,j)</th>
<th>S'(j)</th>
<th>C''(j)</th>
<th>v(j) (priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>-2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ v_j = -(C'' - p_j - S'_j) \]

\[ R_1 = 2 \]
**MS priority rule example (3)**

\[
v_j = -(C''_j - p_j - S'_j^*)
\]

<table>
<thead>
<tr>
<th>Job</th>
<th>p(j)</th>
<th>P(j)</th>
<th>R(1,j)</th>
<th>S'(j)</th>
<th>C''(j)</th>
<th>v(j) (priority)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

R₁ = 2
Priority-rule-based scheduling

Generation scheme
- serial
- parallel

Priority rule
- latest finish time
- minimum slack

Sampling procedure
Priority-rule-based scheduling

Multi-pass priority-rule-based heuristics
multi-priority rule procedures
1 scheduling scheme, $x$ priority rules
$\Rightarrow$ generate $x$ schedules, keep the best found
sampling procedures
1 scheduling scheme, 1 priority rule
jobs are randomly selected from the decision set
Sampling procedure

Random sampling
all jobs have the same probability:
\[ j \in D : v_j = \frac{1}{|D|} \]

Biased random sampling
job with highest priority has highest probability, however not proportionally

Regret-based random sampling
job with highest priority proportionally has the highest probability
Biased random sampling

Probability that job $j$ is selected ($P_j$):

- first sort jobs on non-increasing priority
- $[j]$ is the position of job $j$ in the list

$\Rightarrow$

$$P_j = C \cdot \alpha^{[j]},$$

where:

$$C = \frac{1}{\sum_i \alpha^i}$$

$0 < \alpha \leq 1$ (bias factor)

$\Rightarrow P_j = 1/n$

$\alpha = 1$: random sampling

$\alpha \approx 0$: deterministic
Biased random sampling (example)

D = \{1, 2, 3\}, \upsilon_1 \geq \upsilon_3 \geq \upsilon_2, \text{ suppose } \alpha = 0.5

\Rightarrow

P_1 = C \cdot (0.5)^1 = 0.5 \cdot C
P_2 = C \cdot (0.5)^3 = 0.125 \cdot C
P_3 = C \cdot (0.5)^2 = 0.25 \cdot C
P_1 + P_2 + P_3 = 1 \Rightarrow C = \frac{8}{7}

P_j = C \cdot \alpha^{[j]},

\text{where} : C = \sum_{i}^{1} \alpha^i

0 < \alpha \leq 1 (\text{bias factor})
Regret-based random sampling

Regret of job = difference between priority value and lowest overall priority value:

Probability that job is selected:

\[ w_j = v_j - \min_i v_i \]

\[ P_j = C \cdot (w_j + 1)^\alpha \quad (\Rightarrow P_j > 0) \]

where:

\[ C = \frac{1}{\sum_i (w_i + 1)^\alpha} \quad \alpha = \text{bias factor} \ (\geq 0) \]

\[ \alpha = 0 : \text{random sampling}; \quad \alpha = \infty : \text{deterministic} \]

\[ \rightarrow P_j = 1/n \]
Regret-based random sampling (example)

\[ D = \{1,2\}, \ v_1 = 2; \ v_2 = 1, \suppose: \alpha = 0.5 \]

\[
\min_{i \in D} v_i = v_2 = 1
\]

\[
w_1 = v_1 - \min_{i \in D} v_i = 2 - 1 = 1 \implies P_1 = C \cdot (w_1 + 1)^{0.5} = C \cdot \sqrt{2}
\]

\[
w_2 = v_2 - \min_{i \in D} v_i = 1 - 1 = 0 \implies P_2 = C \cdot (w_2 + 1)^{0.5} = C
\]

\[
P_1 + P_2 = 1 \implies C = \frac{1}{\sqrt{2 + 1}} \approx 0.41
\]

\[ \implies P_1 \approx 0.59 \text{ and } P_2 \approx 0.41 \]
Time/costs trade-off

Assumptions:
by allocating money (for additional resources) to jobs their processing time ($p_j$) can be reduced
linear relation between allocated money and $p_j$
minimum and maximum processing time $p_j^{\text{min}}, p_j^{\text{max}}$
c$_j$ = marginal costs of reducing $p_j$:
\[
c_j^{\text{min}} - c_j^{\text{max}}
\]
\[
\frac{p_j^{\text{max}} - p_j^{\text{min}}}{p_j^{\text{max}} - p_j^{\text{min}}}
\]

\[
costs(p_j) = c_j^{\text{max}} + c_j(p_j^{\text{max}} - p_j^{\text{min}})
\]

$c_0$ = fixed overhead costs per time unit
Time/costs trade-off heuristic

Definitions:

source and sink in precedence graph

critical path: longest path from source to sink

G_{cp} = sub-graph of critical path(s)

cut set: set of nodes in sub-graph G_{cp} whose removal results in disconnecting the source from the sink in the precedence graph

minimal cut set: if putting back 1 node in the graph connects the source to the sink
Time/costs trade-off heuristic

STEP 1: Set

\[ p_j = p_j^\text{max} \quad (\forall j) \]

Determine \( G_{cp} \).

STEP 2: Determine all minimum cut sets mcs in \( G_{cp} \).

Consider only those minimum cut sets of which all processing times \( p_j > p_j^\text{min} \).

If there is no such set: STOP

STEP 3: For each mcs compute the costs of reducing all \( p_j \) in mcs with one time unit

Let mcs* be the mcs with the lowest costs

If the lowest costs are \( < c_0 \) ⇒ apply the changes, revise \( G_{cp} \) and go to STEP 1
## Time/costs trade-off heuristic data

<table>
<thead>
<tr>
<th>Jobs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p_j)-(\text{max})</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>7</td>
<td>12</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>(p_j)-(\text{min})</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>(c_j)</td>
<td>7</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>
Time/costs trade-off heuristic example 4.4.1

longest path = 56 = 1 → 3 → 6 → 9 → 11 → 12 → 14

c_0 = fixed overhead costs per time unit = 6

minimum cut sets: \{1\}, \{3\}, \{6\}, \{9\}, \{11\}, \{12\}, \{14\}

minimum cut sets with lowest costs: \{11\} and \{12\}

c_{12} = 2 < c_0 = 6 \Rightarrow net savings of 4 \Rightarrow p_{12} = 7
Time/costs trade-off heuristic example 4.4.1

longest path = 55 : 1 → 3 → 6 → 9 → 11 → {12,13} → 14
minimum cut sets : {1}, {3}, {6}, {9}, {11}, {12,13}, {14}
minimum cut set with lowest costs : {11}
c_{11} = 2 < c_{0} = 6 \Rightarrow \text{net savings of 4}
⇒ p_{11} := 6
Time/costs trade-off heuristic example 4.4.1

longest path = 54 : 1 → 3 → 6 → 9 → 11 → \{12,13\} → 14

or : 1 → 2 → 4 → 7 → 10 → 12 → 14

minimum cut sets :
\{1\}, \{i, j\} : i \in \{2,4,7,10,12\} + j \in \{3,6,9,11\}, \{12,13\}, \{14\}

minimum cut set with lowest costs : \{2,11\}

\(c_2 + c_{11} = 4 < c_0 = 6 \Rightarrow\) net savings of 2 \(\Rightarrow p_2 := 5^*, p_{11} := 5\)
Time/costs trade-off heuristic example 4.4.1

job 2 hits minimum

longest path $= 53 : 1 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 11 \rightarrow \{12,13\} \rightarrow 14$
or $: 1 \rightarrow 2 \rightarrow 4 \rightarrow 7 \rightarrow 10 \rightarrow 12 \rightarrow 14$
minimum cut sets :
$\{1\}, \{i, j\} : i \in \{4,7,10,12\} + j \in \{3,6,9,11\}, \{12,13\}, \{14\}$
minimum cut set with lowest costs : $\{4,11\}$
$c_4 + c_{11} = 5 < c_0 = 6 \Rightarrow \text{net savings of } 1 \Rightarrow p_4 := 11, p_{11} := 4$
Time/costs trade-off heuristic example 4.4.1

longest path = 52 : 1 → 3 → 6 → 9 → 11 → \{12,13\} → 14

or : 1 → 2 → 4 → 7 → 10 → 12 → 14

minimum cut sets :
\{1\}, \{i, j\} : i \in \{4,7,10,12\} + j \in \{3,6,9\}, \{12,13\}, \{14\}

minimum cut set with lowest costs : \{4,6\}

\(c_4 + c_6 = 6 \neq c_0 = 6\) ⇒ no net savings ⇒ STOP