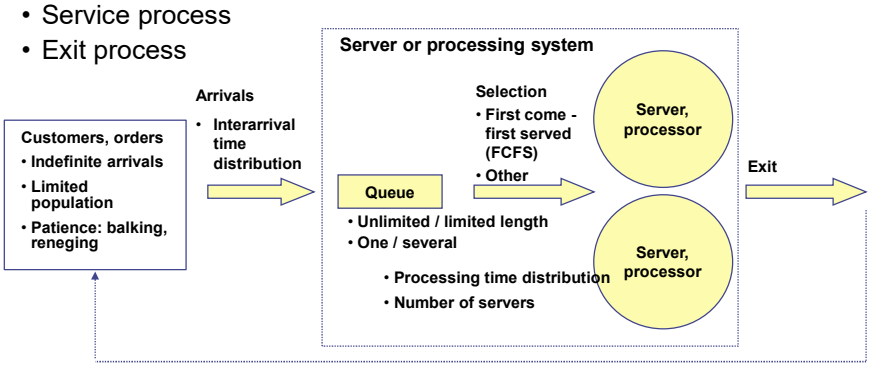
Queuing theory

Main components of a queuing system are:

- Customers
- Arrival process
- Queue
- Queuing discipline
- Service process
- Exit process



esko.niemi@aalto.fi

1

Queuing theory

Queuing theory mathematically examines queuing phenomena. Formulas are developed for various queuing processes and their arrival and processing time distributions. These processes can be further combined to form a network. The formulas can be used to calculate queue length and for example:

- Average number of customers (workpieces) in the process
- Average queuing time
- Average time in the system
- Probability for the system to be empty
- Probability for queuing
- Probability for *n* workpieces in the system

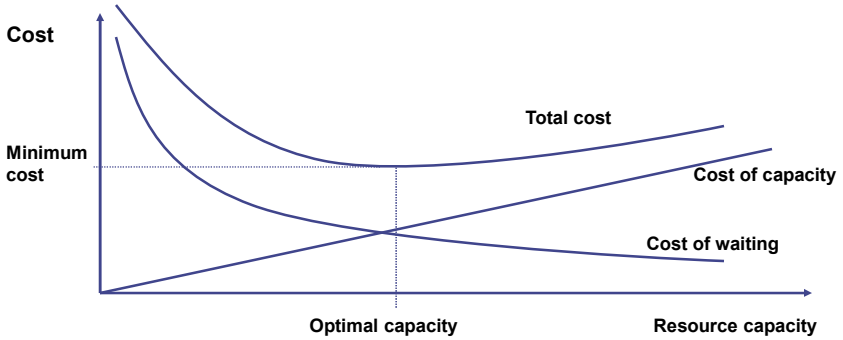
Results are statistical quantities for a stable system. Queuing models can not be used for testing complicated control rules or monitor a single object in the system. Queuing network modelling can be used to model real systems, but its main value in manufacturing is in teaching, because models are simple and they are analytically derived and proven.

esko.niemi@aalto.fi

2

Queuing theory

Final objective is often to reach balance or minimum of service cost and cost of waiting



esko.niemi@aalto.fi

3

Queuing theory

Queuing process is often described with the following notation:

Interarrival time distribution / Service time distribution / Number of parallel servers / Maximum length of queue / Maximum number of customers

where

M = exponential distribution
D = Deterministic (constant) times
G = General distribution (standard deviation given)
E_k = Erlang (Gamma) distribution

esko.niemi@aalto.fi

4

Queuing theory

For example, *M/M/s* denotes a process with exponential interarrival times and service times and that there are *s* parallel servers

Default assumptions:

- Queue length or number of customers are not limited
- Interarrival times and service times are independent of other such events and of each other
- FCFS (First Come First Served) queuing discipline is followed
- The system is stable, i.e. results are static quantities
- Utilization rate of the system is < 100 %

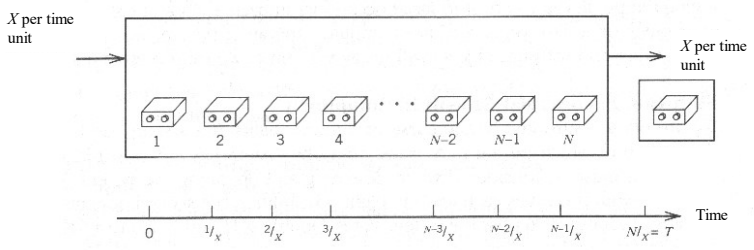
esko.niemi@aalto.fi

5

Little's law[1]

For any production system, machine, workshop etc. in the long run:

*Work in process (N) = Production rate (X) * Throughput time (T)*
$L = \lambda W$



Throughput time (T) = *N* steps * 1/*X* time units per step. That is $N = X\,T$

In other words: "On average *T* days production should be found in the factory"

[1] Little, J.D.C. (1961) "A Proof for the Queuing Formula: $L = \lambda W$", Operations Research, 9(3).

esko.niemi@aalto.fi

6

Little's law

The unit used to measure work in progress (WIP) can be piece, kg, euro etc., as long as the same unit is used. Production rate is correspondingly the same unit per time unit, which must be the same as used for throughput time (min, h, day, year...).

The condition of stability excludes a situation, in which production has started, but no complete products have been finished. Throughput time remains undetermined in such a situation. In the long run the system saturates, and the averages are valid.

Example 1: A machine is processing 12 products per hour and there are on average 20 workpieces waiting on the floor on front of the machine. Average waiting time is = 20 pcs / 12 pcs/h = 1.67 h = 100 min.

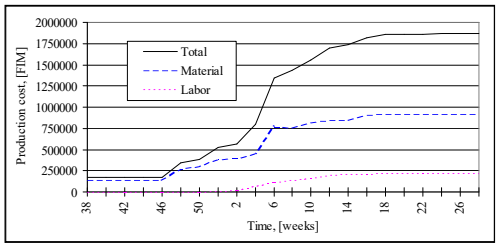
Example 2: A factory produces 2000 products a year and the manufacturing takes about 5 weeks. There should be an average of 2000 pcs/year * 5/52 year = 192 incomplete or complete poducts in the factory.

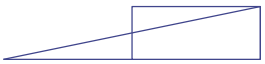
esko.niemi@aalto.fi

7

Little's law

Example 3: The total production cost of an electrical motor factory is 100 MEUR a year, and the value of WIP is 35 MEUR. Costs accumulate over time during production as the figure below shows.. What is the average throughput time?





If all costs would be generated immediately when the production starts, throughput time would be 35/100 = 0.35 years = 18 weeks. If we approximate that the cost accumulation is linear, we can conclude that the throughput time is about 36 weeks in reality.

esko.niemi@aalto.fi

8

Practical implications of Little's law

According to the rule, WIP is in direct relation to throughput time. Therefore:

- If throughput time can be reduced by developing the production process, WIP decreases.
- If WIP is reduced by decreasing order intake, it will reduce as well as output. Actually according to queuing theory, the throughput time decreases, but this is due to lower utilization rate and reduced waiting, but not Little's law.
- Correspondingly, when production rate increases due to improved order intake, WIP increases and cash flow may turn negative for a while. When production approaches full capacity, throughput time (and WIP) may increase sharply due to increased waiting in the process. Under no circumstances can throughput time be decreased by allowing more production on the floor.

esko.niemi@aalto.fi

9

Queuing theory

For queuing systems:

$L = \lambda W$ (WIP)

$L_q = \lambda W_q$ (Queue length)

$L_p = \lambda * 1/\mu = \lambda/\mu$ (In process, machine, being served)

$W = W_q + 1/\mu$ (Throughput time)

$L = L_q + L_p = L_q + \lambda/\mu$ (WIP),

where

λ = Arrival rate (note: $1/\lambda$ is interarrival time)

μ = Processing rate (products per time unit, $1/\mu$ is processing time)

L = Products in system (= WIP, Work in Progress)

W = Throughput time

L_q = Queue length

W_q = Waiting time in queue

esko.niemi@aalto.fi

10

Queuing theory

If queue length or waiting time is known, all other interesting values can be calculated. For given utilization rate, queue length L_q depends mainly on interarrival time and processing time distributions.

For M/M/1

$L_q = \lambda^2 / \mu(\mu - \lambda) = \rho^2/(1 - \rho)$, $L = \rho^2/(1 - \rho) + \rho = \rho/(1 - \rho)$

where

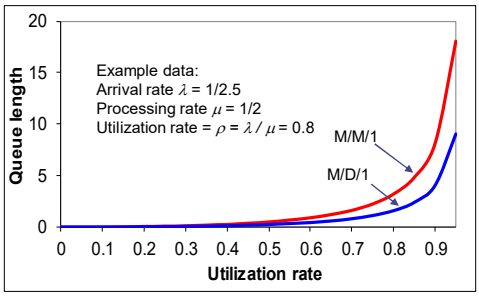
$\rho = \lambda / \mu = L_p$, utilization rate

For example $1/\lambda = 2.5$ ja $1/\mu = 2$:

$L_q = 0.8^2/(1-0.8) = 3.2$

$L = L_q + \lambda/\mu = 3.2 + 0.4/0.5 = 4$

$W = L / \lambda = 4 / (1 / 2.5) = 10$



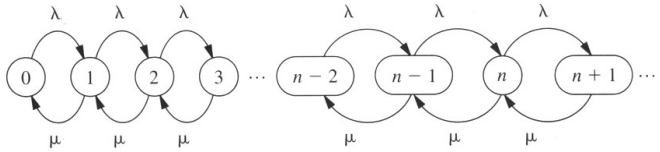
esko.niemi@aalto.fi

11

M/M/1 – queue length

For a Poisson process (like the one described with the negative exponential distribution), the probability of events taking place is the same independent of previous events in the system. State diagram:

State (number of customers, orders in system):



Balance equations (P_n = probability for the system to be in state n):

	In = Out
0	$\mu P_1 = \lambda P_0$
1	$\lambda P_0 + \mu P_2 = (\lambda + \mu) P_1$
2	$\lambda P_1 + \mu P_3 = (\lambda + \mu) P_2$
⋮	⋮
$n - 1$	$\lambda P_{n-2} + \mu P_n = (\lambda + \mu) P_{n-1}$

As sum of P_n must be 1, a numerical solution for this system is possible

esko.niemi@aalto.fi

12

Closed form solution

P_n can be solved for P_0:

State:

$$0: \quad P_1 = \frac{\lambda}{\mu} P_0$$

$$1: \quad P_2 = \frac{\lambda}{\mu} P_1 + \frac{1}{\mu}(\mu P_1 - \lambda P_0) = \frac{\lambda}{\mu} P_1 = \frac{\lambda^2}{\mu^2} P_0$$

$$2: \quad P_3 = \frac{\lambda}{\mu} P_2 + \frac{1}{\mu}(\mu P_2 - \lambda P_1) = \frac{\lambda}{\mu} P_2 = \frac{\lambda^3}{\mu^3} P_0$$

$$\vdots \qquad \vdots$$

$$n-1: \quad P_n = \left(\frac{\lambda}{\mu}\right)^n P_0 = \rho^n P_0, \qquad n = 0, 1, 2, \ldots,$$

In addition, we know that the sum of sate probabilities must be 1:

$$\sum_{n=0}^{\infty} P_n = 1$$

Now P_0 can be solved:

$$P_0 = \left(\sum_{n=0}^{\infty} \rho^n\right)^{-1} = \left(\frac{1}{1-\rho}\right)^{-1} = 1 - \rho.$$

For geometric series: $\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}, \quad \text{if } |x| < 1.$

And further $P_n = (1 - \rho)\rho^n, \qquad n = 0, 1, 2, \ldots.$

esko.niemi@aalto.fi

13

Now L can be solved for ρ, because there must be customers in the system on average:

$$L = \sum_{n=0}^{\infty} nP_n.$$

$$= \sum_{n=0}^{\infty} n(1-\rho)\rho^n$$

$$= (1-\rho)\rho \sum_{n=0}^{\infty} \frac{d}{d\rho}(\rho^n)$$

$$= (1-\rho)\rho \frac{d}{d\rho}\left(\sum_{n=0}^{\infty} \rho^n\right)$$

$$= (1-\rho)\rho \frac{d}{d\rho}\left(\frac{1}{1-\rho}\right)$$

$$= \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda}.$$

And finally, the queue length:

$L_q = L - \rho = \rho/(1 - \rho) - \rho = (\rho - \rho + \rho^2)/(1 - \rho) = \rho^2/(1 - \rho).$

esko.niemi@aalto.fi

14

Queuing models

For manufacturing systems, some readily available queuing models are:

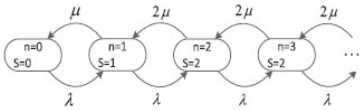
- *M/M/s*
- *M/M/s/-/N*
- *(M/M/s/K)*
- *M/M/s* and state dependent μ
- *M/G/*1
- *M/D/s*
- *M/E$_k$/s*
- Models for other than *M*-arrivals
- Models for other than FCFS priorities

esko.niemi@aalto.fi

15

M/M/2 system

State diagram:



Here:
λ = arrival rate for the whole system
μ = processing rate for one server (machine)
n* = *state
***N* = number of servers**
S = units currently processed

Numerical solution:

		λ	μ	k
		0.4	0.25	2
n	p_n	$p_n\lambda$		
0	0.111845	0.044738	0.044738	$p_{n+1}\mu$
1	0.178952	0.071581	0.071581	$p_{n+1}2\mu$
2	0.143161	0.057264	0.057264	$p_{n+1}2\mu$
3	0.114529	0.045812	0.045812	$p_{n+1}2\mu$
4	0.091623	0.036649	0.036649	$p_{n+1}2\mu$

Balance equations can also be written for flows between nodes; "cut partition method"

$$\lambda p_0 = \mu p_1$$

$$\lambda p_1 = 2\mu p_2$$

$$\vdots$$

$$\lambda p_{N-1} = N\mu p_N$$

$$\lambda p_N = N\mu p_{N+1}$$

$$\lambda p_{N+1} = N\mu p_{N+2}$$

$$\vdots$$

$$\sum_{n=0}^{\infty} p_n = 1.$$

esko.niemi@aalto.fi

16

M/M/s formulas

Here:
λ = arrival rate for the whole system
μ = processing rate for one server (machine)
k = number of servers

Probability for empty system:

$$P_0 = \frac{1}{\sum_{n=0}^{k-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^k}{k!}\left(\frac{k\mu}{k\mu-\lambda}\right)}$$

Queue length:

$$L_q = \frac{(\lambda/\mu)^k \lambda\mu}{(k-1)!(k\mu-\lambda)^2} P_0$$

Waiting time in queue:

$$W_q = \frac{L_q}{\lambda}$$

Probability of n customers (parts, orders) in system:

$$P_n = \frac{(\lambda/\mu)^n}{n!} P_0 \quad \text{for } n \le k$$

$$P_n = \frac{(\lambda/\mu)^n}{k!k^{(n-k)}} P_0 \quad \text{for } n > k$$

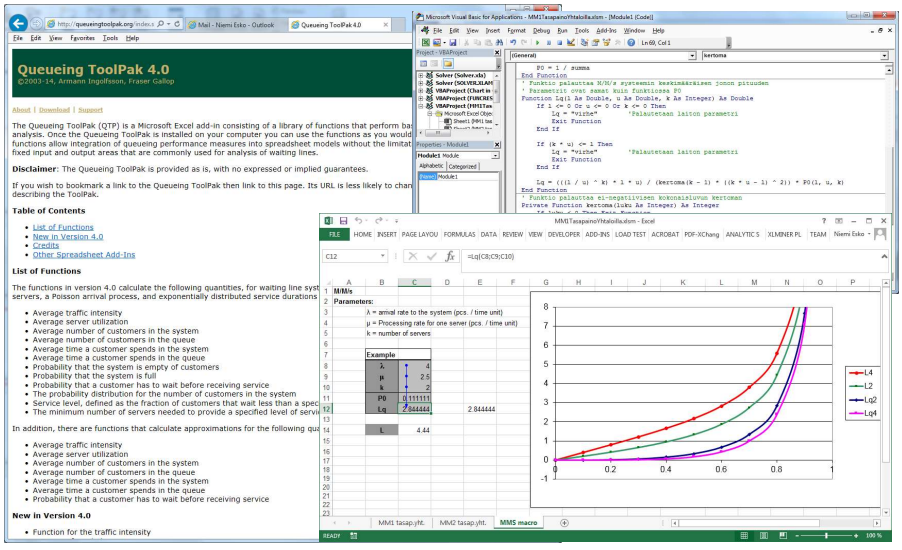
Probability for waiting:

$$P_w = \frac{1}{k!}\left(\frac{\lambda}{\mu}\right)^k \left(\frac{k\mu}{k\mu-\lambda}\right) P_0$$

esko.niemi@aalto.fi

17

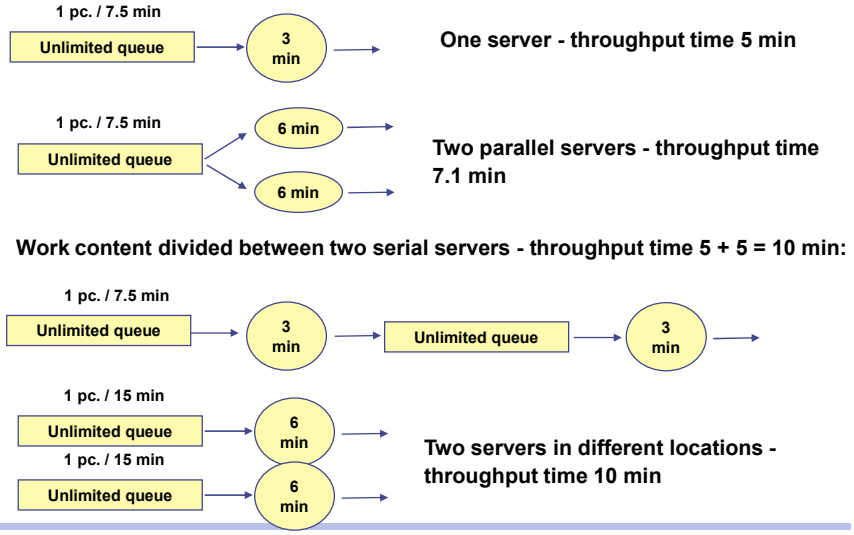
M/M/2 and M/M/4 with Excel functions



esko.niemi@aalto.fi

19

M/M/s server systems with equal capacity



esko.niemi@aalto.fi

20

Cost model for queuing

Final objective is to reach balance or minimum of service cost and cost of waiting

Minimize E(TC) = E(service cost) + E(waiting cost)

Cost of service is typically machine hour, labor hour etc. and easy to determine. Waiting time can be estimated using a suitable formula. Cost of waiting time may be easy to calculated, if the customer is a machine or person the cost of which is the opportunity cost of lost sales. If the customer is external to the service organisation, estimation of costs may be difficult. For example, badwill caused by late deliveries may result in lost sales, but you may not know that you were blacklisted as a supplier and why this happened.

esko.niemi@aalto.fi

21

Cost of waiting – example 1

Tool storage:

Workers fetch tools from a central tool storage 30 times an hour on average. A person in the tool storage can serve 20 customers per hour. Cost of lost production due to workers waiting for service is 48 48 euro/h. How many service persons should be employed, if one costs 20 euro/h?

We apply *M/M/s* model. Workers waiting or being served:

1 storage person: not sufficient; 2 storage persons: 3.43 kpl; 3 storage persons: 1.74 kpl; 4 storage persons: 1.54 kpl; 5 storage persons: 1.51 kpl

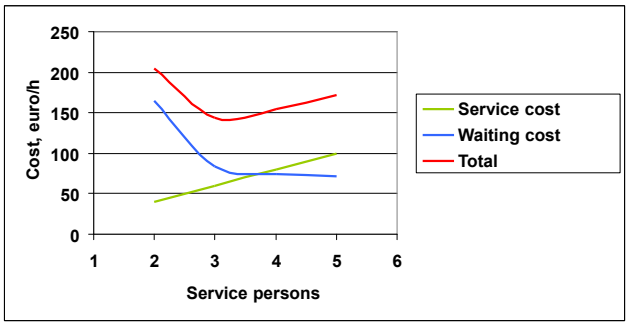
esko.niemi@aalto.fi

22

Cost of waiting – example 1

Costs:

2 storage persons: TC = 20 x 2 + 3.4 x 48 = 205 euro/h
3 storage persons: TC = 20 x 3 + 1.74 x 48 = 144 euro/h
4 storage persons: TC = 20 x 4 + 1.54 x 48 = 154 euro/h
5 storage persons: TC = 20 x 5 + 1.51 x 48 = 172 euro/h



esko.niemi@aalto.fi

23

M/M/1/-/N model formulas

Here:
λ = arrival rate for the whole system
μ = processing rate for one server (machine)
***N* = number of customers**

Probability for empty system:

$$P_0 = \frac{1}{\sum_{n=0}^{N} \frac{N!}{(N-n)!}\left(\frac{\lambda}{\mu}\right)^n}$$

Queue length:

$$L_q = N - \frac{\lambda + \mu}{\lambda}(1 - P_0)$$

WIP:

$$L = L_q + (1 - P_0)$$

Waiting time in queue:

$$W_q = \frac{L_q}{(N - L)\lambda}$$

Probability of *n* customers (parts, orders) in system:

$$P_n = \frac{N!}{(N-n)!}\left(\frac{\lambda}{\mu}\right)^n P_0 \qquad n = 0, 1, \ldots, N$$

esko.niemi@aalto.fi

24

Cost of waiting – example 2

Machine tools break down:

Our machine shop has 10 machine tools, which break down every 10 hours on average. Cost of machine not in production is 48 euro/h. Repairing a machine takes one hour on average for one repair person. We allocate all repair persons to machines according to FIFO priority. How many maintenance persons do we employ, if one costs 20 euro/h?

We assume that Time Between Failures and Time To Repair are exponentially distributed. We apply *M/M/1/-/10* model. We have machines down as follows:

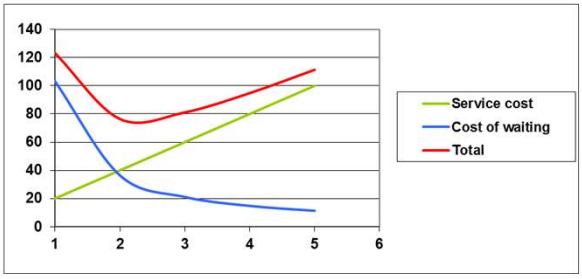
1 repair person: 2.15 machines; 2 repair persons: 0.759 machines; 3 repair persons: 0.44 machines; 4 repair persons: 0.308 machines; 5 repair persons: 0.236 machines

esko.niemi@aalto.fi

25

Cost of waiting – example 2

Costs:
1 repair person: TC = 20 x 1 + 2.15 x 48 = 123.2 euro/h
2 repair persons: TC = 20 x 2 + 0.759 x 48 = 76.4 euro/h
3 repair persons: TC = 20 x 3 + 0.440 x 48 = 81.1 euro/h
4 repair persons: TC = 20 x 4 + 0.308 x 48 = 94.8 euro/h
5 repair persons: TC = 20 x 5 + 0.236 x 48 = 111.3 euro/h



esko.niemi@aalto.fi

26

Waiting and batch sizes – an example

A turning facility uses an automatic bar lathe to manufacture 48 000 hydraulic connectors a year, 40 different types, 3 minutes cutting time / pc. and the set-up time is 100 min. The production is made to order so, that an optimal batch is manufactured each time the stock level is lower than the ordered amount. The demand is uniformly distributed between the different connector types. Material cost is 1 euro/connector and machining cost is 0.5 euro/min in two shifts. Set-up cost is 1 euro/min because it requires the operator in addition to the machine.

We assume that *M*/*M*/1 can be applied and examine the effect of batch size on throughput time and cost.

esko.niemi@aalto.fi

27

Automatic bar turning machines

esko.niemi@aalto.fi

28

Waiting and batch sizes – an example

BatchesAndWaiting.xls [Compatibility Mode] - Excel

FILE | HOME | INSERT | PAGE LAY | FORMULA | DATA | REVIEW | VIEW | DEVELOPE | ADD-INS | LOAD TES | ACROBAT | PDF-XCha | ANALYTIC | XLMINER | TEAM | Niemi Esko

C9 | =B9

	A	B	C	D	E	F	G	H	I	J	K
1	**Throughput time study**										
2	Annual working hours, 2-shift	198000	198000	198000	198000	198000	198000	198000	198000	198000	1980
3	Batch size	100	150	200	250	300	350	400	450	500	5
4	Set-up time	100	100	100	100	100	100	100	100	100	1
5	Processing time	3	3	3	3	3	3	3	3	3	
6	Processing cost	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2
7	Batch time	400	550	700	850	1000	1150	1300	1450	1600	17
8	Batch time, days	0.4	0.6	0.7	0.9	1.0	1.2	1.4	1.5	1.7	
9	Volume/year	48000	48000	48000	48000	48000	48000	48000	48000	48000	480
10	Batch interarrival time	413	619	825	1031	1238	1444	1650	1856	2063	22
11	Utilization rate	0.970	0.889	0.848	0.824	0.808	0.797	0.788	0.781	0.776	0.7
12	Lq	31.03	7.11	4.75	3.87	3.40	3.12	2.93	2.79	2.68	2
13	Lamda	0.0024	0.0016	0.0012	0.0010	0.0008	0.0007	0.0006	0.0005	0.0005	0.00
14	Wq	12800	4400	3920	3986	4211	4502	4829	5175	5535	59
15	Waiting time in queue	13.3	4.6	4.1	4.2	4.4	4.7	5.0	5.4	5.8	6
16	W	13200	4950	4620	4836	5211	5652	6129	6625	7135	76
17	Throughput time, days	13.8	5.2	4.8	5.0	5.4	5.9	6.4	6.9	7.4	8
18											
19	Set-up cost	100	100	100	100	100	100	100	100	100	1
20	Annual volume/type	1200	1200	1200	1200	1200	1200	1200	1200	1200	12
21	Holding cost-%	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.
22	Set-up cost/pc.	1.00	0.67	0.50	0.40	0.33	0.29	0.25	0.22	0.20	0.
23	Holding cost	0.04	0.05	0.06	0.08	0.09	0.10	0.11	0.13	0.14	0.
24	Cost/piece	3.54	3.22	3.06	2.98	2.92	2.89	2.86	2.85	2.84	2.
25											
26											

Throughput time

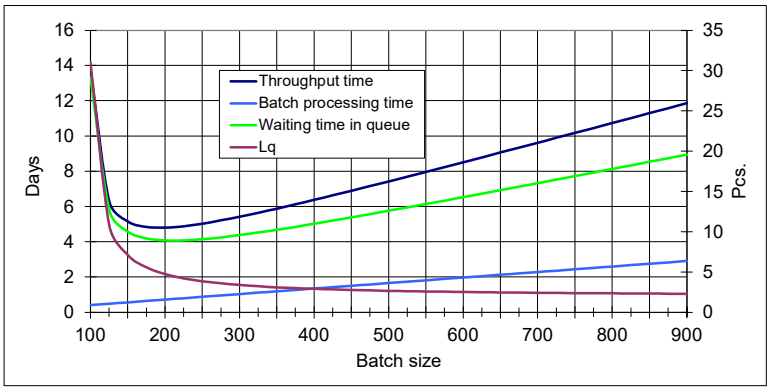
READY | CALCULATE | 100 %

esko.niemi@aalto.fi

29

Waiting and batch sizes – throughput times

When batch size is increased, utilization rate decreases, because the number of set-ups reduces. This shortens queues when measured in number of batches, but the larger batches increase waiting time.



esko.niemi@aalto.fi

30

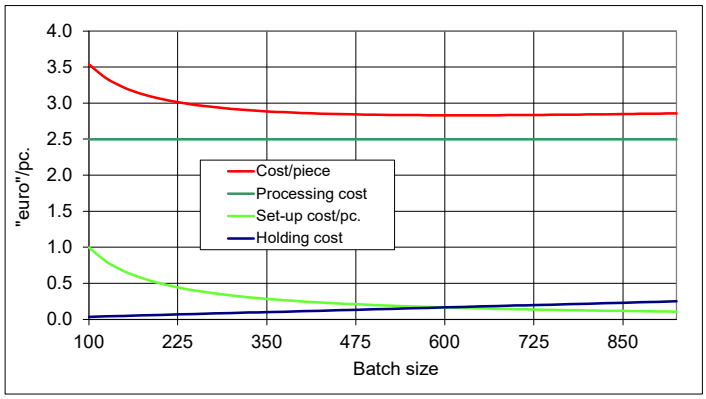
Waiting and batch sizes – cost

Economic Order Quantity can be calculated using the classical formula $Q_{eoq} = \sqrt{2DS/H}$,

which gives the minimum total cost batch size: $C_{tot} = DC + (D/Q)S + (Q/2)H$

Here D = Demand, C = Cost/piece, Q = Batch size, S = Cost/batch (set-up), H = Holding cost

This is mainly intended for production or purchasing to stock, and it does not consider throughput time or limited capacity

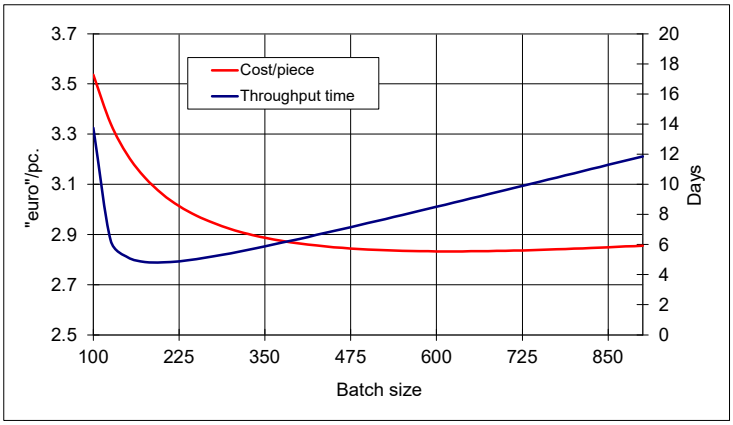


esko.niemi@aalto.fi

31

Waiting and batch sizes

Typically, *EOQ* formula gives such a large ”optimal” batch size, that the corresponding throughput times are too long for MTO production

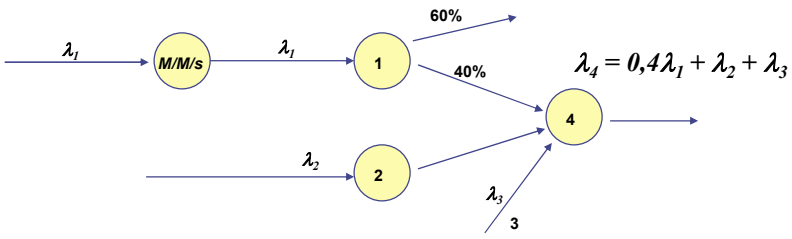


esko.niemi@aalto.fi

32

Queuing networks

If the arrivals are a Poisson process with parameter λ and the μ in for *M/M/s* servers is equal, exit process is also a Poisson process with parameter λ. In these conditions queuing networks can be analysed as a collection of single queuing systems with unlimited queue lengths. The arrival rate from other nodes and from outside has to be determined, for example:

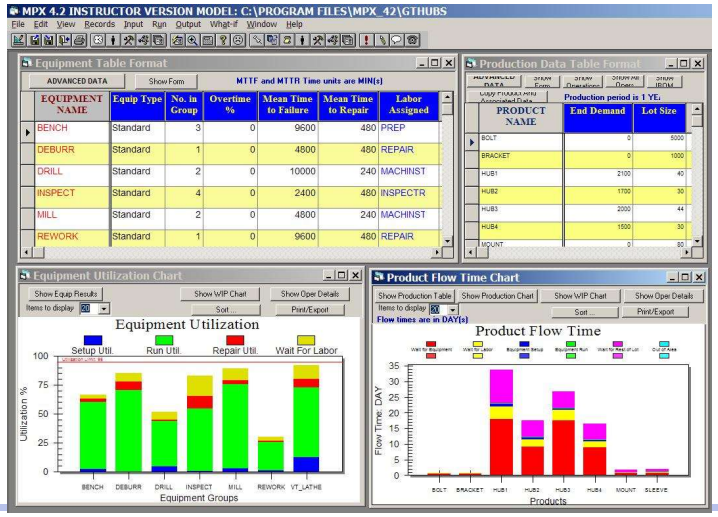


esko.niemi@aalto.fi

33

Queuing networks

A screenshot of MPX queuing network modeling software:



esko.niemi@aalto.fi

34

Summary - throughput time

- In a stochastic process throughput time depends on
  - Utilization rate
  - Variation
  - Processing time (and in case of batch production also of the batch size and set-up time)
- G/G/1 approximation[1]

  Cycle Time = *VUT* + *T*,

  where

  $V = (CV^2_{processing\ time} + CV^2_{interarrival\ time})/2$

  $U = u / (1 - u)$, where u = utilization rate

  T = processing time.
- Because high utilization rate increases *U* strongly, it is beneficial to focus improvement efforts on bottle necks

[1] also known as *Kingman's formula*

esko.niemi@aalto.fi

35

Production control and queuing models

Queuing models provide estimates of averages, variation and probabilities based on assumptions of system and input characteristics. They can be used for rough analyses of real production systems. Practical production is controlled by rough cut capacity and workload matching and prioritizing of orders and tasks in the short run. Under these circumstances the assumptions about applicability of typical probability distributions may be quite valid, especially for arrival processes. For service processes, the main sources of variability are product variation and batch size variation. In manual work, also differences between workers and random human behavior increase variability.

Real (optimized) production control usually requires formulating and solving of binary mathematical programming problems. This can be very hard…

esko.niemi@aalto.fi

Short Introduction to Optimization | 17.1.2022

Esko Niemi | Helsinki

EIT Manufacturing is supported by the EIT,
a body of the European Union

Outline

TF KnowNet

General introduction

- Introduction & Production allocation example
- Solving linear problems, sensitivity
- Non-linear problems, convexity
- Solving integer problems
- Modeling using binary variables & Production control example
- Heuristics

eit Manufacturing

Aalto University
School of Engineering

Optimization – introduction

- In optimization one models the system to be optimized and the best possible solution to a problem related to it is described using mathematical formulas. This may mean for example:
  - Cost minimization
  - Profit maximization
  - Throughput time minimization
  - Utilization maximization etc.
- Basic model:

  Min $f(x)$

  so that,

  $h(x) = 0$

  $g(x) =< 0$

  $x >= 0$

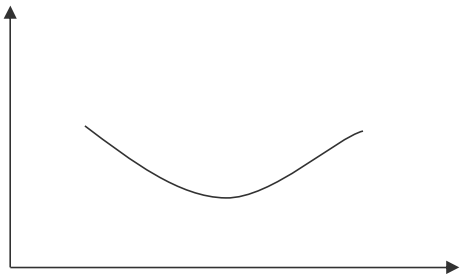
  $x \in R^n$
- In the model ”decision variables” (x) represent those inputs, the values of which are determined by solving the problem.

Optimization – introduction

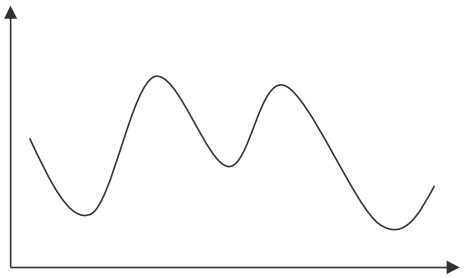
- The value (goodness) of the solution is a function of the decision variables. This function is called *cost function* or *objective function*: *f(x)*
- The variable values can be constrained by setting *constraints*, which are also functions of the decision variables: *h(x)* = 0, *g(x)* =< 0, or *bounds*: *x* >= 0
- The constants appearing in the objective function or the constraints are called *parameters* or *data*
- The model can for example determine, that the objective is to choose such values for the decision variables, that the value of the objective function is minimized and the constraints are not violated
- If such a model is formulated as a mathematical problem and especially if it is solved using deterministic methods, that finally always find the optimal solution, the term *mathematical programming* is often used

Optimization – introduction

- Mathematical optimization problems are solved either using a general purpose solver engine or special purpose software
- The difficulty of the problem and the solution method depend on the problem type in relation to linearity and integer constraints on variables
- For example:
  - Linear Programming, LP
  - Integer Linear Programming, ILP
  - Non-Linear Programming, NLP
- In addition convexity or lack of it is relevant:



Convex function with one minimum



Non-convex function with several local minima

Optimization – introduction

- Because problems are often very complex, optimal solution may not be worth trying for. In these cases heuristic methods may very quickly result in solutions that are good from the practical point of view, although not optimal.
- Actually proving optimality of the solution is typically very difficult.
- Determining parameter values can be difficult in practice. It may require collection and interpretation of hard to get and ambiguous data. Therefore one often resorts to estimates. Sensitivity analysis is done to find out how a change in values changes the optimal solution.
- The model must be tested and validated. This usually leads to improvement of the model.
- Finally the model is implemented and used.

Linear programming - LP

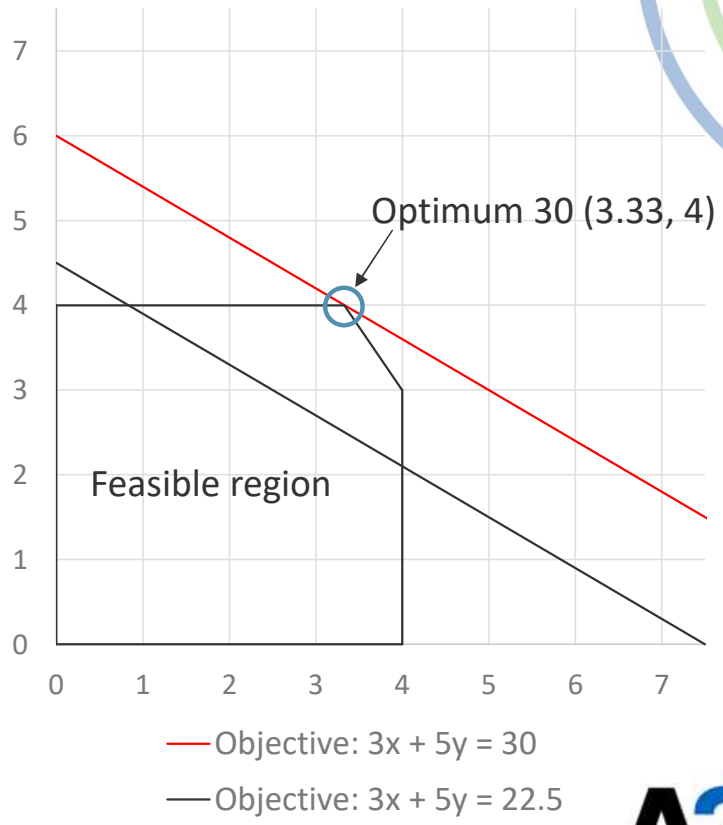
The following simple two-dimensional (two variable) problem is linear, convex, and the decision variables may take real values. To solve it, one must find such values for decision variables *x* and *y* that the value of objective function (*z*) is maximized. In addition, the solution (*x, y*) must satisfy a set of constraints:

Maximize $z = 3x + 5y$

so that,

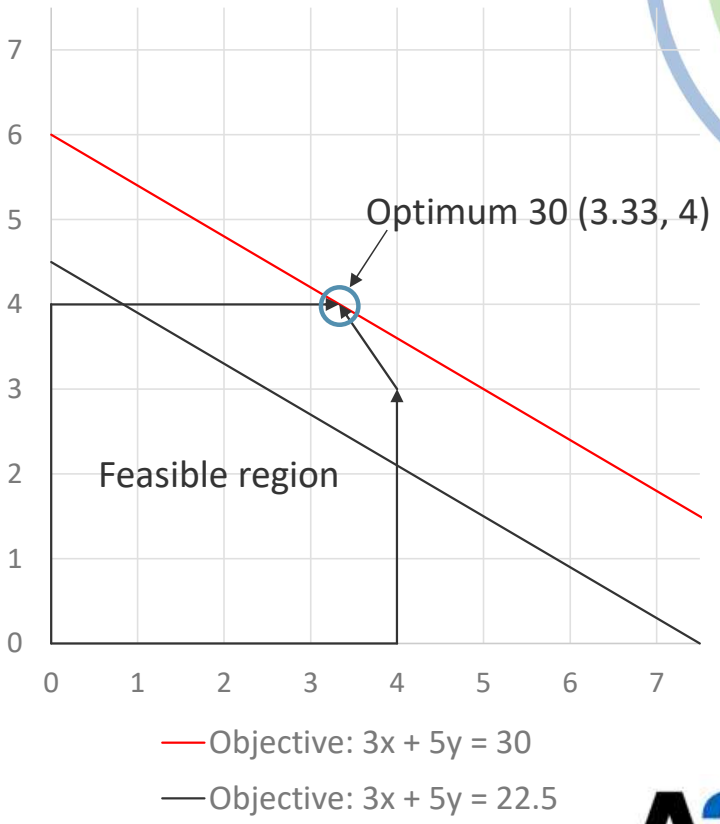
$x <= 4$
$y <= 4$
$3x + 2y <= 18$
$x >= 0, y >= 0$

The solution must be a point within the feasible region limited by the constraints.



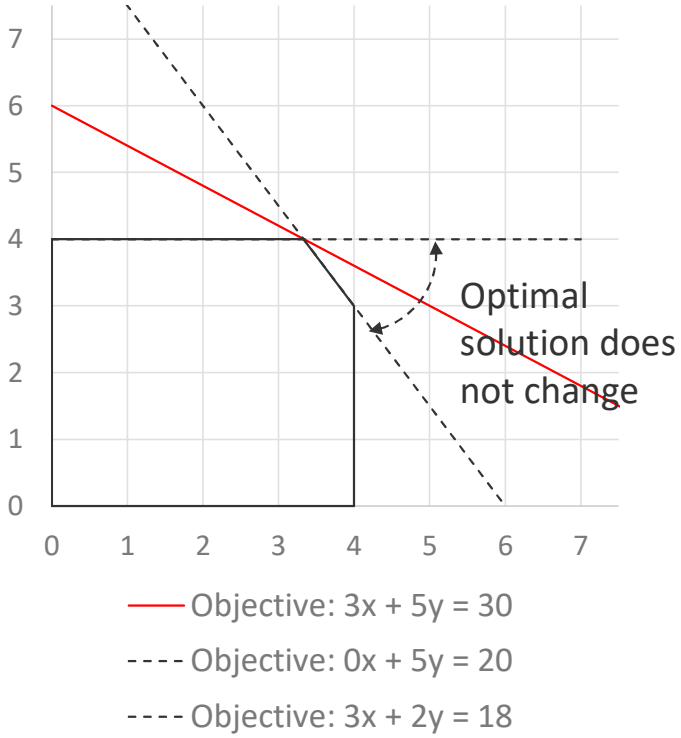
Solving of linear problems

- Here the optimal solution (x = 3.33, y = 4) is shown on the red objective line $3x + 5y = 30$
- Optimal solutions to LP problems are always in the corners of the feasible region
- In the Simplex method one moves along the constraining lines to the direction in which the solution improves
- When the result does not improve anymore, the optimal solution has been found
- In the interior-point approach starting point is within the feasible region and one moves iteratively in the result-improving direction. The constraints are included in the objective function so that hitting the constraint line is penalized with a barrier function.

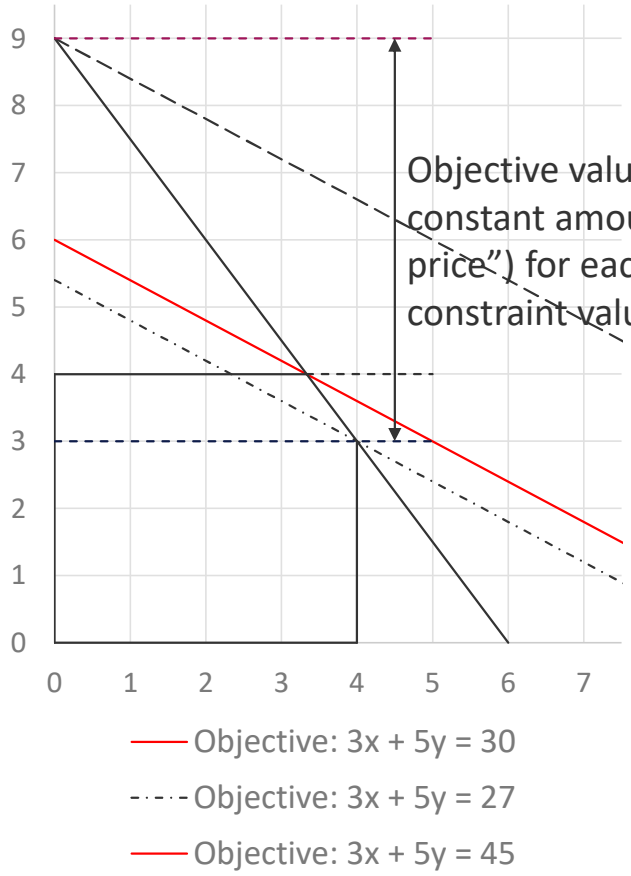


Solution sensitivity to parameter values

Sensitivity of the optimal solution (3.33, 4) to objective function coefficients (slope)



Sensitivity of objective value to change in RHS of constraint y <= 4



Examples of manufacturing problems for optimization

Following examples are typical for manufacturing industries. The problem is to define:

- Allocation of jobs to machines in order to minimize set-ups and throughput time within capacity constraints
- Schedule for a machine system in order to maximize utilization and minimize tardiness
- Tool order in tool magazine in order to minimize waiting for tool change
- Sheet metal part nests in order to minimize waste and storage levels
- Raw material standard storage dimensions in order to minimize waste
- Cutting parameter values to minimize costs
- Factory and warehouse locations
- A model fitted to data in order to be able to estimate costs of future orders

Optimization example – Excel/Solver model

Allocation of jobs to machines

Six different products are machined using three automatic lathes
Problem: Which machines are used for which products so that the total cost is minimized?

Spreadsheet model:

Processing times (c_{ij})

	Products	Machines: Biglia	Miyano	Okuma	Amounts, d_j
	Sleeve 1	2.5	3.5	3.3	19000
	Sleeve 2	2.2	2.6	3.8	24000
	Nipple 1	2.1	3.2	3.0	32000
	Nipple 2	2.3	3.2	3.8	8000
	Connector 1	2.1	3.1	3.5	41000
	Connector 2	2.4	2.6	3.8	22000
	Capacity, s_i	134400	134400	134400	

Parameters

Produced amounts (x_{ij})

		Biglia	Miyano	Okuma	
	Sleeve 1	0.0	0.0	19000.0	19000
	Sleeve 2	0.0	24000.0	0.0	24000
	Nipple 1	20520.5	0.0	11479.5	32000
	Nipple 2	3216.4	4783.6	0.0	8000
	Connector 1	41000.0	0.0	0.0	41000
	Connector 2	0.0	22000.0	0.0	22000
		i			

Solution (=decision varia

Total times

j		Biglia	Miyano	Okuma	
	Sleeve 1	0	0	62639	
	Sleeve 2	0	61247	0	
	Nipple 1	42969	0	34475	
	Nipple 2	7326	15207	0	*Objective function value*
	Connector 1	84104	0	0	
	Connector 2	0	57946	0	*Z*
		134400	134400	97114	365914

← Spreadsheet model

Mathematical optimization model:

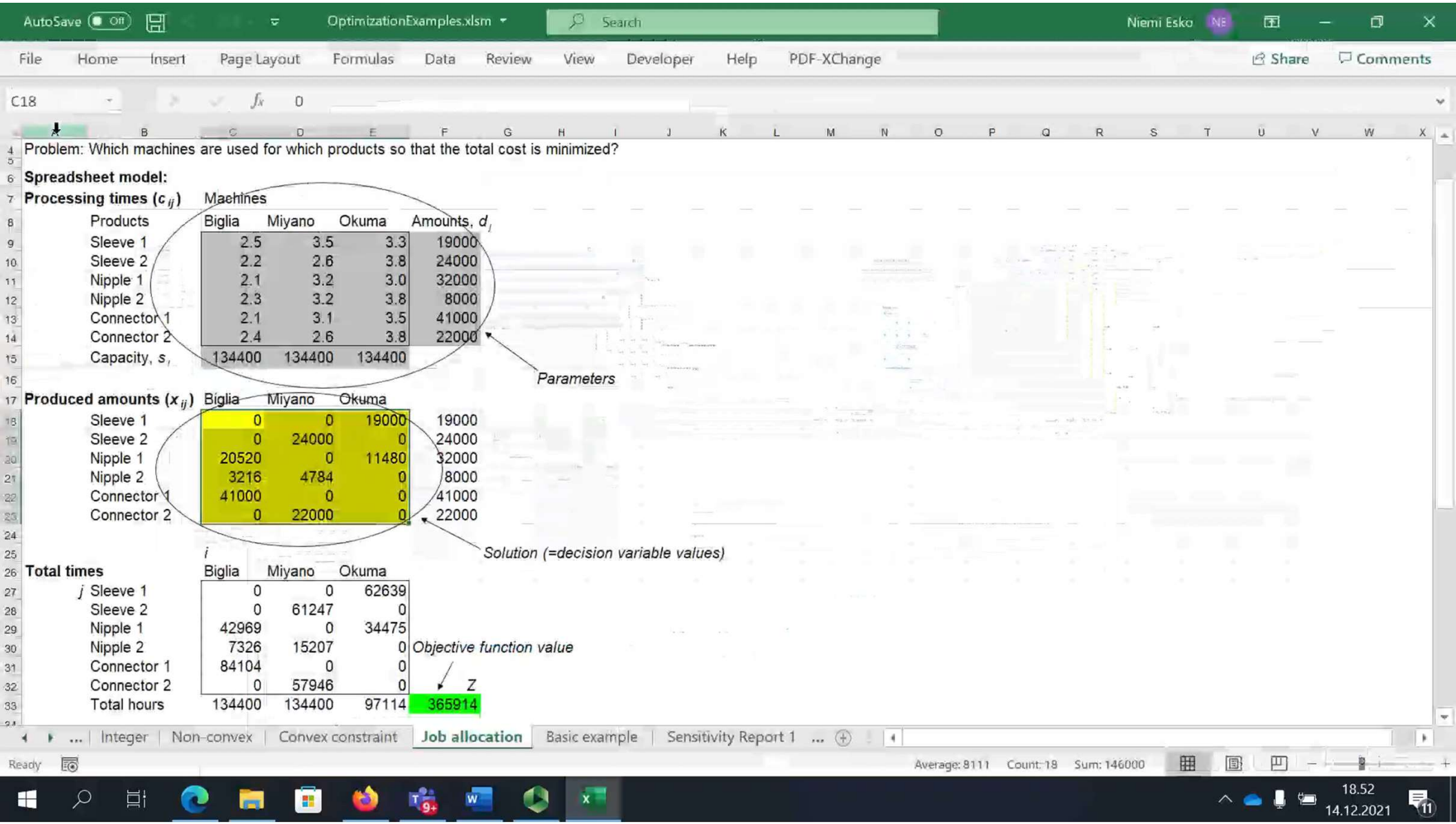
$$\text{Min} \quad Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij},$$

so that

$$\sum_{j=1}^{n} c_{ij} x_{ij} \le s_i \quad \text{for } i = 1,..,m$$

$$\sum_{i=1}^{m} x_{ij} = d_j \quad \text{for } j = 1,..,n$$

$$x_{ij} \ge 0 \quad \text{for all } i \text{ and } j.$$


AutoSave Off
OptimizationExamples.xlsm
Search
Niemi Esko
File Home Insert Page Layout Formulas Data Review View Developer Help PDF-XChange
Share Comments
C18
Problem: Which machines are used for which products so that the total cost is minimized?
Spreadsheet model:
Processing times (c_{ij}) Machines
Products Biglia Miyano Okuma Amounts, d_j
Sleeve 1 2.5 3.5 3.3 19000
Sleeve 2 2.2 2.6 3.8 24000
Nipple 1 2.1 3.2 3.0 32000
Nipple 2 2.3 3.2 3.8 8000
Connector 1 2.1 3.1 3.5 41000
Connector 2 2.4 2.6 3.8 22000
Capacity, s_i 134400 134400 134400
Parameters
Produced amounts (x_{ij}) Biglia Miyano Okuma
Sleeve 1 0 0 19000 19000
Sleeve 2 0 24000 0 24000
Nipple 1 20520 0 11480 32000
Nipple 2 3216 4784 0 8000
Connector 1 41000 0 0 41000
Connector 2 0 22000 0 22000
Solution (=decision variable values)
i
Total times Biglia Miyano Okuma
j Sleeve 1 0 0 62639
Sleeve 2 0 61247 0
Nipple 1 42969 0 34475
Nipple 2 7326 15207 0
Connector 1 84104 0 0
Connector 2 0 57946 0
Total hours 134400 134400 97114
Objective function value
Z
365914
Integer Non-convex Convex constraint Job allocation Basic example Sensitivity Report 1
Ready
Average: 8111 Count: 18 Sum: 146000
18.52
14.12.2021

Optimization example – CPLEX model

```
min
2.5x11 + 3.5x21 + 3.3x31 + 2.2x12 + 2.6x22 + 3.8x32
+ 2.1x13 + 3.2x23 + 3x33 + 2.3x14 + 3.2x24 + 3.8x34
+ 2.1x15 + 3.1x25 + 3.5x35 + 2.4x16 + 2.6x26 + 3.8x36
st
2.5x11 + 2.2x12 + 2.1x13 + 2.3x14 + 2.1x15 + 2.4x16 <= 134400
3.5x21 + 2.6x22 + 3.2x23 + 3.2x24 + 3.1x25 + 2.6x26 <= 134400
3.3x31 + 3.8x32 + 3x33 + 3.8x34 + 3.5x35 + 3.8x36 <= 134400
x11 + x21 + x31 = 19000
x12 + x22 + x32 = 24000
x13 + x23 + x33 = 32000
x14 + x24 + x34 = 8000
x15 + x25 + x35 = 41000
x16 + x26 + x36 = 22000
bounds
x11 >= 0, x21 >= 0, x31 >= 0, x12 >= 0, x22 >= 0, x32 >= 0,
x13 >= 0, x23 >= 0, x33 >= 0, x14 >= 0, x24 >= 0, x34 >= 0,
x15 >= 0, x25 >= 0, x35 >= 0, x16 >= 0, x26 >= 0, x36 >= 0
end
```

← CPLEX model

Mathematical optimization model:

$$\text{Min} \quad Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij},$$

so that

$$\sum_{j=1}^{n} c_{ij} x_{ij} \le s_i \quad \text{for } i = 1,..,m$$

$$\sum_{i=1}^{m} x_{ij} = d_j \quad \text{for } j = 1,..,n$$

$$x_{ij} \ge 0 \quad \text{for all } i \text{ and } j.$$

Optimization example – OPL/CPLEX model

```
int m = ...;
int n = ...;
float c[1..m][1..n]= ...;
float s[1..m] = ...;
float d[1..n] = ...;

dvar float+ x[1..m][1..n];

minimize
    sum(i in 1..m)
        sum(j in 1..n)
            c[i][j]*x[i][j];

subject to{
    forall (i in 1..m)
        sum(j in 1..n)
            c[i][j]*x[i][j]<=s[i];
forall (j in 1..n)
        sum(i in 1..m)
            x[i][j]==d[j];
}
```

← CPLEX model

Data separately:

```
m = 3;
n = 6;
c = [[2.5 2.2 2.1 2.3 2.1 2.4]
     [3.5 2.6 3.2 3.2 3.1 2.6]
     [3.3 3.8 3 3.8 3.5 3.8]];
s = [1.344e+5 1.344e+5 1.344e+5];
d = [19000 24000 32000 8000 41000
22000];
```

Mathematical optimization model:

$$\text{Min} \quad Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij},$$

so that

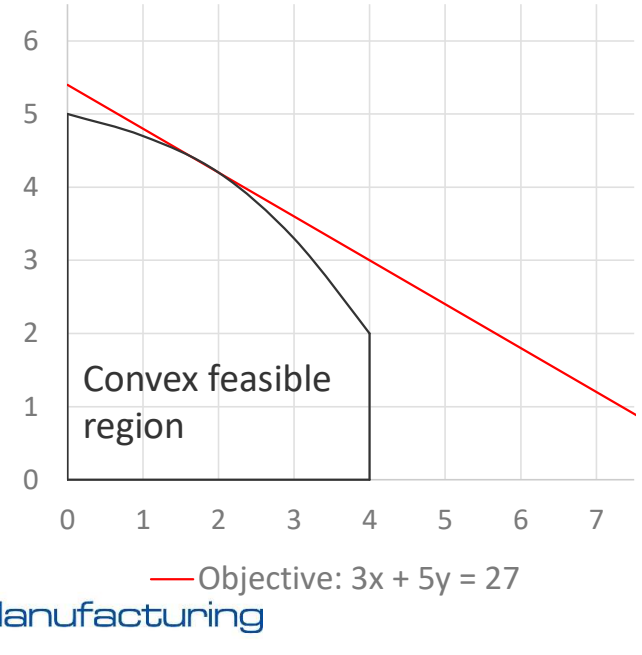
$$\sum_{j=1}^{n} c_{ij} x_{ij} \le s_i \quad \text{for } i = 1,..,m$$

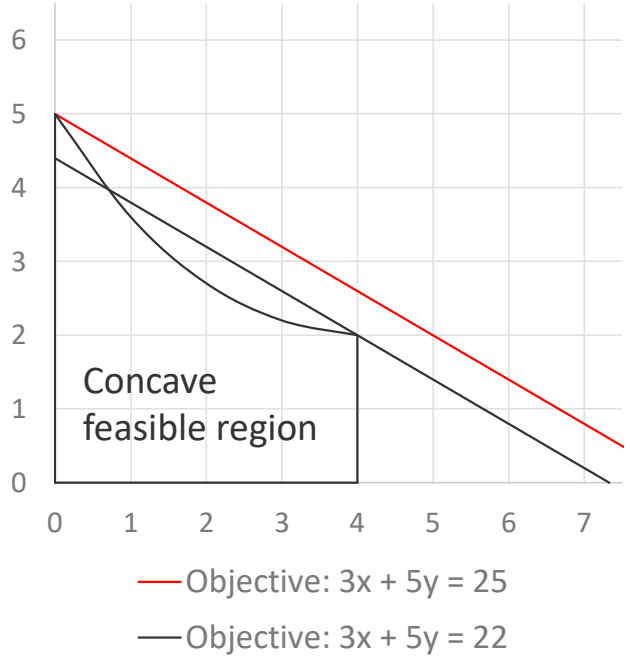
$$\sum_{i=1}^{m} x_{ij} = d_j \quad \text{for } j = 1,..,n$$

$$x_{ij} \ge 0 \quad \text{for all } i \text{ and } j.$$

Nonlinear optimization

If a smooth nonlinear objective function has only one global optimum value and constraints are linear or concave, or there are none, gradient methods can be applied to solving and solving is easy. This type is called convex problem. Solving is especially efficient if problems are quadratic: they contain only second degree polynomials or contain terms that are products of two variables. – If the objective function of a nonlinear problem has several peaks and/or constraints are concave one has to resort to search methods. Finding optimal solution is hard and it may not be guaranteed.





Integer programming – IP

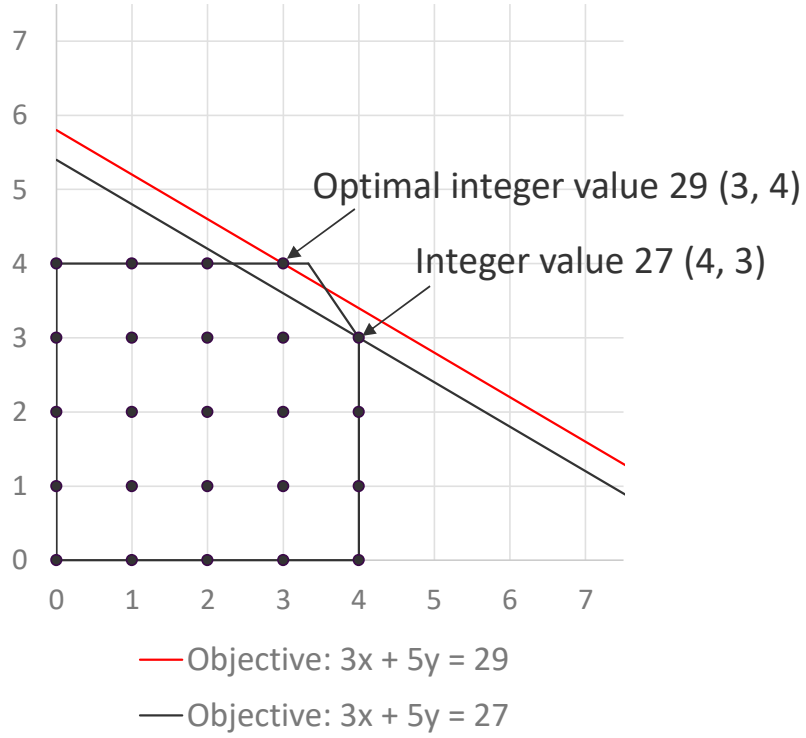
- In integer problems decision variables take integer values
- If some variables take real values, the problem is called a mixed integer programming, MIP, problem
- If variables take only values 1 or 0, the problem is called a binary programming, BIP, problem
- Default assumption is that the problem is otherwise linear
- Integer programming problems are typically much more difficult to solve than LP problems
- This is because whereas LP problem solution is always in a corner point of the feasible region, this is not the case with IP problems
- Although the number of feasible integer solutions is limited, this number may be vast
- Sometimes optimal solution of an LP problem is integer. This solution must be the optimal solution to the integer constrained version of the problem as well
- Sometimes a real valued solution of an IP problem can be rounded to and integer value without any practical harm

Solving of integer programming problems

- Optimal solution methods for IP problems are based on enumeration, which is systematic consideration of all possibilities. However, here as large parts of the variable value space as possible are excluded from consideration by concluding, that they are not relevant.
- With linear problems the exclusion is based on solving the LP relaxation of the problem with certain added constraints. This way a solution, better than which any integer solution cannot be, is found quickly. If this “bound” is worse than the best integer value already found, the area in focus can be excluded from further examination.
- Basic methods are cutting plane and branch and bound methods. In cutting plane methods additional constraints (cutting planes) are used to limit the solution space. In branch and bound methods, the solution space is a tree-type data structure, of which branches are excluded.
- In nonlinear cases other type of deduction for problem size reduction can be tried
- Linearity is thus a very welcome property of an IP problem
- In all cases the number of variables should be kept as low as possible and as tight constraints as possible defined

Integer programming example

- Integer requirement added to the simple example problem
- For the job allocation example, rounding of original non-integer optimal result values would be practical



Binary integer programming – BIP

- Binary variables are commonly used in production planning and scheduling problems to e.g.:
  - Indicate events taking place
  - To indicate choices between alternatives
  - To exclude constraints from the model
  - To activate one time costs or events

- The idea is to keep the model otherwise linear so that efficient enumeration methods can be applied for solving
- In the following formulation of the job allocation problem, a fixed "cost" burdens the machines that are equipped to machine any part type. This cost is activated in the objective function with binary variables.
- The binary variables (y) are forced to take the value 1 if any production (x) takes place with the following "auxiliary" constraint:

  $x <= My$

  Here M is a large constant. If x takes any positive value, y must be 1 for the constraint to hold.

Job allocation example – BIP for fixed cost inclusion

Allocation of jobs to machines

Six different products are machined using three automatic lathes
Problem: Which machines are used for which products so that the total cost is minimized?

Spreadsheet model:

Processing times (c_{ij})

Products	Biglia	Miyano	Okuma	Amounts
Sleeve 1	2.5	3.5	3.3	19000
Sleeve 2	2.2	2.6	3.8	24000
Nipple 1	2.1	3.2	3.0	32000
Nipple 2	2.3	3.2	3.8	8000
Connector 1	2.1	3.1	3.5	41000
Connector 2	2.4	2.6	3.8	22000
Capacity	134400	134400	134400	

Fixed cost for production: 2000

Big M: 99999

Parameters

Produced amounts (x_{ij})

Products	Biglia	Miyano	Okuma							
Sleeve 1	0	0	19000	19000	0	0	1	0	0	99999
Sleeve 2	0	24000	0	24000	0	1	0	0	99999	0
Nipple 1	22109	0	9891	32000	1	0	1	99999	0	99999
Nipple 2	0	8000	0	8000	0	1	0	0	99999	0
Connector 1	41000	0	0	41000	1	0	0	99999	0	0
Connector 2	0	15840	6160	22000	0	1	1	0	99999	99999
	i									

Solution (=decision variable values)

Total times

j	Biglia	Miyano	Okuma
Sleeve 1	0	0	64639
Sleeve 2	0	63247	0
Nipple 1	48296	0	31705
Nipple 2	0	27432	0
Connector 1	86104	0	0
Connector 2	0	43721	25361
Total loading times	134400	134400	121705

Objective function value

Z = 390505

← Spreadsheet model

Mathematical optimization model:

$$\text{Min } Z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + F\,y_{ij}$$

so that

$$\sum_{j=1}^{n} c_{ij}x_{ij} \le s_i \text{ for } i = 1,..,m$$

$$\sum_{i=1}^{m} x_{ij} = d_j \text{ for} j = 1,..,n$$

$x_{ij} \le My_{ij}$ for all i and j

$x_{ij} \ge 0$ for all i and j.

Integer programming – production control example

Objective is to allocate six jobs to one machine so, that jobs end as close to the deadline as possible. Job durations are given, and the machine processes only one job at a time

Time advances in discrete steps in this model and commonly in production planning models, but especially in operative control a continuous time formulation is possible (here too!)

$$\text{Min} \sum_{\forall i} [f_i - d_i]^2$$

$$f_i = \sum_{\forall t} t x_{it}, \text{ for all } i$$

$$\sum_{\forall t} x_{it} = 1, \text{ for all } i$$

$$y_{it} = \sum_{u=t}^{t+p_i-1} x_{iu}, \text{ for all } i, t$$

$$p_i = \sum_{\forall t} y_{it}, \text{ for all } i$$

$$\sum_{\forall i} y_{it} \le 1, \text{ for all } t$$

$$x_{it} = \begin{cases} 1, \text{if } i \text{ finishes at time } t \\ 0, \text{otherwise} \end{cases}$$

Timing, y_{it}:

Job, i	Throughput, p_i		Time, t:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	3	3						1	1	1														
2	2	2												1	1									
3	3	3																	1	1	1			
4	3	3														1	1	1						
5	3	3				1	1	1																
6	2	2										1	1											
				1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1			

Finishing time, x_{it}:

	d_i Dead line	Time, t:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	8								1														
2	9												1										
3	17																			1			
4	15																1						
5	6					1																	
6	8										1												

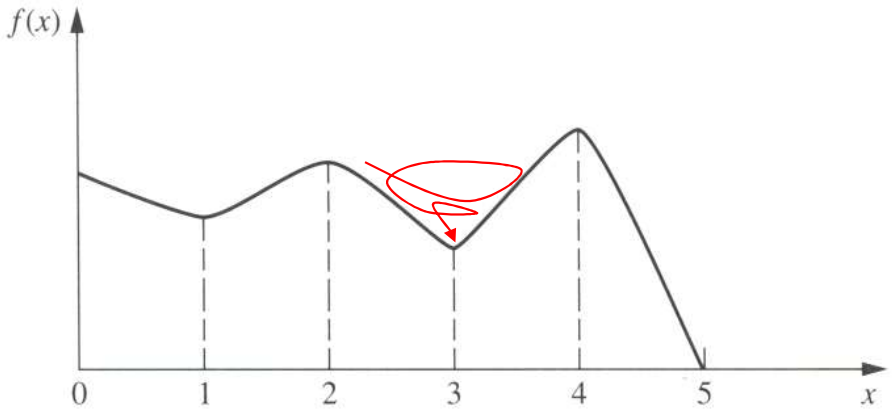
	f_i Finished	Difference	Diffeence^2
1	7	-1	1
1	11	2	4
1	18	1	1
1	15		
1	4	-2	4
1	9	1	1
			11

Solving integer problems using heuristic methods

- Optimization problems are often so large and difficult, that one has to resort to heuristic methods. This is particularly so, when problems are non-linear, non-smooth, and/or non-convex
- In heuristic methods values of decision variables are changed according to some simple rule, and after calculation of objective value, next action is decided. Usually greedy approach is applied, in which the new decision variable values are taken as the new starting point in search, if the objective value improved.
- Heuristic methods are very efficient, but typically only good values for variables are found – not the optimal values. Even if the solution would be optimal, this is not necessarily known. On the other hand, in practice a good solution is usually sufficient
- A simple heuristic is to change each variable at a time and look for the best change.
- In production control problems a better schedule can be searched for by interchanging two jobs in the sequence at a time.

Solving integer problems using heuristic methods

- Problem with heuristic methods is that the algorithm may find a poor local optimum or start circling around without improvement
- These problems can be circumvented by (partial) random change of solution and restart, or keeping record of recently tried solutions
- Methods of former type are among others *genetic algorithms* and *simulated annealing*, and *tabu search* is of the latter type



TF KnowNet

Aalto University
School of Engineering

thank you

For questions please contact:
Prof. Esko NIEMI (esko.niemi@aalto.fi)

TF KnowNet Consortium:

LMS Laboratory for Manufacturing Systems & Automation

VOLVO VOLVO GROUP

Volkswagen

PRIMA INDUSTRIE

POLITECNICO MILANO 1863

Technische Universität Braunschweig

Aalto University

MONDRAGON UNIBERTSITATEA

eit Manufacturing

Integer programming example

Selection of nests to be cut from a set of existing nests

$$\text{Min} \quad Z(x) = \sum_{i=1}^{M} G_i \left(\sum_{j=1}^{N} A_{ij} x_j + L_i - R_i \right)$$

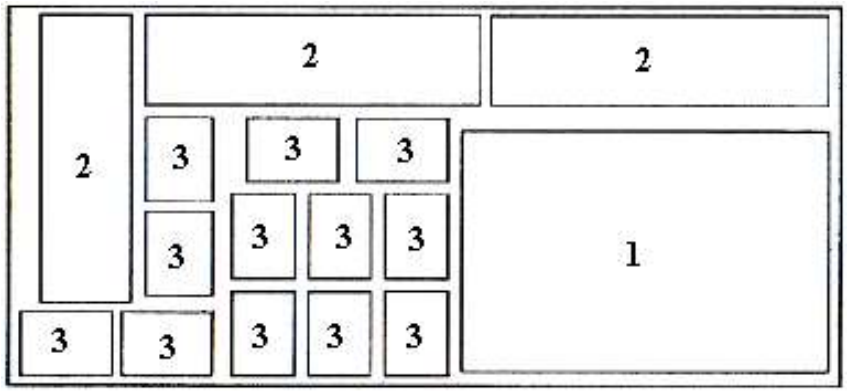
so that

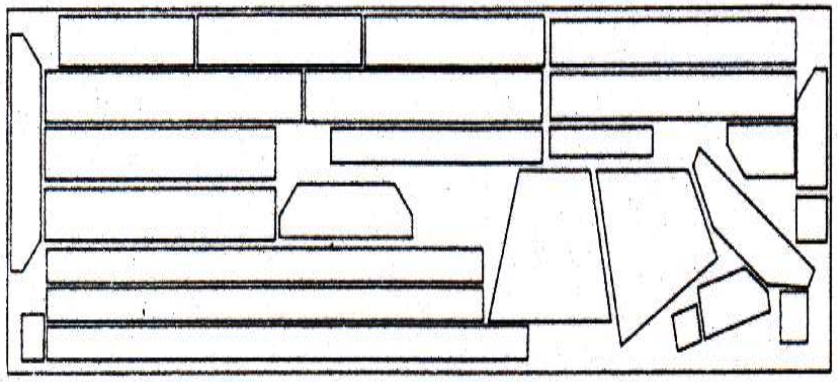
$$\sum_{j=1}^{N} A_{ij} x_j + L_i \geq R_i, \qquad i = 1,..M,$$

$$x_j \geq 0, \text{ integer},$$

where

M **= number of part types**
N **= number of nests**
G_i **= Weight of part *i* (value)**
A_{ij} **= parts *i* in nest *j***
x_j **= number of nests *j* cut**
L_i **= parts *i* in stock already**
R_i **= demand of parts *i***





Integer programming – nonlinear example

Tool magazine optimization

Problem is the classical Quadratic Assignment Problem (QAP). Objective is to find such an order for tools in a magazine of N slots for *M* tools that magazine travel is minimized. The magazine travel with tool *i* in slot *k* and tool *j* in slot *l* is c_{ijkl}, which is the product of distance between slots *k* and *l* and number of tool changes between tools *i* and *j*:

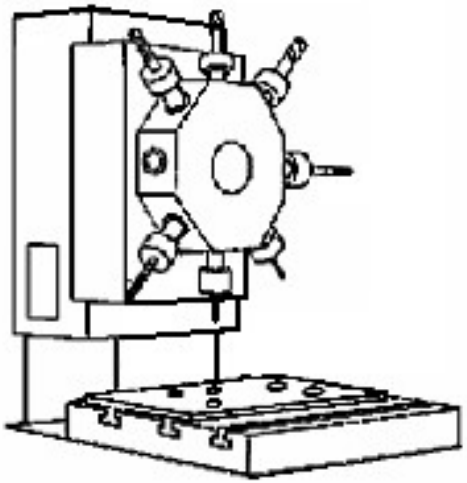
$$\text{Min} \sum_{i=1}^{M}\sum_{k=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{N} c_{ijkl}\, x_{ik}\, x_{jl}$$

st

$$\sum_{i=1}^{M} x_{ik} \le 1 \qquad k = 1,\ldots,N$$ **Every slot is occupied at most by 1 tool**

$$\sum_{k=1}^{N} x_{ik} = 1 \qquad i = 1,\ldots,M$$ **Every tool is in a slot**

x_{ik} = 1 if tool *i* is in slot *k*, otherwise 0.



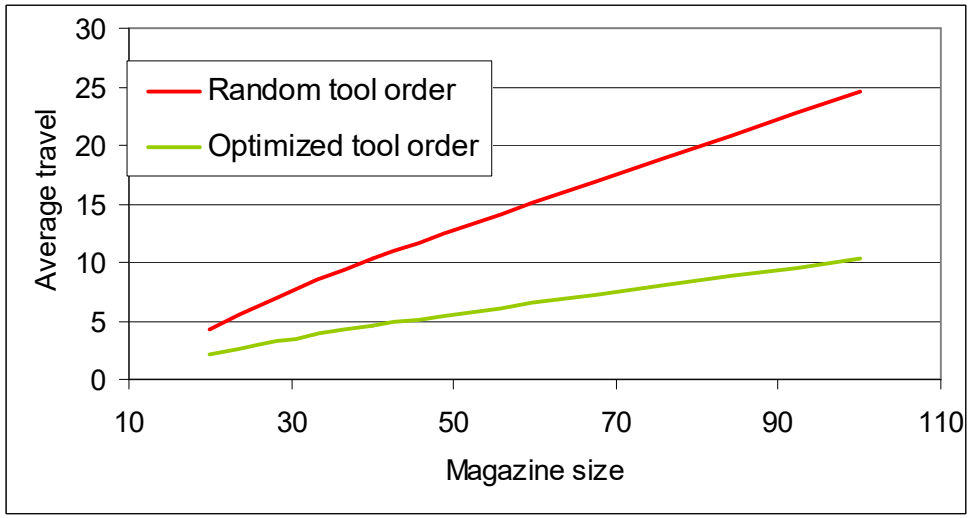
Number of possible relevant permutations is (N - 1)!/2

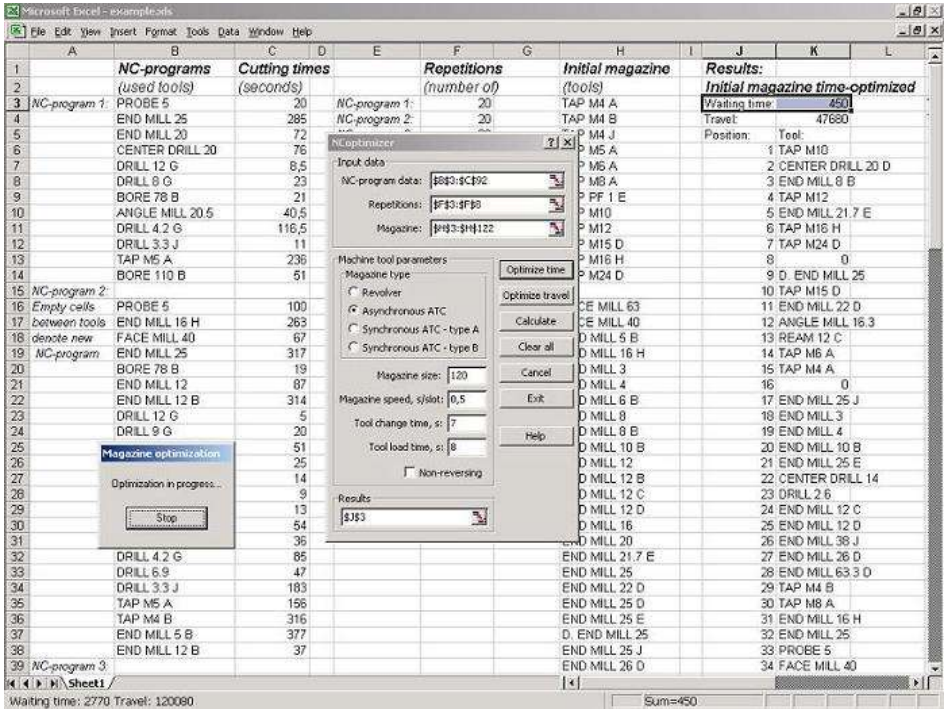
A machine with a buffering tool changer and magazine is a different story!

Tool magazine heuristic optimization

Optimization can be done for example using following heuristic:

1. **Locate tools in random slots and calculate total travel for the set of given NC programs**
2. **Try all possible pairwise interchanges of tools and calculate travel every time. Tools are returned to the previous order after each interchange**
3. **If result improved at any interchanges, freeze the best order and return to step 1**
4. **Otherwise stop**





eit Manufacturing

Aalto University
School of Engineering

Fitting models to data | 24.1.2022

Esko Niemi | Helsinki

EIT Manufacturing is supported by the EIT, a body of the European Union

TF KnowNET

1

Outline

TF KnowNET

Linear regression

- Introduction
- Simple linear regression – Introductory example
- Multiple linear regression – Fitting the model
- Multiple linear regression – Interpretation of meaning
- Multiple linear regression – Significance
- Multiple linear regression – Assumptions and limitations

Neural networks

eit Manufacturing

Aalto University
School of Engineering

2

2

Regression models

TF KnowNET

- Regression models are used to find and model a statistical dependence between variables. This is done by fitting a function to collected input (independent) and output (dependent) data
- The type of function is chosen based on the type of assumed dependence
- The values of the parameters of the regression function are defined so that some fitting criterion is optimized, usually it is the sum of squares of distances between measured output data and values predicted by the model (Least Squares Method)

Linear regression

- Useful when there is reason to believe that dependencies between inputs and outputs are linear
- Does not explain causality. On the other hand, describes well reality
- Quality of dependence can be evaluated and understanding of the process in question increases
- Very useful in many kinds of production related applications: cost estimation, work content estimation, modeling of various production processes, market demand prediction etc.

eit Manufacturing

Aalto University
School of Engineering

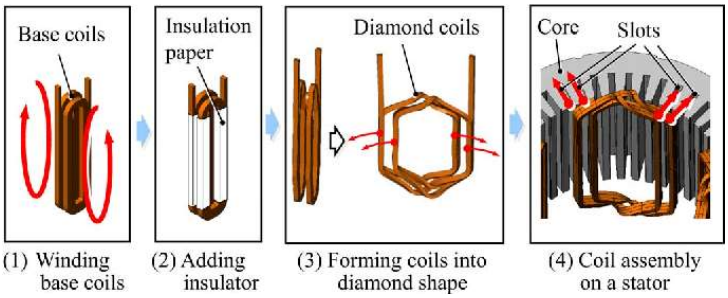
3

3

Linear regression – Stator example

TF KnowNET

We predict work content of manufacturing of stator coils of large electrical AC motors based on delivered orders. Technical specifications of orders are known and working hours have been recorded.



https://www.semanticscholar.org/paper/Motor-Stator-With-Thick-Rectangular-Wire-Lap-for-Ishigami-Tanaka

eit Manufacturing

Aalto University
School of Engineering

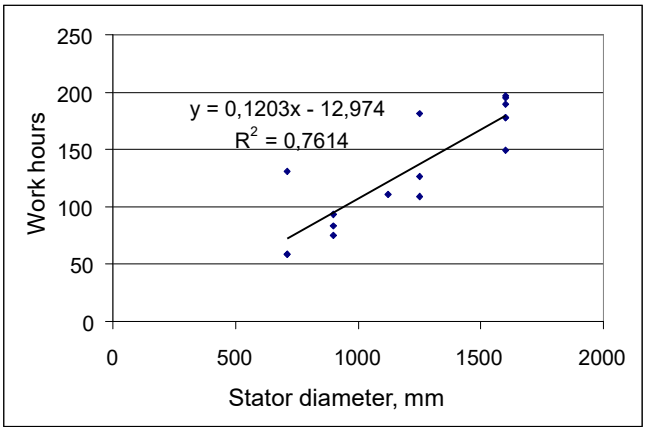
4

4

Simple linear regression – Stator example

TF KnowNET

Regression based on one property: We assume that the stator diameter affects the number or size of the coils and therefore the working hours needed to make them.



eit Manufacturing

Aalto University
School of Engineering

5

5

Multiple linear regression – Stator example continues

TF KnowNET

Many factors affect the amount of work required. We want to improve our model and assume that the following are important: Stator diameter (size), stator length (size), number of poles (number of coils in a stator) and voltage (insulation layer thickness). All of them can be taken into account in multiple linear regression analysis.

Linear model is of form:

$$y = b_n x_n + b_{n-1} x_{n-1} + b_{n-2} x_{n-2} + \ldots + b_1 x_1 + b_0$$

Here $x_1 \ldots x_n$ are input variables and $b_0 \ldots b_n$ are constants, the value of which must be determined so that the model predicts as well as possible.

eit Manufacturing

Aalto University
School of Engineering

6

6

Multiple linear regression – Stator example continues

Stator coil manufacturing working hours						
					Four variable model error	
Hours	Stator diameter	Stator length	Number of poles	Voltage	Predicted hours	Error
59	710	650	10	4000	48.4	10.6
83	900	1150	10	4000	90.0	-7.0
111	1120	850	12	5000	105.0	6.0
181	1250	1250	10	13000	166.0	15.0
195	1600	1050	14	13000	197.8	-2.8
126	1250	1350	10	5000	136.9	-10.9
149	1600	750	12	9000	162.7	-13.7
93	900	950	10	11000	110.2	-17.2
178	1600	1050	12	9000	176.9	1.1
190	1600	1350	14	800		
109	1250	750	12	400		
75	900	950	10	300		
197	1600	1250	12	1100		
178	1600	1050	12	600		
131	710	1350	10	1500		
59	710	750	6	700		

Left: Data and calculated hours. Note that R^2 is now 0.967!

Right: Results of fitting model using Excel tools (Tools -> Data Analysis -> Regression)

Regression Statistics		*Formulas*
Multiple R	0.98319774	0.98319774
R Square	0.966677796	0.966677796
Adjusted R Square	0.954560631	0.954560631
Standard Error	10.63204001	10.63204001
Observations	16	

ANOVA

	df	*SS*	*MS*	*F*	*S*
Regression	4	36072.3	9018.1	79.77755504	
Residual	11	1243.4	113.0		
Total	15	37315.8			

	Coefficients	*Standard Error*	*t Stat*	*P-value*
Intercept	-86.9467944	19.1529192	-4.53961057	0.000844404
Stator diameter	0.094769015	0.012342802	7.678079546	9.63187E-06
Stator length	0.047320526	0.013166316	3.594059789	0.004213544
Number of poles	2.036548752	2.280536175	0.893013132	0.390968525
Voltage	0.004227445	0.000813669	5.195531958	0.000296524

eit Manufacturing

Aalto University
School of Engineering

7

7

Multiple linear regression

We fit the multidimensional line equation to our data set by determining the values of b_i so that we minimize

SSE = Sum of Squares of Errors = $\Sigma((Y_i - \text{Est}Y_i)^2)$,

where

Y_i = measured data

EstY_i = values predicted (calculated) using model

Coefficient of determination, R^2 expresses how big a share of variation in data is explained by the model:

$R^2 = (1 - (SSE/SST))$,

where

SST = Total Sum of Squares = $\Sigma((Y_i - \text{Mean}Y)^2)$,

where

Mean$Y$ = average of data point values

eit Manufacturing

Aalto University
School of Engineering

8

8

Multiple linear regression – Interpretation of model

TF KnowNET

Coefficients b_i of the model determine how much a change in the value of variable x_i affects the result. This effect can be evaluated by:

- Calculating the real contribution. This is done by multiplying average of the related input data values with the coefficient. The result is the expected contribution of each input factor to the average result.
- Standardizing the model. Average of the data values of each type is subtracted from the individual values and divided by the standard deviation of each type of input values. We obtain values, the average of which is 0 and standard deviation 1. By fitting the model to this standardized data the obtained coefficients are comparable and give the relative contribution of the different inputs to the output values. In other words: we can directly conclude which factors affect the result most and are most important

eit Manufacturing

Aalto University
School of Engineering

9

9

Multiple linear regression - Significance

TF KnowNET

SEE = Standard Error of Estimate = Sqrt(*SSE*/*df*),

where

df = degrees of freedom = $n - k - 1$,

where

n = number of measurements

k = number of input variables

SEE is used for calculating significance of model and its parameters and confidence interval of parameter values and confidence interval of prediction for individual values and the average.

eit Manufacturing

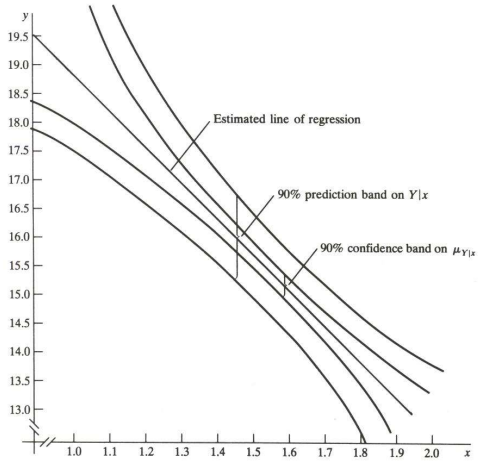
Aalto University
School of Engineering

10

10

Multiple linear regression - Significance

Simple linear regression model and confidence intervals for average and a single measurement



Note! You do not get these from Excel's analysis tools, but you get them from statistical software packages like Statistix

11

Multiple linear regression - Significance

Significance of R and R^2 and the model can be tested with F test. F depends on R^2, numbers of measurements and variables:

$$F_{k,\, n-k-1} = (\mathrm{Est}Y_i^2/k) \,/\, SEE = [R^2/k] \,/\, [(1 - R^2)/(n - k - 1)].$$

Significance testing is important, because with a large number of variables the R^2 can be high, although SEE is high. In this case the model may not be good with other data than that it was fitted to. The same purpose is served by:

$$\text{Adjusted } R^2 = 1 - ((1-R^2)(N\text{-}1 \,/\, N - k - 1)).$$

Adjusted R^2 approaches R^2 when k decreases.

12

Multiple linear regression - Significance

Significance of R and R^2 and the model can be tested with F test. F depends on R^2, numbers of measurements and variables:

$$F_{k,\, n-k-1} = (\mathrm{Est}Y_i^2/k) \,/\, SEE = [R^2/k] \,/\, [(1 - R^2)/(n - k - 1)].$$

Significance testing is important, because with a large number of variables the R^2 can be high, although SEE is high. In this case the model may not be good with other data than that it was fitted to. The same purpose is served by:

$$\text{Adjusted } R^2 = 1 - ((1-R^2)(N\text{-}1 \,/\, N - k - 1)).$$

Adjusted R^2 approaches R^2 when k decreases.

eit Manufacturing

Aalto University
School of Engineering

13

13

Multiple linear regression - Assumptions and limitations

- Data samples should represent well the typical behaviour of the phenomenon modelled. The input values (and outputs) should cover the whole typical value range of each variable and typical combinations of input values should appear in the data.
- The model should only contain the factors that truly contribute to the result. Procedures for eliminating unnecessary variables exist and are implemented is statistics software. The number of variables affects the required size of data sample and generally it should be at least 10 – 20 times the number of variables.
- The modelled phenomenon should behave linearly. Linearity can be evaluated by graphing the errors as functions of each variable separately. No distinguishable trend should exist.

eit Manufacturing

Aalto University
School of Engineering

14

14

Multiple linear regression - Assumptions and limitations

TF KnowNET

- Input variables should be independent of each other. Dependence is called multicollinearity and it can be tested by doing regression analysis of each (x_i) variable's dependence on all the other variables. The R^2 obtained should be low (< 0,2). Multicollinearity shows as large standard error of coefficients (b_i) and low reliability of the model.
- Errors should ne normally distributed. This can be evaluated by drawing a histogram for each variable's errors.
- Regression analysis only describes correlations between variables. It does not explain the reasons for correlations. For example, insurance compensation for losses due to fire correlate strongly with the number of firefighters at the location of fire => ?

eit Manufacturing

Aalto University
School of Engineering

15

15

Outline

TF KnowNET

Neural networks

- Introduction
- Neuron
- Network
- Training a single linear neuron
- Nonlinear transfer functions
- Scaling output
- Modeling and using a neural network
- Example application: Estimating cost of anodizing process
- Conclusions

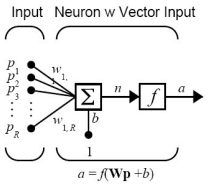
eit Manufacturing

Aalto University
School of Engineering

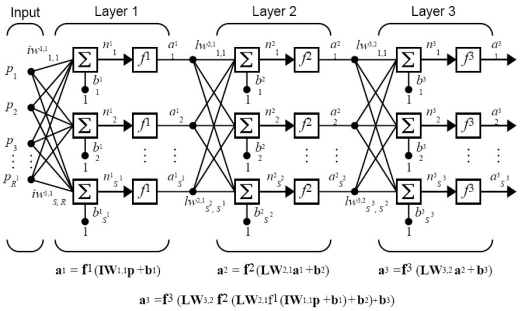
16

16

Neural networks



Machine learning, an important category of *Artificial intelligence*, is usually realized with *Neural networks*



Most of the figures in this presentation are from *Matlab* documentation

17

17

Neural networks

- Neural network is a ”black box” between given inputs and obtained outputs in a similar fashion to a regression model
- Realization of a neural network and setting of parameters, however, is different, and
  - Neural networks can be used to model nonlinear systems
  - Actually, almost no limiting assumptions about the modeled system’s behavior need to be made
  - Several outputs can be modelled at the same time unlike (most) regression models
- On the other hand, similar justified conclusions about reliability of neural networks can not be done as about linear regression models. They are “unpredictable”.
- Neural networks can be used for classification and clustering with or without training, but here we only focus on process modeling using static *feedforward* networks

18

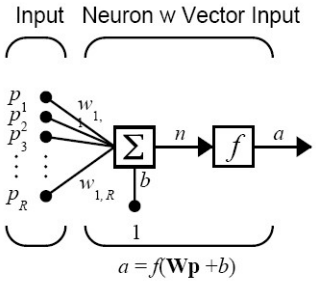
18

Neuron

TF KnowNET

Neural network consists of neurons, which receive inputs (p_i), weigh the inputs with weights (w_i), possibly add a constant, bias (b), calculate the sum (n) and change it with a transfer function *($a = f(n)$)*.

One neuron:



Where...

R = number of elements in input vector

eit Manufacturing

Aalto University
School of Engineering

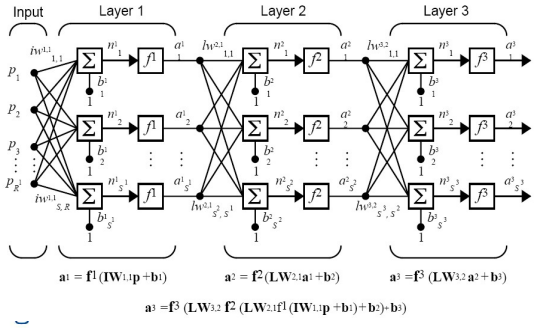
Network of neurons

TF KnowNET

Neurons can be combined with outputs and inputs to form (multiple) layers:

- A neuron can take several inputs
- A neuron has only one output,
- but this output can be connected as input to several neurons
- The last (output) layer has to have as many neurons as there are final outputs
- Layers inside the network are called hidden layers

Input layer only distributes inputs to first neuron layer



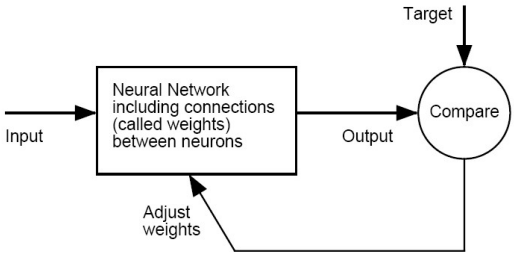
eit Manufactur

Aalto University
School of Engineering

Fitting network to data - training

TF KnowNet

Network is trained with input-target pairs so, that the output is compared to target and the weights and bias are adjusted according to error and learning algorithm recursively and repeatedly. This is continued as long as error converges or some preset target value for error is reached. The best known learning algorithm is called backpropagation.



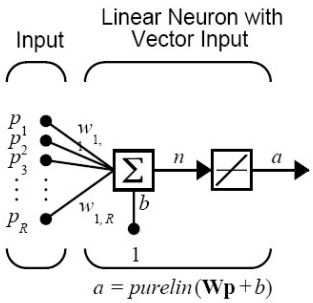
eit Manufacturing

Aalto University School of Engineering

Linear neuron

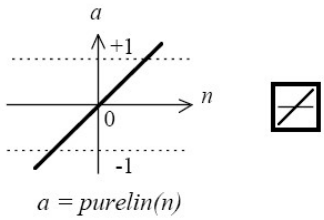
TF KnowNet

Clearly, one neuron with a linear transfer function (that does nothing), functions like a linear regression model



Where...

R = number of elements in input vector



Linear Transfer Function

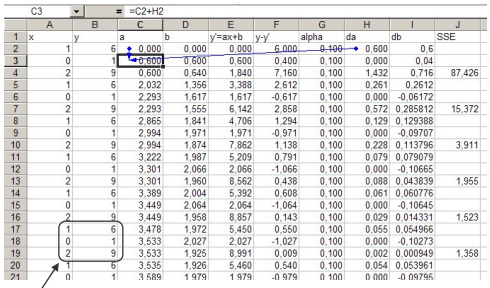
$$n = w_{1,1}p_1 + w_{1,2}p_2 + \ldots + w_{1,R}p_R + b$$

eit Manufacturing

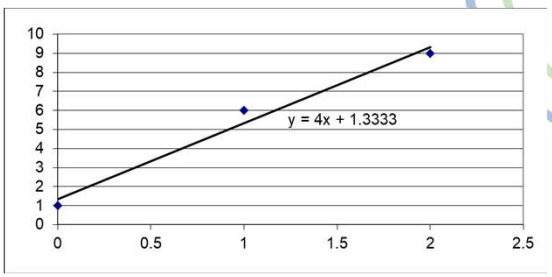
Aalto University School of Engineering

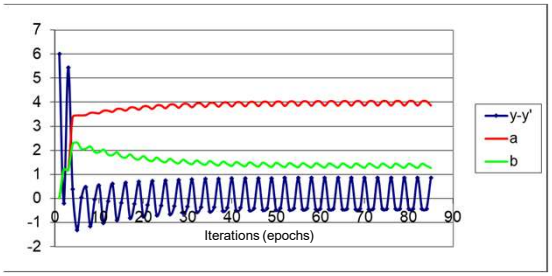
Training a linear neuron - example

TF KnowNet



Training data set





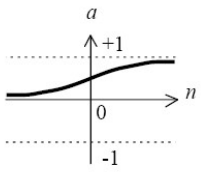
eit Manufacturing

Aalto University
School of Engineering

Nonlinear transfer functions

TF KnowNet

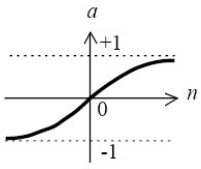
For modeling nonlinear systems transfer functions have to be nonlinear. Sigmoid functions are a typical choice:



$a = logsig(n)$

Log-Sigmoid Transfer Function

$a = 1/(1 + e^{-n})$



$a = tansig(n)$

Tan-Sigmoid Transfer Function

$a = (1 - e^{-2n})/(1 + e^{-2n})$

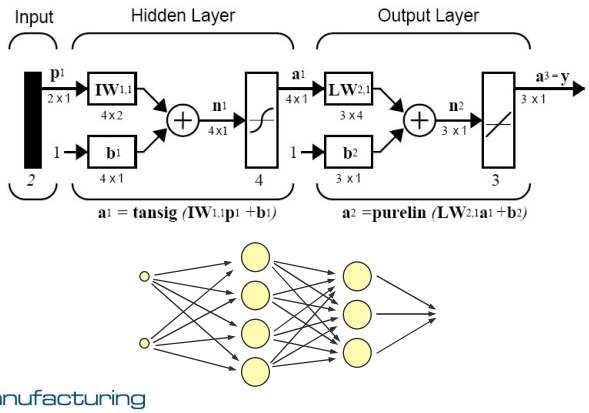
eit Manufacturing

Aalto University
School of Engineering

Scaling output

TF KnowNET

Because sigmoid functions only return values in range (0, 1) or (-1, 1), output layer may have a linear transfer function, which scales the results:



eit Manufacturing

Aalto University
School of Engineering

Neural network modeling process

TF KnowNET

1. Define purpose, inputs and outputs and collect data.
2. Network design and creation (neuron types, parallel neurons and layers…).
3. Network *training*. Training requires several learning cycles (epochs), during which the weights are adjusted. This may lead to a situation, in which it "memorizes" the training data, but does not work well with other data from the same system. This is common if the modeled process is highly nonlinear or otherwise erratic or the data is not representative. It is essential to *validate* and *test* the network also with other than the training data.
4. Training can be automated as follows: Data is divided to training, validation and test sets. Only training set is used for adjusting weights, but during the process the network is tested with the validation set. When fit to the validation set does not improve anymore, training is stopped. This method aims at ensuring network's generalization capability.
5. Network is tested with the test set and used.

eit Manufacturing

Aalto University
School of Engineering

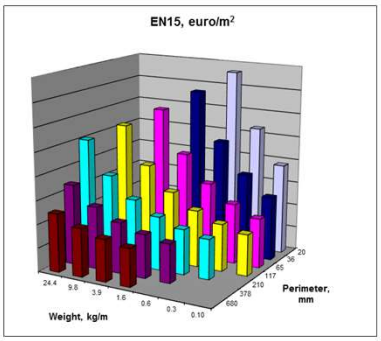
Example: cost of anodizing extruded aluminium profiles

TF KnowNET

Anodizing is a process in which the thickness of the (natural) oxidation layer on aluminium is electrolytically increased. Aluminium profiles are anodized in large automatic plants, in which the profiles are manually loaded to the process and finally unloaded. Significant costs are related to labor, energy, chemicals and their handling, rent, maintenance, and initial investment. – Based on the process it can be concluded that the costs can be allocated to products based on their area, weight and oxidation layer thickness.

Cost data obtained from plant cost data collection system for various profiles:

	A	B	C	D
1	**Data**			
2	[µm]	[mm]	[kg]	[euro/m2]
3	**Thickne**	**Perimeter**	**Weight**	**Cost**
4	5	20	0.09	5.6
5	10	36	0.2	5.2
6	15	65	0.6	5.4
7	20	117	1.6	5.4
8	5	210	3.9	4.1
9	10	378	9.8	4.1
10	15	680	24.4	4.2
11	20	65	0.10	3.8



eit Manufacturing

Aalto University
School of Engineering

27

27

Example: cost of anodizing extruded aluminium profiles

TF KnowNET

A view from an automated anodizing plant

Extruded aluminium profiles

eit Manufacturing

Aalto University
School of Engineering

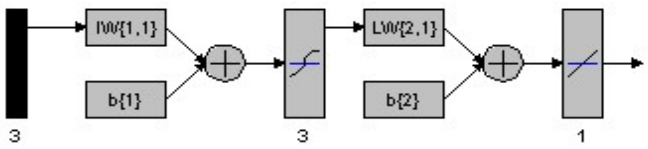
28

28

Example: cost of anodizing extruded aluminium profiles

TF KnowNET

Chosen network topology:



Weights for trained network (Matlab) and a simulation result on spread sheet:

	A	C	E	G	I	K	M	O
1	Input layer	Hidden layer				Output layer		
2	Inputs *p*	Weights *w*	Bias *b*	Weighed sums	Tan-Sigmoid	Weights *w*	Bias *b*	Output *a*
3	5	-5.22	12.2655	-8.79	-1.000	-0.7	4.08	4.926
4		0.31				0.3		=I3*K3+I9*K4+I15*K5+M3
5		43.2956				0.5		
9	20	0.36	-6.8176	-1.42	-0.890			
10		0.25			=(1-EXP(-2*G9))/1/(1+EXP(-2*G9))			
11		29.71						
15	0.09	0.92	19.1583	13.01	1.000			
16		0.61		=C15*A3+C16*A9+C17*A15+E15				
17		15.46						

Aalto University
School of Engineering

Example: cost of anodizing extruded aluminium profiles

TF KnowNET

Coefficients and R^2 for a linear regression model (Excel) :

Example calculation:

0.048*5 – 0.0071*20 + 0.18*0.09 + 4.45 = 4.57

Coefficiens for standardized data:

		Coefficients
16		
17	Intercept	2.31755E-16
18	Thickness	0.27453502
19	Perimeter	-1.368670503
20	Weight	1.130921913

	A	B	C
1	SUMMARY OUTPUT		
2			
3	*Regression Statistics*		
4	Multiple R	0.695649449	
5	R Square	0.483928156	
6	Adjusted R Square	0.387164685	
7	Standard Error	0.792960684	
8	Observations	20	
9			
10	ANOVA		
11		*df*	*SS*
12	Regression	3	9.433959736
13	Residual	16	10.06058635
14	Total	19	19.49454608
15			
16		*Coefficients*	*Standard Error*
17	Intercept	4.456101086	0.51808277
18	Thickness	0.048485771	0.032461021
19	Perimeter	-0.007135217	0.002209216
20	Weight	0.187033182	0.069502135
21			
22			
23			
24	RESIDUAL OUTPUT		
25			
26	*Observation*	*Predicted Cost*	*Residuals*
27	1	4.572658579	1.008439342

Manufacturing

Aalto University
School of Engineering

Example: cost of anodizing extruded aluminium profiles

TF KnowNet

- For NN training data R^2 = 0.997. For linear regression model R^2 obtained is 0.484.
- In this simple example the number of weights is 16, although we have only three inputs and one output. In practice commercial neural network software can be used (e.g. Matlab) to find suitable network design and values for weights.
- When the network has been trained and values for weights obtained, it is easy to implement. As the parameters of cost calculation or the model itself are not (and should not be) adjusted frequently (typically 1 – 2 times a year), the model could be integrated to cost a calculation program or e.g. to a spread sheet application.
- Many other, e.g. process control applications, can be realized similarly
- Applications requiring more frequent retraining could be for example object recognition or product demand prediction.

eit Manufacturing

Aalto University
School of Engineering

31

31

TF KnowNet

Aalto University
School of Engineering

Thank you

For questions please contact:
Prof. Esko NIEMI (esko.niemi@aalto.fi)

TF KnowNet Consortium:

LMS Laboratory for Manufacturing Systems & Automation
VOLVO VOLVO GROUP
Volkswagen
PRIMA INDUSTRIE
POLITECNICO MILANO 1863
Technische Universität Braunschweig
Aalto University
MONDRAGON UNIBERTSITATEA

eit Manufacturing

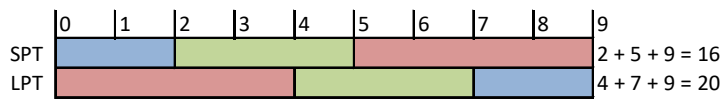
32

Production control

- Single machine models
  - Prioritizing
  - Optimizing models
  - Optimal algorithms
  - Heuristics
  - Models with set-ups
- Flow shops and lines
- Parallel machines
- Job shops
- FM-systems
- Lean methods
  - CONWIP
  - Kanban
  - Bottleneck control

Single machine scheduling

- We assume that we have a set of jobs available for scheduling
- If there are no constraints related to starting or finishing jobs, a no-wait schedule (no gaps between jobs) is optimal for normal criteria
- Makespan is not affected by the order of jobs
- We may take that each job in the beginning of the schedule delays all later jobs, therefore
- **Average (and total) flow time (starting from zero) is minimized by Shortest Processing Time (SPT) order**:

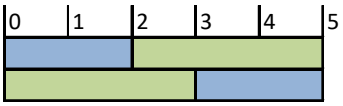


- To minimize WIP, processing times may be weighted by dividing them by the value of jobs (parts)

Total flow time minimization - SPT

Optimality of SPT rule for total flow time is easy to prove as follows:

- Let us examine two consecutive jobs. Independent of the order, makespan, that is finishing time of the last job, is the sum of the processing times
- Therefore only the finishing time of the first job affects the sum of completion times
- The Sum is minimized by scheduling the shorter job first
- In a longer schedule exchanging two consecutive jobs does not affect the timing of other jobs in any way
- By doing exchanges of consecutive jobs so that the shorter job always comes first as long as possible we end up in SPT order



Total flow time minimization – MILP model

- SPT algorithm is a sorting process, which are (can be) polynomial
- This means that the solution time increases in a polynomial relationship to size of the problem (in this case the number of jobs)
- MILP (Mixed Integer Linear Program) optimization model does not utilize this fact but leaves the solution to the solver
- In MILP formulation
  1. Timing of jobs is generated but constrained so that they do not overlap

  or

  2. Order of jobs is generated and timing is constrained so that the next job does not start before the previous job has ended

  or

  3. As a discrete time problem like the alternative 1 above

Total flow time minimization – IP model 1 (”Manne’s formulation”)

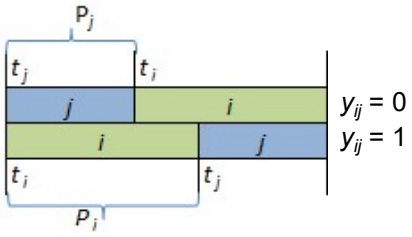
Starting times of jobs are generated, but constrained so that they do not overlap

Constants:

I	number of jobs
P_i	Duration of job i
M	Large number

Decision variables:

t_i	starting time of job i
y_{ij}	takes value 1, is job i precedes job j, otherwise 0



$y_{ij} = 0$

$y_{ij} = 1$

$$\text{Min} \sum_{\forall i} (t_i + P_i) \quad \text{st.}$$

$$t_i \geq 0, \qquad \forall i$$

$$My_{ij} + (t_i - t_j) \geq P_j, \qquad \forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\}$$

$$M(1 - y_{ij}) + (t_j - t_i) \geq P_i, \qquad \forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\}$$

$$y_{ij} \in \{0,1\}, \qquad \forall i, j$$

Total flow time minimization – IP model 2 (”Wagner’s formulation”)

Order of jobs is generated and timing is constrained so that the next job does not start before the previous job has ended

Now decision variables y_{ij} take value 1, if job i is in position j in the schedule, otherwise 0

$$\text{Min} \sum_{\forall j} \left(t_j + \sum_{\forall i} P_i y_{ij} \right)$$

$$t_j \geq 0, \qquad \forall j$$

$$\sum_i y_{ij} = 1, \qquad \forall j$$

$$\sum_j y_{ij} = 1, \qquad \forall i$$

$$t_j + \sum_{\forall i} P_i y_{ij} \leq t_{j+1}, \qquad \forall j \in \{1,..,I-1\}$$

$$y_{ij} \in \{0,1\}, \qquad \forall i, j$$

			j					
Job, i	P_i	y_{ij}	1	2	3	4	5	
1	4		-0	1	0	0	0	1
2	6		0	0	1	0	0	1
3	8		0	0	0	1	0	1
4	3		1	-0	0	0	0	1
5	9		0	0	-0	0	1	1
			1	1	1	1	1	
		t_j	0	3	7	13	21	
		P_j	3	4	6	8	9	
		c_j	3	7	13	21	30	74

Total flow time – IP

Job, i	P_i	t_i	P_i+t_i	y_{ij}	j 1	2	3	4	5
1	4	3	7		1	1	0	1	
2	6	7	13			1	0	1	
3	8	13	21				0	1	
4	3	0	3					1	
5	9	21	30						
			74						

Model 1

y_{ij} take value 1, if job i precedes j, otherwise 0

i \ t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1																														
2																														
3																														
4																														
5																														

Job, i	P_i	y_{ij}	j 1	2	3	4	5	
1	4		0	1	0	0	0	1
2	6		0	0	1	-0	0	1
3	8		0	0	0	1	0	1
4	3		1	0	0	0	0	1
5	9		0	0	0	0	1	1
			1	1	1	1	1	
		t_j	0	3	7	13	21	
		P_j	3	4	6	8	9	
		c_j	3	7	13	21	30	74

Model 2

y_{ij} take value 1, if job i is in position j in schedule, otherwise 0

Total flow time – discrete time model

- End times x_{it} are generated for jobs and running time is calculated backward and indicated with auxiliary variables y_{it}
- Job overlap is prevented
- Each job ends once

t time

Decision variables:

x_{it} take value 1, if job i ends at time t

y_{it} take value 1, if job i is processed at time t

$$\text{Min} \sum_{\forall i} \sum_{\forall t} t x_{it}$$

$$\sum_{\forall t} x_{it} = 1, \qquad \forall i$$

$$y_{it} = \sum_{u=t}^{t+P_i-1} x_{iu}, \qquad \forall i, t$$

$$\sum_{\forall i} y_{it} \le 1, \qquad \forall t$$

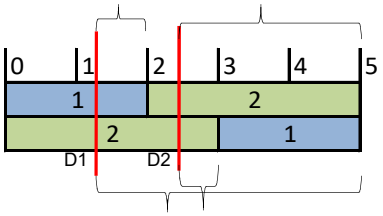
Job, i	Duration, P_i	Time, t	Timing, y_{it}: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
1	4	4	
2	6	6	
3	8	8	
4	3	3	
5	9	9	
Σ			1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Aika, t	Finishing, x_{it}: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	Σ	f_i End
1		1	7
2		1	13
3		1	21
4		1	3
5		1	30
			74

Maximum tardiness minimization - EDD

Maximum tardiness is minimized by sorting them in the Earliest Due Date order:

- We examine two consecutive jobs and tardinesses in different alternatives:



- From the figure can be seen, that tardiness is maximized when the job with earlier due date is processed later, because a worse alternative can not exist
- The result can not be worse in the other alternative
- Alternatives, in which no tardiness (or partial) exist are not relevant (or do not change the result)
- By doing exchanges of consecutive jobs as long as possible we end up in EDD order

Maximum tardiness minimization – IP model

IP model based on Manne's model

New constants:

D_i Due date of job i

And decision variable:

f maximum tardiness

Min f	
$t_i + P_i - D_i \le f,$	$\forall i$
$t_i \ge 0, f \ge 0,$	$\forall i$
$My_{ij} + (t_i - t_j) \ge P_j,$	$\forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\}$
$M(1 - y_{ij}) + (t_j - t_i) \ge P_i,$	$\forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\}$
$y_{ij} \in \{0,1\},$	$\forall i, j$

Total tardiness minimization

For total tardiness minimization no simple algorithm exists. Many heuristics are based on that

- If the schedule is loose and there are few late jobs, EDD rule probably works well
- If the schedule is tight and most jobs are late, SPT rule probably works well

IP model

New variables f_i express tardiness of job i

$$
\begin{array}{ll}
\text{Min } \sum_{\forall i} f_i & \\
t_i + P_i - D_i \le f_i, & \forall i \\
t_i \ge 0,\ f_i \ge 0, & \forall i \\
My_{ij} + (t_i - t_j) \ge P_j, & \forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\} \\
M(1 - y_{ij}) + (t_j - t_i) \ge P_i, & \forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\} \\
y_{ij} \in \{0,1\}, & \forall i, j
\end{array}
$$

Minimization of number of tardy jobs

Minimizing number of tardy jobs is necessary when on time delivery is maximized. Efficient algorithms exist for this. For example:

Jobs are first sorted according to due dates, then

1. Find first tardy job k
2. Find longest job 1..k and remove it and move last in schedule
3. Return to step 2 until there are no more tardy jobs (in the original schedule)

Example:

Job i	Duration P_i	Due date D_i
1	1	2
2	5	7
3	3	8
4	9	13
5	7	11

1. {1,2,3,5,4} k = 3, longest in 1..3 is job 2
2. {1,3,5,4} {2} k = 4, which itself is longest
3. {1,3,5} {2,4}
4. No more late jobs, final schedule: {1,3,5,2,4}

Minimization of number of tardy jobs

IP model for minimization of number of tardy jobs is achieved easily as a modification of Manne's model:

New variables z_i take value 1, if job i is late, otherwise 0

$$\begin{array}{ll}
\text{Min } \sum_{\forall i} z_i & \\
t_i + p_i - D_i \le M z_i, & \forall i \\
t_i \ge 0, & \forall i \\
M y_{ij} + (t_i - t_j) \ge P_j, & \forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\} \\
M(1 - y_{ij}) + (t_j - t_i) \ge P_i, & \forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\} \\
y_{ij} \in \{0,1\}, & \forall i, j \\
z_i \in \{0,1\}, & \forall i
\end{array}$$

Single machine scheduling

- The presented optimization models are flexible, because
  - Same models are easy to modify for different objectives
  - Constraints for earliest starting times are easy to add (in this case the optimal rules presented do not apply)
  - Constraints for job precedence are easy to add
- Manne's model is often easier to apply and
  - It is easy to extend to job shops
- Wagner's model directly expresses job order and then
  - Set-up times are easy to model
  - Choice from alternative (parallel) machines is easy to model
- Discrete model has advantages of both continuous time model types, but it is computationally heavy and inaccurate because of the limited resolution

Single machine and set-ups

Total flow time minimization (Wagner's model) with set-ups

- Similar products are grouped to set-up groups
- Set-up time S_k does not realize, if products belonging to same set-up group k immediately follow each other
- Set-up time does not depend on the type of previous (different) job

Now variables s_{jk} take value 1, if set-up changes, otherwise 0.
Constants R_{ik} indicate if job i belongs to group k, in which case R_{ik} = 1

$$\text{Min} \sum_{\forall j}\left(t_j + \sum_{\forall i} P_i y_{ij} + \sum_{\forall k} S_k s_{jk}\right)$$

$$t_j \geq 0, \qquad \forall j$$

$$\sum_i y_{ij} = 1, \qquad \forall j$$

$$\sum_j y_{ij} = 1, \qquad \forall i$$

$$t_j + \sum_{\forall i} P_i y_{ij} + \sum_{\forall k} S_k s_{jk} \leq t_{j+1}, \quad \forall j \in \{1,..,I-1\}$$

$$Ms_{jk} \geq \sum_{\forall i} R_{ik} y_{ij} - \sum_{\forall i} R_{ik} y_{i,j-1}, \forall j,k$$

$$y_{ij} \in \{0,1\}, s_{jk} \in \{0,1\} \qquad \forall i,j,k$$

Makespan optimization is a relevant criterion when set-ups are taken into account

s is set to 1 if two consecutive jobs are different

Sequence dependent set-up time problem (SDSTP)

- Sequence dependent set-up times are common in manufacturing. Examples:
  - Machining concerning fixtures and tools
  - Sheet metal cutting
  - Painting
  - Injection molding etc.
- Naturally no set-up is necessary here either, if similar jobs can be scheduled consequently. In the next examples the job sequence is not considered from the delivery time perspective
- One popular heuristic is to always choose the job with the smallest set-up time next (nearest neighbor or closest insertion algorithm). The starting job affects the result. Therefore it is useful to try starting from each job
- A MILP model is also possible, although a little complicated

Heuristic solution for SDSTP

- In make-to-order production schedule is made into near future and the last job in sequence is not important
- In make-to-stock production repeating the same cycle may be beneficial

Set-up costs

From: / To: Job	1	2	3	4	5
1	-	18	3	3	6
2	19	-	9	10	5
3	9	18	-	13	20
4	6	6	1	-	2
5	17	1	13	17	-

Closest insertion -algorithm

All jobs once

Sequence	Cost
1 - 3 - 4 - 5 - 2:	3 + 13 + 2 + 1 = 19
1 - 4 - 3 - 2 - 5:	3 + 1 + 18 + 5 = 27
2 - 5 - 3 - 1 - 4:	5 + 13 + 9 + 3 = 30
3 - 1 - 4 - 5 - 2:	9 + 3 + 2 + 1 = **15**
4 - 3 - 1 - 5 - 2:	1 + 9 + 6 + 1 = 17
5 - 2 - 3 - 1 - 4:	1 + 9 + 9 + 3 = 22

Full cycle

Sequence	Cost
1 - 3 - 4 - 5 - 2 - 1:	3 + 13 + 2 + 1 + 19 = 38
1 - 4 - 3 - 2 - 5 - 1:	3 + 1 + 18 + 5 + 17 = 44
2 - 5 - 3 - 1 - 4 - 2:	5 + 13 + 9 + 3 + 6 = 36
3 - 1 - 4 - 5 - 2 - 3:	9 + 3 + 2 + 1 + 9 = **24**
4 - 3 - 1 - 5 - 2 - 4:	1 + 9 + 6 + 1 + 10 = 27
5 - 2 - 3 - 1 - 4 - 5:	1 + 9 + 9 + 3 + 2 = **24**

Optimisation model for sequence-dependent setup time problem

- SDSTP is a variation of the travelling salesman problem, but usually with
  - Asymmetric travelling (setup) times
  - Open sequences
- MILP formulation, closed loop (J = number of jobs, S_{ij} = setup time from i to j, x = binary variable, o = position in sequence, variable)

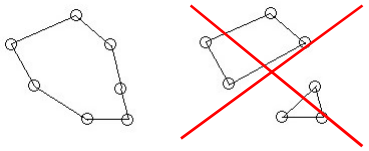
$$\text{Min} \quad \sum_{i=1}^{J}\sum_{j=1}^{J} x_{ij}S_{ij}$$

so that

$$\sum_{i=1}^{J} x_{ij} = 1 \qquad \forall j \in \{1,2,\ldots,J\}$$

$$\sum_{j=1}^{J} x_{ij} = 1, \qquad \forall i \in \{1,2,\ldots,J\}$$

$$o_i - o_j + Jx_{ij} \le J - 1 \qquad \forall 2 \le i \neq j \le J$$

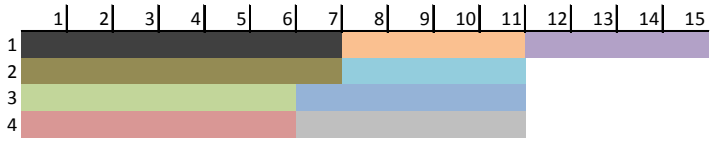


You can only go forward in sequence

However, you can jump backward once at $i = 1$

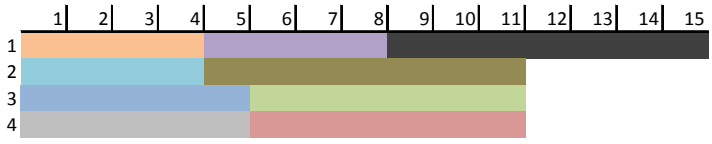
Parallel machines

- We assume, that the machines are similar to the extent that they can be used to process all jobs
- However, machines do not need to be equally fast for the MILP optimization, and the differences in speed may be job specific
- Unlike basic single machine case, makespan is also relevant criterion for parallel machines
- c_{max} (makespan) is minimized quite well by sorting in LPT order and allocating evenly (longest job first to the shortest schedule, but not always:



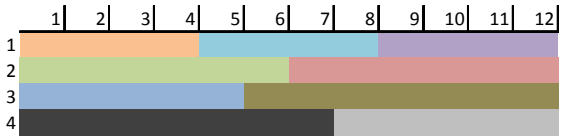
Parallel machines

- In a second step the jobs can be ordered for each machine with single machine rules
- Total flow time for the previous example is optimized using SPT rule:



Flow time
4+8+15
4+11
5+11
5+11 =74

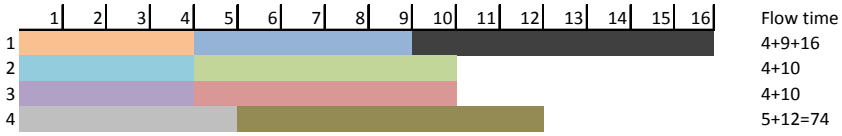
- In this case c_{max} can be optimized using MILP optimization:



Flow time: 76

Parallel machines

- Procedure, in which jobs are ordered with a simple rule and then allocated evenly to machines also usually works well
- In fact, total flow time is optimized using SPT rule. Previous example:



- Due date related criteria can not be optimized using simple rules, but a good result is usually obtained using the described two-phase procedures

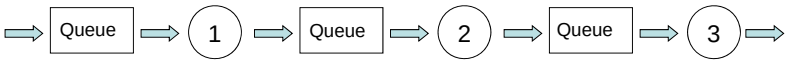
Parallel machines – MILP model

- MILP formulation allocates jobs *i* to machines *k* in order *j*. The result is realized in binary y_{ijk} values
- If machines are not similar, durations V_{ik} can be used for each job-machine combination (high value for V_{ik} for unfit machine-job combinations)
- Model minimizes total tardiness, but any objective is possible with minor modifications

$\text{Min } \sum_{\forall k}\sum_{\forall j} f_{jk}$		Total tardiness
$c_{0,k} = 0$	$\forall j,k$	
$c_{jk} = c_{j-1,k} + \sum_{\forall i} P_i y_{ijk}$	$\forall j,k$	Finishing time
$c_{jk} - \sum_{\forall i} D_i y_{ijk} \le f_{jk}$	$\forall j,k$	Tardiness determination
$\sum_i y_{ijk} \le 1,$	$\forall j,k$	Job order
$\sum_k \sum_j y_{ijk} = 1,$	$\forall i$	Each Job is executed once
$y_{ijk} \in \{0,1\},$	$\forall i,j,k$	
$f_{jk} \ge 0,$	$\forall j,k$	

Flow shops

- A flow shop is a serial production system where the process flow of all products (orders) is similar
- Queuing is allowed (unlike in lines) and jobs can pass each other, i.e. processing order of jobs does not need to be the same at all processing steps
- If jobs are processed in the same order at each step, we have a permutation schedule (there are N! permutation schedules for N orders)
- Optimal schedule is not necessarily a permutation schedule, but usually the best permutation schedule is not far from optimum.
- Flow shop is a common and efficient arrangement for production of product families with a lot of product variability



Two machine flow shop – Johnson's algorithm

- Johnson's algorithm minimizes makespan of a two machine flow shop as follows:
- Job *i* precedes *j* in schedule if

  min {P_{i1}, P_{j2}} <= min {P_{i2}, P_{j1}}
- Procedure:

1. Find min *i* {P_{i1}, P_{i2}}
2. If smallest value is on machine 1, schedule is filled from beginning
3. If smallest value is on machine 2, schedule is filled from end
4. Return to step 1 until all jobs are scheduled

Idea: Because first machine is running all the time anyway, the schedule is extended only if the second machine waits. By scheduling shorter job first on the first machine, this waiting is minimized.

Two machine flow shop – Johnson's algorithm

Example:

Job *i*	1	2	3	4	5
P_{i1}	3	5	1	6	7
P_{i2}	6	2	2	6	5

Step	Jobs left	Position	Schedule
1	1,2,3,4,5	3 = [1]	3 X X X X
2	1,2,4,5	2 = [5]	3 X X X 2
3	1,4,5	1 = [2]	3 1 X X 2
4	4,5	5 = [4]	3 1 X 5 2
5	4	4 = [3]	3 1 4 5 2

Flow shop optimization

Flow shops with more than 2 machines can not be optimized with simple rules

Optimization model is obtained from Manne's single machine model by:

- Adding indexes for the machines
- Determining order of machines in the process

In this model tardiness is only counted for the last step.

IP model, $k,..,K$ is machine index

$\text{Min} \sum_{\forall i} f_i$	
$t_{iK} + P_{iK} - D_i \le f_i,$	$\forall i$
$t_{ik} \ge 0,\ f_i \ge 0,$	$\forall i,k$
$t_{ik} + P_{ik} \le t_{ik+1},$	$\forall i,k \in \{1,..,K-1\}$
$My_{ijk} + (t_{ik} - t_{jk}) \ge P_{jk},$	$\forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\}, k$
$M(1-y_{ijk}) + (t_{jk} - t_{ik}) \ge P_{ik},$	$\forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\}, k$
$y_{ijk} \in \{0,1\},$	$\forall i,j,k$

Flow shop – permutation model

Optimization model is obtained easily by adding a constraint to the previous model that forces values of y_{ijk} to be equal $\forall k$

or

From Wagner's single machine variation as follows:

- Indexes are added to machines
- Job order on machines and between machines so that there is no overlap

$$\text{Min} \sum_{\forall j}\left(t_{jK} + \sum_{\forall i} P_{iK} y_{ij}\right)$$

$$t_{jk} \geq 0, y_{ij} \in \{0,1\}, \qquad \forall ijk$$

$$\sum_i y_{ij} = 1, \qquad \forall j$$

$$\sum_j y_{ij} = 1, \qquad \forall i$$

$$t_{jk} + \sum_{\forall i} P_{ik} y_{ij} \leq t_{j+1,k}, \qquad \forall j \in \{1..,I-1\}, k$$

$$t_{jk} + \sum_{\forall i} P_{ik} y_{ij} \leq t_{j,k+1}, \qquad \forall j,k \in \{1..,K-1\}$$

Sum of flow times (all jobs available at $t = 0$)

Job order on machines

Job order between machines

Flow shop example

We have five jobs and a four machine flow shop

Example data:

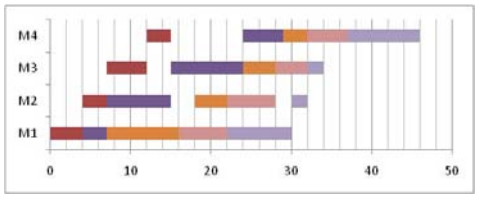
Job *i*	P_{i1}	P_{i2}	P_{i3}	P_{i4}	D_i
1	4	3	5	3	17
2	6	6	4	5	35
3	8	2	2	9	46
4	3	8	9	5	25
5	9	4	4	3	23

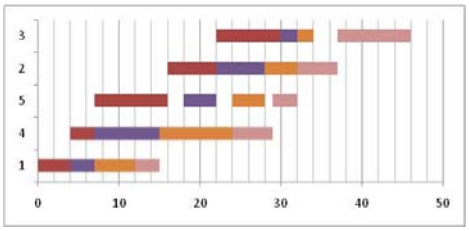
Various optimization criteria are relevant:

- Total taridness
- Max tardiness
- Makespan
- Total throughput time
- Utilization rate

Below permutation schedule for minimization of total tardiness

Note! Colors are not equal in the two views

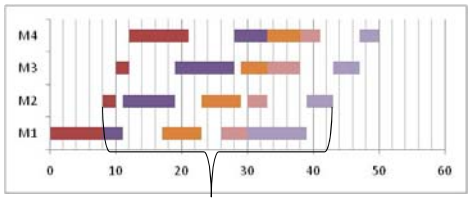




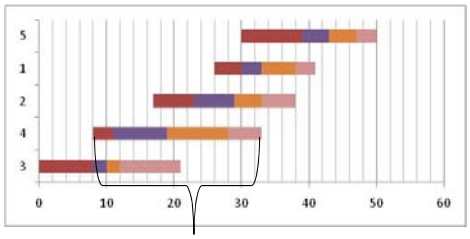
Flow shop example (permutation schedules)

Combined Utilization and throughput time minimization

Weights: Throughput time 10; Utilization 0

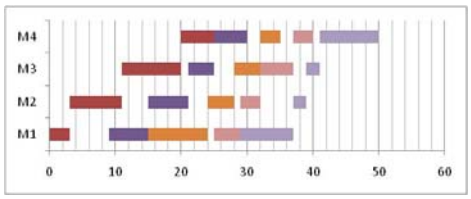


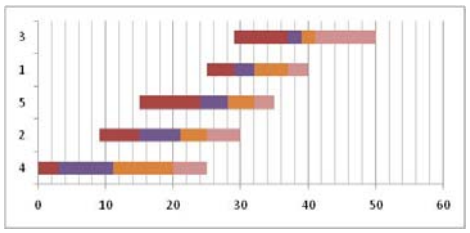
Utilization



Throughput time

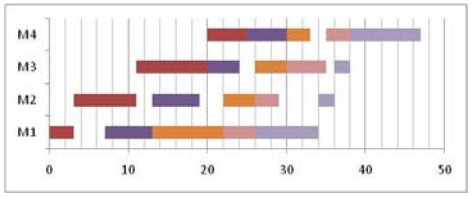
Weights: Throughput time 9; Utilization 1

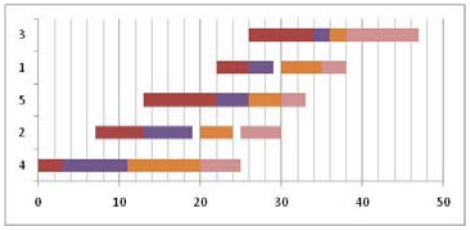




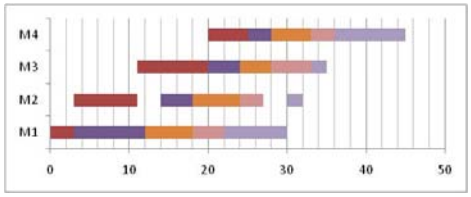
Flow shop - example

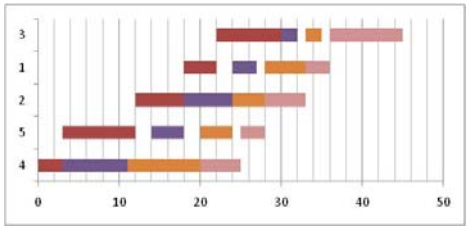
Weights: Throughput time 8; Utilization 2





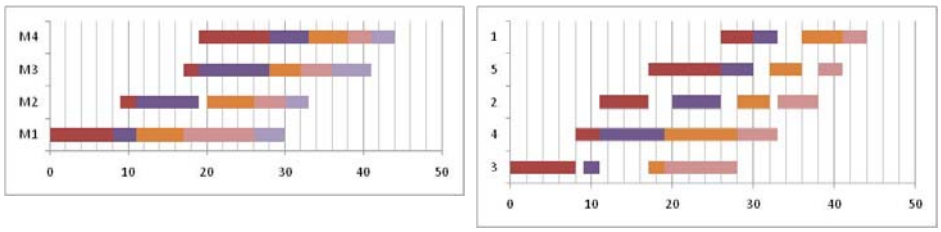
Weights: Throughput time 5; Utilization 5



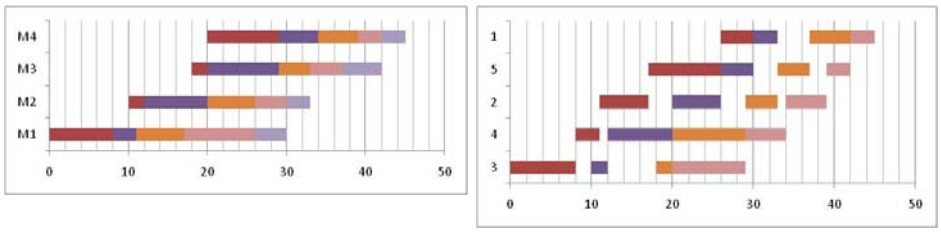


Flow shop - example

Weights: Throughput time 2; Utilization 8

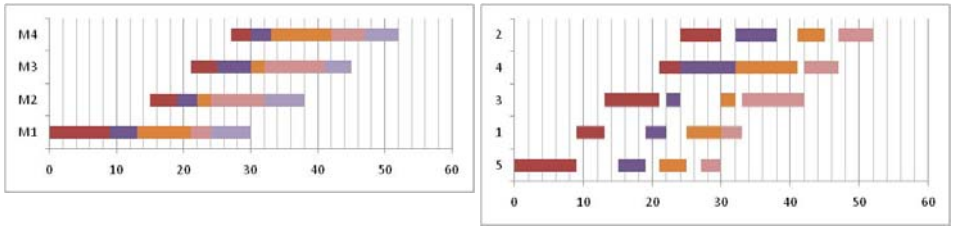


Weights: Throughput time 1; Utilization 9



Flow shop - example

Weights: Throughput time 0; Utilization 10

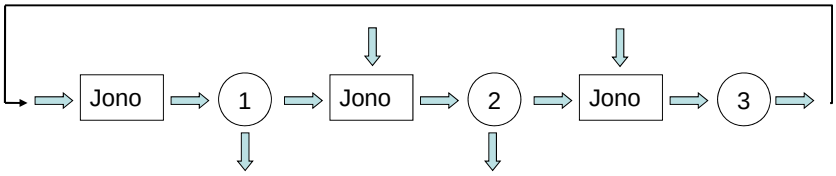


Open Flow shop and re-entrant Flow shop

- In an open flow shop processing may start from any step and stop at any step – flowing order stays the same
- In a re-entrant Flow shop orders may return to the same machines

Optimization model is the same as for other flow shops, but:

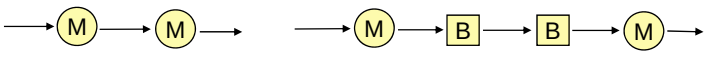
- Processing times are set to zero before the first processing step and after the last processing step
- Re-entering jobs are handled like new orders and a precedence constraint is defined between orders. This of course works for a single machine, too.



Production line

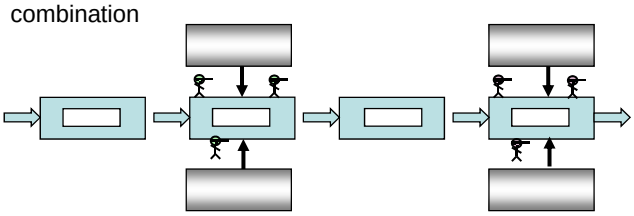
A production line consists of serial work stations between which there is no queuing or only a limited buffer

- Work is allocated to the stations
- Work at the stations is standardized, specialized, and efficient
- Learning is fast
- Resource utilization is high – only one set of equipment of a type is required
- Materials and tools can be located within reach of the worker
- Material delivery to one point only
- Job transfer between stations is required
- Vulnerable to disturbances



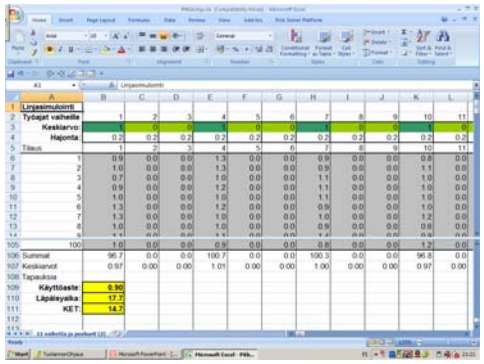
Synchronous and asynchronous lines

- In ***synchronous lines*** jobs move at the same time
  - The main objective is to balance work content between stages
- In ***asynchronous lines*** workpieces may move forward at different times
  - Allows processing time variation compensation, especially if buffers are used
  - Downside is longer throughput time and WIP and longer line if buffers are used
  - Line length limits use of buffers with large products
  - Line pressure is lower than with synchronous lines and conveyors
  - Undermanned asynchronous line and moving workers is a popular combination



Asynchronous line simulation using Excel

- Asynchronous line is mainly evaluated according to line utilization, in practice high output or low waiting are desirable
- Waiting occurs if line is blocked or job in previous stage is not finished (starving)
- Throughput time is not a very relevant criterion, as not much can be done about it after the line is built

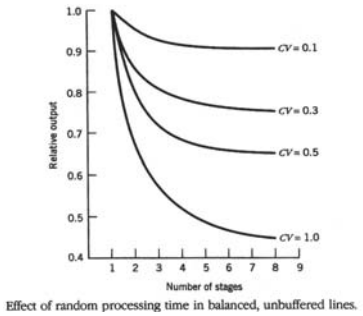


”LineSimulator.xls”

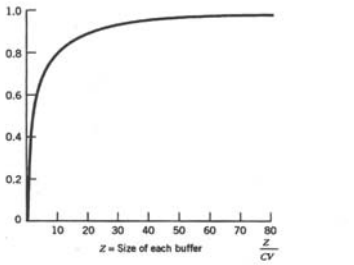
Esko Niemi

Asynchronous line simulation

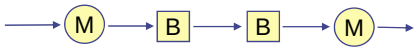
- If stages are equal, capacity of line depends on line length, processing time variation and buffer sizes
- Here cv = processing time standard deviation / mean processing time
- Stages are in balance on average
- Jobs are always available in the beginning of the line



Effect of random processing time in balanced, unbuffered lines.



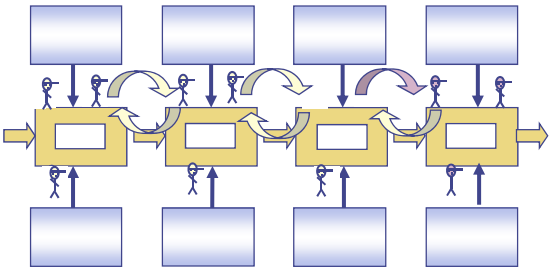
Proportion of lost output recovered by buffering in balanced lines.



Esko Niemi

Asynchronous line – moving workers

- If products are large, buffers may be out of question
- Processing time variation can be balanced by moving workers
- This requires definition of some roles for workers or active management
- Causes losses like congestion due to non-optimal number of workers
- Differences in workers may be utilized, for example: only the best move

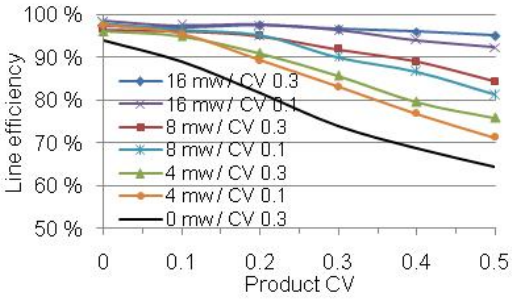


Esko Niemi

Asynchronous line – moving workers

Simulation results for 4 station line:

- About 80 % (90 %) efficiency can be reached when product variation (processing time) CV is 0.3 (0.2)
- In these results a significant congestion loss is involved
- The share of moving workers is 16 – 33 %
- One can assume, that the specialization, logistics and other benefits are greater than losses



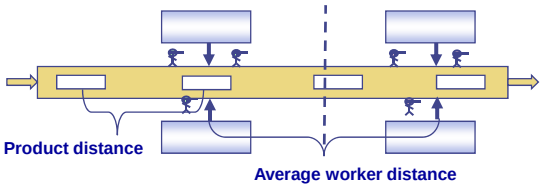
Example

- Totally 48 workers and no congestion loss
- mw = moving workers
- CV 0.1 and 0.3 indicate worker productivity differences

Esko Niemi

Conveyor line – large products

- Line is constantly moving
- Product transfer does not require any resources
- Either workers or floor must move with product
- A buffer effect is created by locating products on the line tighter than workers
- Product sequence rules can be used to limit required buffer size

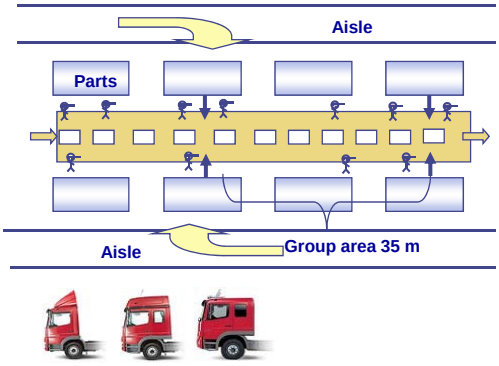


Esko Niemi

Truck cabin assembly conveyor line

- Conveyor speed ca. 1.5 m/min, cabin distance 5 m => cycle time ca. 3.3 min
- Group area 35 m => about 7 cabins in the area
- Large product variation
- Balancing with product mix, too
- Only some main parts are sequence controlled, others standard from stock

- Line speed always the same, capacity is adjusted with flexible working time:
- 2 shifts +/- 300 h/y
- Line length 450 m
- Throughput time 5 h



Esko Niemi

Asynchronous line optimization

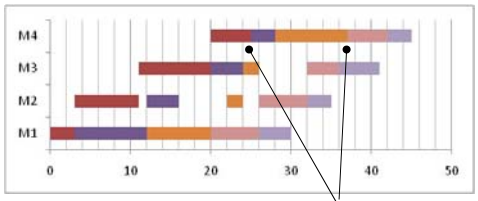
- Relevant decision variable is job order
- Previous flow shop models are appended with a constraint that prevents starting of processing of a job before the processing of the previous job on the next station has started – i.e. the next machine (or buffer position) must be empty before product can move
- Buffer positions are modeled as machines with 0 processing time

- **IP optimization model**

$\text{Min } t_{IK} + \sum_{\forall i} P_{iK} y_{il}$		Completion of last job
$t_{jk} \geq 0, y_{ij} \in \{0,1\},$	$\forall ijk$	
$\sum_i y_{ij} = 1,$	$\forall j$	
$\sum_j y_{ij} = 1,$	$\forall i$	
$t_{jk} + \sum_{\forall i} P_{ik} y_{ij} \leq t_{j+1,k},$	$\forall j \in \{1..,I-1\}, k$	Job order on machines
$t_{jk} + \sum_{\forall i} P_{ik} y_{ij} \leq t_{j,k+1},$	$\forall j,k \in \{1..,K-1\}$	Job order between machines
$t_{j+1,k} \geq t_{j,k+1},$	$\forall j \in \{1..,I-1\}, k \in \{1..,K-1\}$	Line condition

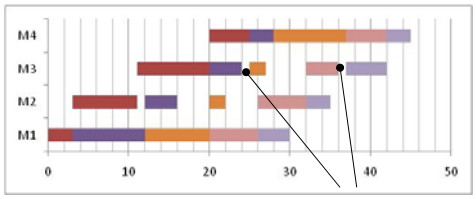
Line example

- Same case as a flow shop and (no-buffer) line
- Makespan is optimized
- Here job order is the same (permutation schedules)



Flow shop

Waiting – next job on machine 3 can start immediately

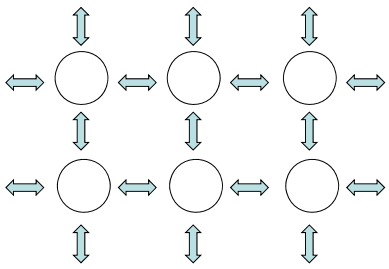


Line

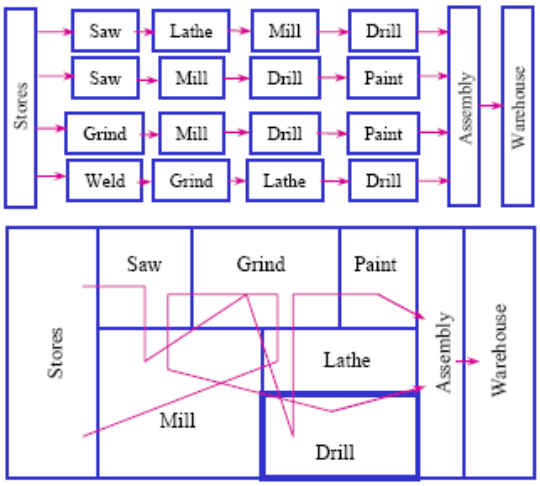
Blocked – job waits on machine 3 (gap in schedule), until machine 4 is freed and next job can start

Job shop

- In a job shop job routing is not costrained in any way and any recipe can be followed
- Typical properties:
  - Product variation is large
  - Process and processing time variation are large
  - Process stages number and order vary
  - Processes are flexible
  - Value adding time in relation to throughput time is small, because of the variation and set-up times (-> batches)



Flow shop vs. Job shop



Flow shop
- Similar processes for product families
- Short transportation distances
- Visual control
- Short throughput times

Job shop
- Control is based on queuing priorities
- Difficult to balance
- Long transportation times
- Long throughput times
- Utilization rate of skilled labor is high

Kuvat Hopp et al. / Factory physics

Job shop

Optimization model is obtained from the flow shop model with minor changes:

- Job's processing order is determined with recipe specific constraints
- Set *O* represents processing steps (machines) *o* and *p* of all jobs *i* that are consecutive in the chain
- If job *i* does not use machine *k*, set $t_{ik} = 0$
- In the example total tardiness is minimized and in order to keep notation simple, all steps are examined for tardiness and not just he last one

$$\text{Min} \sum_{\forall i} f_i$$

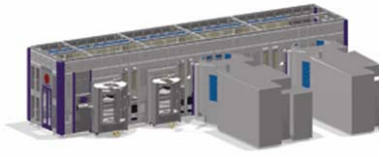
$t_{ik} + P_{ik} - D_i \le f_i,$	$\forall i,k$
$t_{ik} \ge 0,\ f_i \ge 0,$	$\forall i,k$
$t_{io} + P_{io} \le t_{ip},$	$\forall io,ip \in O$
$My_{ijk} + (t_{ik} - t_{jk}) \ge P_{jk},$	$\forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\},k$
$M(1 - y_{ijk}) + (t_{jk} - t_{ik}) \ge P_{ik},$	$\forall i \in \{1,..,I-1\}, j \in \{i+1,..,I\},k$
$y_{ijk} \in \{0,1\},$	$\forall i,j,k$

Prioritizing

- Because production control is difficult, scheduling in practice usually resorts to queue prioritizing
- This means choosing the next job from those available with a ”rule of thumb”
- Typical rules:
  - SPT Shortest Processing Time
  - EDD Earlies Due Date
  - FIFO (FCFS) First In (Come) First Out (Served)
  - S/RO Slack per Remaining Operation
  - Covert Cost of delay over remaining processing time
  - CR Critical Ratio – Slack per Total Work Remaining or machine loading rate
  - LTWK Least Total Work
  - LWKR Least Work Remaining
  - RANDOM
  - WINQ Work in Next Queue

FM systems

- FM systems (FMS, Flexible Manufacturing System) are automatic centrally controlled machining systems that consist typically of NC machine tools and supporting machines like washing machines etc., central workpiece changing and storing system (AS/RS, Automatic Storage / Retrieval System) and loading and unloading stations
- The purpose is to increase utilization rate and weekly running time of machines with automation
- Throughput time is secondary objective in the sense that long daily running time guarantees a high turnover rate of orders anyway
- Use of flexible technology makes manufacturing a large variety of products possible
- Set-ups are mostly external and therefore small batch sizes or one-off manufacturing is possible



FM systems' properties

From the production control point of view FM systems are job shops with the following features

- Alternative parallel machines, but
- Similarity of machines is affected by tool set-up
- AS/RS is a scheduled or at least loaded resource in the system
- Number of pallets is limited
- Set-ups are external and they mainly consist of change of fixtures on the pallets
- Change of fixtures takes considerable time
- Workpiece change and fixture change are done manually during manned shifts

FM system control constraints

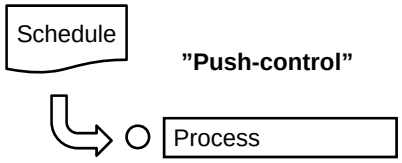
- Tool set-up must allow execution of manufacturing program
- Fixture set-up must allow execution of manufacturing program
- There should be enough work for the unmanned shifts
- Fixture and workpiece change must be executed during the manned time
- AS/RS system must not become too limiting a bottleneck
- In addition, normal manufacturing targets must be met: required parts must be manufactured within the given time

FM system planning and control

- The products to be manufactured are selected based on their relative fit and economy
- In the medium term the products are determined in the master production schedule
- In the short term the job selection can be formulated as an optimization problem with the following constraints
  - Tool set-up is such that the jobs can be executed
  - There is sufficient work (loaded pallets) for the unmanned shifts
  - Workpiece loading and fixture changes can be done during the manned shifts
- System control is based on prioritizing, where the AS/RS system serves machines based on priorities set to jobs and machines
- Operator selects jobs to be loaded on pallets from the job queue and calls for pallet and material to the loading station
- Unloading has its own job queue

CONWIP (Constant WIP) control

- Classical aggregate planning pushes orders to the system without consideration of the system status

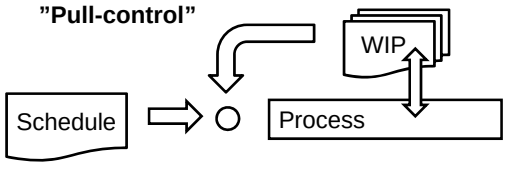


”Push-control”

- Jobs are easily congested due to poor planning and disturbances

- In Conwip systems jobs are released only if system status (WIP) allows it

”Pull-control”



CONWIP systems

- Work in process is controlled with the number of Kanban cards or similar book keeping in the EDP system
- Same effect takes place in line production, where buffers limit the amount of WIP

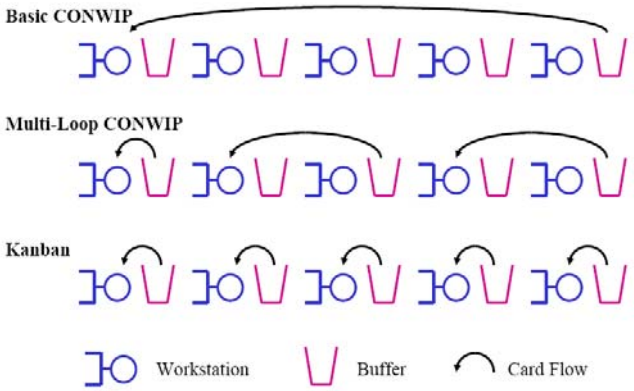


Figure: Hopp et al. / Factory physics

Bottleneck control

- Because production bottleneck constrains the production flow of the whole system, it requires special attention
- Bottleneck can be handled as a single machine, which simplifies control
- More waiting and large batch sizes are allowed in the bottleneck than elsewhere in the production process
- In practice the bottleneck tends to travel, and the situation is more complicated

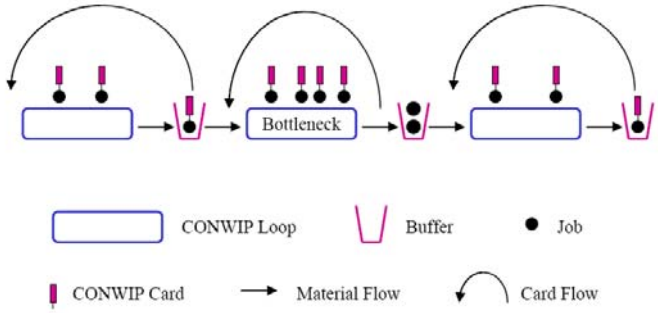


Figure Hopp et al. / Factory physics

Medium term planning

- Capacity planning
- Aggregate planning
  - Stock replenishment
  - Make-to-stock, batch production
  - Make-to-order

Capacity planning

- In planning of production any further than immediate near future one has to use (inaccurate) estimates
- Unit of workload in planning is usually working hour or average product
- Timing resolution is rough, usually day, week or month
- The objective is to match predicted work load with capacity at minimum cost
- In modeling and reality decision variables concerning capacity are, depending on local circumstances
  - Work force adjustment
    - Hiring and firing
    - Overtime
    - Personnel leasing
    - Subcontracting
  - Manufacturing to stock
- All measures involve cost aspects and limitations on quantity

Capacity planning

- In its basic form Capacity planning is easy to formulate as an optimization model
- Time proceeds in discrete steps determined by detail level of planning
- Here we manufacture average products and examine the whole factory as a single resource
- Notation, parameters:
  - t = time period index, $t = 1,..,T$
  - D_t = demand during period t
  - B = production (products) per worker on a time period
  - C = cost of one worker for a time period
  - O = overtime cost of one worker for a full time period
  - P = Hiring cost of a worker
  - E = Firing cost of a worker
  - H = inventory holding cost of a product for a time period

Capacity planning

- Decision variables:
  - q_t = production amount on time period t
  - o_t = overtime done in worker input on time period t
  - p_t = workers hired in the beginning of time period t
  - e_t = workers fired in the beginning of period t
  - W_t = Number of workers on period t; intermediate result, not a real decision variable
  - I_t = stock level on period t; intermediate result
- Optimization model:

$$\text{Min } c = \sum_{\forall t} \left(CW_t + Oo_t + Pp_t + Ee_t + HI_t \right)$$

$s.t.$

$$W_t = W_{t-1} + p_t - e_t, \quad \forall t$$

$$I_t = I_{t-1} + q_t - D_t, \quad \forall t$$

$$q_t \le (W_t + o_t)B, \quad \forall t$$

$$q_t, o_t, p_t, e_t, I_t \ge 0, \quad \forall t$$

Capacity planning – Excel-model

- Additional constraints on overtime, capacity, ending inventory are easy to add as well as different products, worker skills etc.

Production plan - overtime, firing & hiring and stocks used

	January	February	March	April	May	June	
Demand, *D*	2760	3320	3970	3540	3180	2900	
Overtime, *o*	0.0	0.0	0.0	2.4	0.0	0.0	
Workers hired, *p*	6.9	0.0	0.0	0.0	0.0	0.0	
Workers laid off, *e*	0.0	0.0	0.0	0.0	2.1	0.0	
Workers available, *W*	41.9	41.9	41.9	41.9	39.8	39.8	
Hiring cost	17875	0	0	0	0	0	
Lay-off cost	0	0	0	0	5525	0	
Labor cost	100500	100500	100500	100500	95400	95400	
Overtime labor cost	0	0	0	9975	0	0	
Capacity	3350	3350	3350	3540	3180	3180	
Units produced, *q*	3350	3350	3350	3540	3180	2900	
Net inventory, *I*	590	620	0	0	0	0	
Holding cost	2950	3100	0	0	0	0	
Total cost	121325	103600	100500	110475	100925	95400	
Grand total							632225

Parameters	
Wage/period, *C*	2400
Overtime/period, *O*	4200
Production/worker/period, *B*	80
Hiring cost/worker, *P*	2600
Firing cost/worker, *E*	2600
Holding cost/unit/period, *H*	5
Initial workers	35.0

Color code:
Variable
Parameter
Objective
Costs
Intermediate result

Capacity planning – subcontracting with quantity discount

- New parameters
  - *S1* = subcontracting cost below quantity discount limit
  - *S2* = subcontracting cost above quantity discount limit, *S2* < *S1*
  - *S1U* = lower discount limit
  - *S2U* = upper discount limit
- New variables:
  - $s1_t$ = subcontracting up to *S1U* on period *t* in worker inputs
  - $s2_t$ = subcontracting from *S1U* on period *t* in worker inputs
  - z_t = binary auxiliary variable indicating production above *S1U*
- In the model subcontracting is taken into account with its quantity dependent cost
- Because price reduces with increased quantity, we must use auxiliary variables to force the more expensive subcontracting to be used first

Capacity planning – subcontracting with quantity discount

$$\text{Min}\, c = \sum_{\forall t} \left(CW_t + Oo_t + S1s1_t + S2s2_t + Pp_t + Ee_t + HI_t \right)$$

s.t.

$$W_t = W_{t-1} + p_t - e_t, \qquad \forall t$$

$$I_t = I_{t-1} + q_t - D_t, \qquad \forall t$$

$$q_t \le \left(W_t + o_t + s1_t + s2_t \right) B, \qquad \forall t$$

$$s1_t \ge z_t S1U \qquad \forall t$$

$$s1_t \le S1U \qquad \forall t$$

$$s2_t \le z_t M \qquad \forall t$$

$$s2_t \le S2U \qquad \forall t$$

$$z_t \in \{1,0\} \qquad \forall t$$

$$q_t, o_t, p_t, e_t, I_t \ge 0, \qquad \forall t$$

Capacity planning – subcontracting with quantity discount

Production plan - overtime, firing & hiring, stocks and subcontracting used

	January	February	March	April	May	June	
Demand	2760	3320	3970	3540	3180	2900	
Overtime	0.0	0.0	0.9	0.0	0.0	0.0	
Subcontracting 1, *s1*	0.0	3.0	3.0	3.0	3.0	0.0	
Subcontracting 2, *s2*	0.0	5.0	5.0	5.0	0.5	0.0	
Workers hired	1.3	0.0	0.0	0.0	0.0	0.0	
Workers laid off	0.0	0.0	0.0	0.0	0.0	0.0	
Workers available	36.3	36.3	36.3	36.3	36.3	36.3	
Hiring cost	3250	0	0	0	0	0	
Lay-off cost	0	0	0	0	0	0	
Labor cost	79750	79750	79750	79750	79750	79750	
Overtime labor cost	0	0	2800	0	0	0	
Cost of Subcontracting 1	0	8400	8400	8400	8400	0	
Cost of Subcontracting 2	0	12500	12500	12500	1250	0	
Capacity	2900	3540	3610	3540	3180	2900	
Units produced	2900	3540	3610	3540	3180	2900	
Net inventory	140	360	0	0	0	0	
Holding cost	700	1800	0	0	0	0	
Total cost	83700	102450	103450	100650	89400	79750	
Grand total							559400
Parameters							
Wage/period	2200	Cost S1	2800		**Color code:**		
Overtime/period	3200	Cost S2	2500		Variable		
Production/worker/period	80	Max S1, *S1U*	3		Parameter		
Hiring cost/worker	2600	Max S2, *S2U*	5		Objective		
Firing cost/worker	2600	Big M	1000		Costs		
Holding cost/unit/period	5	Initial workers	35.0		Intermediate result		
Subcontracting 2, *z*	0	1	1	1	1	0	
Cond. Sub. 1 limit	0	3	3	3	3	0	
Cond. Sub. 2 limit	0	1000	1000	1000	1000	0	

Stock replenishment

- Stocks are used to store parts or products to wait for need by production or purchase by customers
- The advantage is immediate availability. Downside is holding cost of inventory.
- Stocking is possible (and obligatory) for standard items, the demand of which is somewhat continuous
- Stock replenishment is done in batches for economical reasons due to one-time costs
- Replenishment can take place as purchasing orders or manufacturing lots

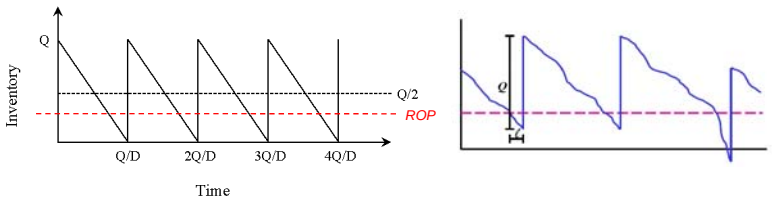
Stock replenishment – even demand

- In stock replenishment batch sizes have to be determined
- One must consider one-time (set-up or ordering) costs and holding costs
- For even demand ”economic order quantity” and replenishment schedule can be calculated starting from total cost:

$$C_{tot} = DC + (D/Q)S + (Q/2)H$$

where

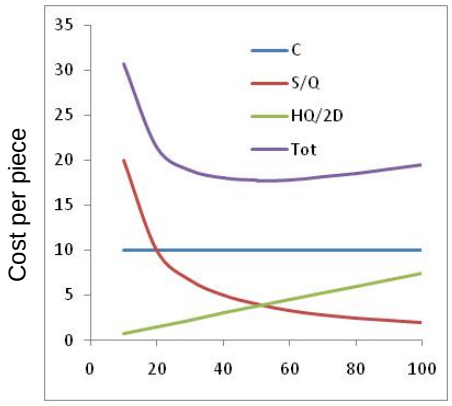
D = demand, C = variable cost, Q = batch size, S = one-time cost, H = holding cost



Stock replenishment – even demand

- Taking derivative with respect to *Q* and setting it equal to zero we solve optimal *Q*:

$$Q_{eoq} = \sqrt{2DS/H}$$



- Formula does not consider capacity, and it applies better to replenishing stock by purchasing
- Consumption during delivery is anticipated by setting reorder point
- As in the figure, result usually is not very sensitive to batch size

Stock replenishment – uncertain demand and delivery time

- In reality demand (consumption) varies and delivery time varies too
- Therefore *ROP* (= reorder point) must be set so that it is sufficient with reasonable certainty

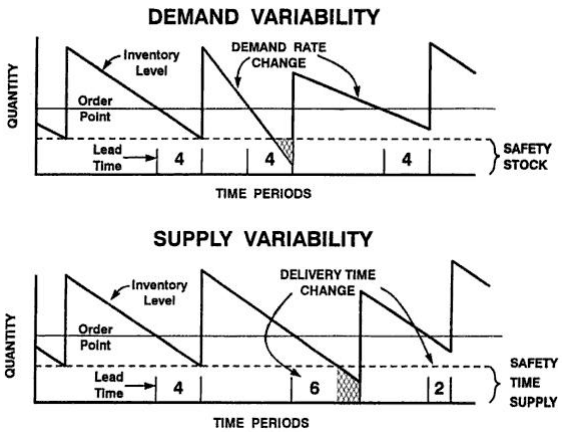


Figure: Greene, APICS

Stock replenishment – uncertain demand and delivery time

- *ROP* setting is based on demand and delivery time and their variations
- (Standard) deviation is multiplied by safety factor 2...3
- Two independent variances may be summed, and thus:

$$ROP = \mu_D \mu_{LT} + Z\sqrt{\mu_D \sigma_{LT}^2 + \sigma_{LDT}^2}$$

where

μ_D = average demand per time unit

μ_{LT} = average delivery time

σ_{LT} = standard deviation of delivery time

σ_{LDT} = standard deviation of demand during average delivery time

Z = safety factor from normal distribution

- Shape of distribution affects the result and one way to find *ROP* is to use simulation with historical demand data

Stock replenishment – varying but known demand

- If demand can be forecast accurately, a ”dynamic lot-sizing” model can be formulated
- A safety stock can easily be added if there is some uncertainty in demand
- Mathematical formulation using familiar notation:

q_t = replenishment quantity during t

I_t = inventory after t (t = 1,..,T)

D_t = demand during t

H = holding cost for one unit (product) and one time unit

S = set-up cost or other one-time cost

M = large number

s_t indicates one-time cost during t (if $q_t > 0$, $s_t = 1$, and if $q_t = 0$, $s_t = 0$)

Dynamic lot-sizing – varying but known demand

- Economical batch sizes are solved by minimizing

$$\sum_{t=1}^{T}(S s_t + H I_t),$$

which minimizes sum of one-time costs and holding costs over all t with the following constraints:

$$I_{t-1} + q_t - I_t = D_t$$
$$q_t - s_t M \leq 0$$
$$q_t \geq 0$$
$$I_t \geq 0$$
$$s_t \in \{0,1\}$$

- First constraint assures that demand is met
- Second sets s = 1, if replenishment takes place
- Rest are positivity and binary constraints

Dynamic lot-sizing – Excel model

- Basic model, one-time cost takes place when products are ordered or manufactured
- Resources and capacity are not considered
- It is assumed that replenishment takes place immediately and cost is constant per product
- Cost/piece has no effect on optimization

Replenishment plan

Period, t	1	2	3	4	5	6	7	8	9	10
Demand, D_t	108	90	73	100	48	49	69	99	140	95
Replenishment, q_t	108	163	0	147	0	117	0	99	235	0
One-time cost, s_t	1	1	0	1	0	1	0	1	1	0
One-time cost, Ss_t	100	100	0	100	0	100	0	100	100	0
Net inventory, I_t	0	73	0	48	0	69	0	0	95	0
Holding cost, $I_t H$	0	73	0	48	0	69	0	0	95	0
Total cost	100	173	0	148	0	169	0	100	195	0
One-time constraint aux. variab	999	999	0	999	0	999	0	999	999	0
Parameters										884
Holding cost/unit/period, H	1									
Big M	999									
One-time cost, S	100									

Replenishment in standard batches

- By changing M to transportation container size and s_t to integers we obtain a model for optimization of deliveries in standard batches
- Demand is met by multiples of M and rest is stored:

Repelnishment plan - containers

Period, t	1	2	3	4	5	6	7	8	9	10
Demand, D_t	108	90	73	100	48	49	69	99	140	95
Replenishment, q_t	148	50	88	100	50	50	50	99	140	95
Containers, s_t	3	1	2	2	1	1	1	2	3	2
Total container cost, Ss_t	300	100	200	200	100	100	100	200	300	200
Net inventory, I_t	40	0	15	15	17	19	0	0	0	0
Holding cost, $I_t H$	40	0	15	15	17	19	0	0	0	0
Total cost	340	100	215	215	117	119	100	200	300	200
Total container capacity	150	50	100	100	50	50	50	100	150	100
Parameters										1906
Holding cost/unit/period, H	1									
Container capacity	50									
Container cost, S	100									

Make-to-stock production - MTS

- Make-to-stock (MTS) production is used to manufacture parts or products in batches and store them in stock to wait for need
- An example could be a screw manufacturer, that makes screws to stock and delivers them to customers when orders arrive
- The previous dynamic lot sizing formulations are appended with capacity constraints
- We add more products and later more departments to the system
- One-time costs are usually related to set-ups in manufacturing
- In MTS production problems related to timing are usually easier than in E/MTO production
- This is because there often is no immediate need for the product and scheduling is more flexible

MTS – model

- We combine capacity constraints, batch sizing and timing
- Capacity optimization described earlier could easily be integrated to the models, but is left out for clarity
- Several products *i*, but only one resource
- Set-up takes place if product is manufactured – in aggregate planning we assume that several products are manufactured during the time period and set-ups have to be changed
- Model does not consider job order within time period

$$\text{Min } c = \sum_{\forall t}\sum_{i}\left(S_i s_{it} + H_i I_{it}\right)$$ Set-up cost and holding cost

s.t.

$$I_{it} = I_{i,t-1} + q_{it} - D_{it}, \quad \forall i,t$$ Inventory balance

$$\sum_{\forall i} Q_i q_{it} + St_i s_{it} \le W_t, \forall t$$ Work content (Q_i for i) and set-up times (St_i for i) less than capacity

$$q_{it} - s_{it} M \le 0, \quad \forall i,t$$ Auxiliary constraints for set-ups

$$q_{it}, s_{it}, I_{it} \ge 0, \quad \forall i,t$$

$$s_{it} \in \{0,1\}$$

MTS – Excel model

- Several products (3), one resource

Period	1	2	3	4	5	6	7	8	9	10
Demand 1	20	50	10	50	50	10	20	40	20	30
Demand 2	10	50	25	50	10	50	50	10	20	50
Demand 3	50	50	10	50	25	50	50	50	10	20
Production 1, units	70	0	10	50	60	0	20	60	0	30
Production 2, units	10	50	60	25	0	70	40	0	20	50
Production 3, units	50	60	0	50	25	50	50	60	0	20
Set-up 1 - aux.v.	1	0	1	1	1	0	1	1	0	1
Set-up 2 - aux.v.	1	1	1	1	0	1	1	0	1	1
Set-up 3 - aux.v.	1	1	0	1	1	1	1	1	0	1
Set-up time	180	140	100	180	120	140	180	120	60	180
Processing time	240	280	130	250	135	290	250	240	40	190
Total time	420	420	230	430	255	430	430	360	100	370
Net inventory 1	50	0	0	0	10	0	0	20	0	0
Net inventory 2	0	0	35	10	0	20	10	0	0	0
Net inventory 3	0	10	0	0	0	0	0	10	0	0
Total (holding) cost	50	30	70	20	10	40	20	50	0	0
										290

Parameters

Processing time/unit 1	1	Set-up time 1	40	Big M	999	
Processing time/unit 2	2	Set-up time 2	60	Capacity	430	
Processing time/unit 3	3	Set-up time 3	80			
Holding cost/unit/period 1	1					
Holding cost/unit/period 2	2					
Holding cost/unit/period 3	3					

Set-up constraint	999	0	999	999	999	0	999	999	0	999
	999	999	999	999	0	999	999	0	999	999
	999	999	0	999	999	999	999	999	0	999

MTS – total process

- Here stage (department) *k*-1 feeds stage *k*, which is taken into account in the inventory balance constraints
- Any process can be modeled in this manner

$$\text{Min}\, c = \sum_{\forall t}\sum_{i}\sum_{k}\left(S_{ik}s_{ikt} + H_i I_{ikt}\right)$$

K is last stage, where end demand takes place

s.t.

$$I_{i,k-1,t} = I_{i,k-1,t-1} + q_{i,k-1,t} - q_{ikt}, \quad \forall i,t,k \in \{1,..,K-1\}$$

$$I_{iKt} = I_{iK,t-1} + q_{i,K,t} - D_{it}, \qquad \forall i,t$$

$$\sum_{\forall i} Q_{ik}q_{ikt} + St_{ik}s_{ikt} \le W_{kt}, \qquad \forall k,t$$

$$q_{ikt} - s_{ikt}M \le 0, \qquad \forall i,k,t$$

$$q_{ikt}, s_{ikt}, I_{ikt} \ge 0, \qquad \forall i,k,t$$

$$s_{ikt} \in \{0,1\}$$

MTS – total process Excel model

- 2 products, 2 stage process

	Period	1	2	3	4	5	6	7	8	9	10
Product 1	Demand 1	20	50	10	50	50	10	20	40	20	30
Product 2	Demand 2		50	25	50	10	50	50	10	20	50
Stage 2	Production 1, units	70	0	60	0	80	0	0	60	0	30
	Production 2, units	0	60	15	60	0	50	65	0	65	0
Stage 1	Production 1, units	130	0	0	0	80	0	0	90	0	0
	Production 2, units	0	65	65	55	0	65	65	0	0	0
Stage 2	Production 1	1	0	1	0	1	0	0	1	0	1
	Production 2	0	1	1	1	0	1	1	0	1	0
Stage 1	Production 1	1	0	0	0	1	0	0	1	0	0
	Production 2	0	1	1	1	0	1	1	0	0	0
Stage 2	Set-up time	40	60	100	60	40	60	60	40	60	40
	Processing time	70	120	90	120	80	100	130	60	130	30
	Total time	110	180	190	180	120	160	190	100	190	70
Stage 1	Set-up time	100	100	100	100	100	100	100	100	0	0
	Processing time	130	130	130	110	80	130	130	90	0	0
	Total time	230	230	230	210	180	230	230	190	0	0
Stage 2	Net inventory 1	50	0	50	0	30	20	0	20	0	0
	Net inventory 2	0	10	0	10	0	0	15	5	50	0
Stage 1	Net inventory 1	60	60	0	0	0	0	0	30	30	0
	Net inventory 2	0	5	55	50	50	65	65	65	0	0
	Holding cost	50	10	50	10	30	20	15	25	50	0
	Total cost	190	170	250	170	170	180	175	165	110	40
											1620

	Parameters										
Stage 2	Holding cost/unit/period 1	1	Processing time/unit 1			1	Set-up time 1		40	Big M	999
	Holding cost/unit/period 2	1	Processing time/unit 2			2	Set-up time 2		60	Capacity	190
Stage 1	Holding cost/unit/period 1	0	Processing time/unit 1			1	Set-up time 1		100	Capacity	230
	Holding cost/unit/period 2	0	Processing time/unit 2			2	Set-up time 2		100		
Stage 2	Aux production 1	999	0	999	0	999	0	0	999	0	999
	Aux production 2	0	999	999	999	0	999	999	0	999	0
Stage 1	Aux production 1	999	0	0	0	999	0	0	999	0	0
	Aux production 2	0	999	999	999	0	999	999	0	0	0

MTS – only one product at a time

- Set-up takes place only if product type changes
- Only one product type can be processed concurrently – in this sense this is not aggregate production but short-run control
- In this model only one resource

$$\text{Min } c = \sum_{\forall t}\sum_{i}\left(S_i z_{it} + H_i I_{it}\right)$$ Set-up cost and holding cost

$s.t.$

$I_{it} = I_{i,t-1} + q_{it} - D_{it},$	$\forall i,t$	Inventory balance
$\sum_{\forall i}\left(Q_i q_{it} + St_i z_{it}\right) \le W_t,$	$\forall t$	Work content and set-up times less than capacity
$q_{it} \le s_{it} M,$	$\forall i,t$	Production indicator
$s_{it} - s_{i,t-1} \le z_{it} M$	$\forall i,t$	Set-up change
$\sum_{i} s_{it} \le 1$	$\forall t$	Only one product is processed at any time
$q_{it}, s_{it}, I_{it} \ge 0, s_{it} \in \{0,1\}$	$\forall i,t$	

MTS – only one product at a time

Period	1	2	3	4	5	6	7	8	9	10
Demand 1	20	50	10	50	50	10	20	40	20	30
Demand 2	0	50	25	50	10	50	50	10	20	50
Production 1, units	90	0	0	120	0	0	0	90	0	0
Production 2, units	0	50	75	0	10	50	60	0	20	50
Production 1, s_{1t}	1	0	0	1	0	0	0	1	0	0
Production 2, s_{2t}	0	1	1	0	1	1	1	0	1	1
Set-up time	40	60	0	40	60	0	0	40	60	0
Processing time	90	100	150	120	20	100	120	90	40	100
Total time	130	160	150	160	80	100	120	130	100	100
Net inventory 1	70	20	10	80	30	20	0	50	30	0
Net inventory 2	0	0	50	0	0	0	10	0	0	0
Holding cost	70	20	110	80	30	20	20	50	30	0
Total cost	110	80	110	120	90	20	20	90	90	0
										730

Parameters						
Processing time/unit 1	1	Set-up time 1	40	Big M	999	
Processing time/unit 2	2	Set-up time 2	60	Capacity	160	
Holding cost/unit/period 1	1					
Holding cost/unit/period 2	2					

Aux production 1	999	0	0	999	0	0	0	999	0	0
Aux production 2	0	999	999	0	999	999	999	0	999	999
Product change 1	1	-1	0	1	-1	0	0	1	-1	0
Product change 1	0	1	0	-1	1	0	0	-1	1	0
Set-up 1, z_{it}	1	0	0	1	0	0	0	1	0	0
Set-up 2, z_{it}	0	1	0	0	1	0	0	0	1	0
Aux Set-up 1	999	0	0	999	0	0	0	999	0	0
Aux Set-up 2	0	999	0	0	999	0	0	0	999	0
Concurrent production constr.	1	1	1	1	1	1	1	1	1	1

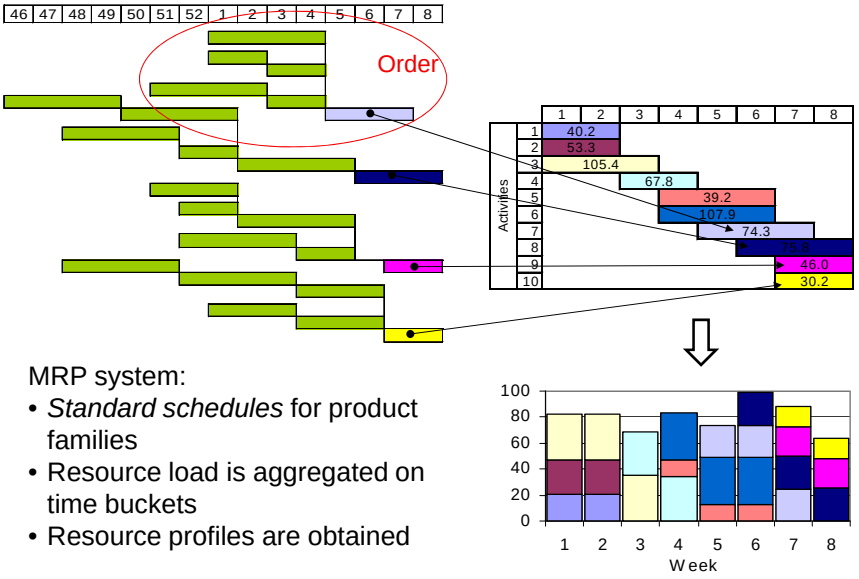
Aggregate planning in MTO-production

- In MTO-production batch formation is not usually a relevant problem, because product variation is large, which in fact is the reason for MTO-production in the firs place
- Product structures and production processes are typically complicated
- Orders appear randomly and resource load is fluctuating
- Pressure for short delivery times is severe
- Leveling of resource load is important for economical reasons
- Levelling affects resource utilization rate and need of capacity and reduces marginal cost due to overtime work, and emergency subcontracting
- Aggregate planning systems appear as MRP (Manufacturing Resource Planning) functions in ERP (Enterprise Resource Planning systems

Aggregate planning in MTO-production

- Purpose of aggregate planning is to schedule orders based on experience, present status and forecasts so that
  - The ability to take new orders is good
  - Resource loading matches capacity
  - Delays are minimized
  - Work in process is kept at a low level
  - Need for rescheduling is kept minimal
- MTO aggregate planning system's basic inputs:
  - Order release dates
  - Order due dates
  - Material delivery times
  - Resource capacities
  - Orders' resource loading (work contents)
  - Process precedence requirements (process flow)

Aggregate planning in MTO-production



MRP system:

- *Standard schedules* for product families
- Resource load is aggregated on time buckets
- Resource profiles are obtained

Aggregate planning in MTO-production

- System user sees the status of the system when allocating new orders and knows basic specifications of the orders
- In addition an order forecast may be available
- Two extreme cases are *forward planning*, in which orders are scheduled as early as possible, and *backward planning*, in which orders are started as late as possible taking capacity constraints and release and due dates into consideration
- Slack and possibilities to affect timing are the bigger the bigger the difference between shortest possible delivery time and due date is and the more capacity is available
- In practice room for optimization exist, because only some orders are taken with minimum delivery time

MTO aggregate planning strategies

- **Forward planning** provides the best ability to take new orders, but work in process is maximized and need for rescheduling may be considerable
- On the other hand, reserving capacity for rush orders is important, because these orders are taken at best price and make your customers happy
- In **backward planning** orders may be lost for lack of capacity
- But, in MTO production customers often make changes to orders afterwards. These changes are the easier to make the later manufacturing takes place

- In practice one has to use sales forecasts and adjust timing tactics accordingly
- Rescheduling causes confusion, errors and generates indirect administrative costs

MTO aggregate planning optimization

Optimization criteria

- In the following models the main optimization criteria are resource levelling or peak load minimization
- The idea is that in practice production needs to deliver products as promised to customers and therefore due dates are constraints that can not be violated and capacity has to be flexible
- Tardiness minimization is also a common optimization criterion with capacity taken as a constraint
- Both am. criteria can be combined in multi-objective optimization, where penalties are determined to resource load limit and due date violations

Optimization model

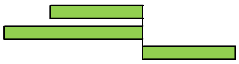
- Because different kinds of orders load resources at different amounts and at different times in relation to finishing time, the problem is difficult to solve manually
- As a linear optimization model it works well

MTO aggregate planning – standard schedules

- Usually customer orders are handled as projects with standard templates for schedules for product families
- Work content is evenly distributed over allowed process step duration
- Resource loading is aggregated on each time bucket

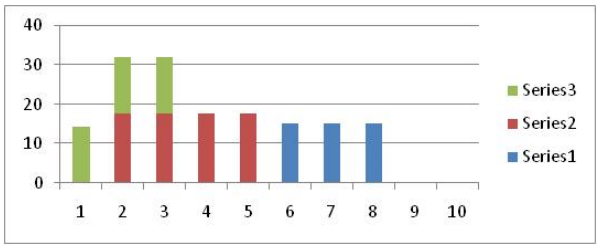
Schedule

Precedences:



Order #	Resource #	P	WC	1	2	3	4	5	6	7	8	9	10
1	1	3	45	0	0	0	0	0	15	15	15	0	0
1	2	2	67	0	0	0	0	0	0	33.5	33.5	0	0
1	3	2	23	0	0	0	0	0	0	0	0	12	11.5
2	1	4	70	0	17.5	17.5	17.5	17.5	0	0	0	0	0
2	2	2	34	0	0	0	17	17	0	0	0	0	0
2	3	3	89	0	0	0	0	0	29.7	29.7	29.7	0	0
3	1	3	43	14.3	14.3	14.3	0	0	0	0	0	0	0
3	2	1	35	0	0	35	0	0	0	0	0	0	0
3	3	2	43	0	0	0	21.5	21.5	0	0	0	0	0

Resource 1



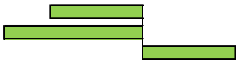
P = Allowed duration (window) for process step
WC = Work Content

MTO aggregate planning – project control

- In project control method single process steps may be moved within timing and precedence constraints

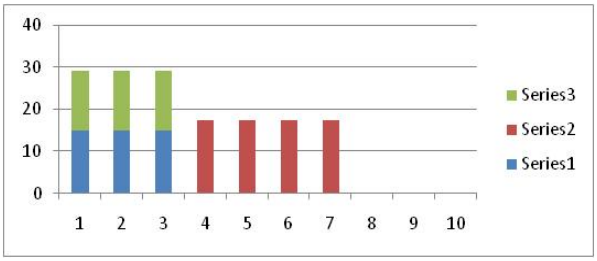
Schedule

Precedences:



Order #	Resource #	P	WC	1	2	3	4	5	6	7	8	9	10
1	1	3	45	15	15	15	0	0	0	0	0	0	0
1	2	2	67	0	0	0	33.5	33.5	0	0	0	0	0
1	3	2	23	0	0	0	0	0	11.5	11.5	0	0	0
2	1	4	70	0	0	0	17.5	17.5	17.5	17.5	0	0	0
2	2	2	34	17	17	0	0	0	0	0	0	0	0
2	3	3	89	0	0	0	0	0	0	0	29.7	29.7	29.7
3	1	3	43	14.3	14.3	14.3	0	0	0	0	0	0	0
3	2	1	35	0	0	35	0	0	0	0	0	0	0
3	3	2	43	0	0	0	21.5	21.5	0	0	0	0	0

Resource 1



MTO planning optimization – project control

$\text{Min} \sum_{\forall k} f_k$ — Minimization of sum of resource load peaks

s.t.

$C_{ik} = \sum_{t=1}^{HZ} t c_{ikt}, \quad \forall i,k$ — c_{ikt} = 1if job is finished at time t, otherwise 0; C_{ik} = Finishing time of job *i*

$\sum_{t=1}^{HZ} c_{ikt} = 1, \quad \forall i,k$ — Jobs end once

$D_i \geq C_{ik} \geq C_{hk} + P_{ik}, \ \forall i,h \in U(i),k$ — Precedences and due dates not violated; *U(i)* is set containing steps preceding *i*

$C_{ik} - P_{ik} \geq A_{ik}, \quad \forall i,k$ — Earliest allowed starting time A_{ik}

$R_{ikt} = WC_{ik} / P_{ik} \sum_{u=t}^{t+P_{ik}-1} c_{iku}, \quad \forall i,k$ — Total work content is evenly distributed over all *t* in timing window R_{ikt}

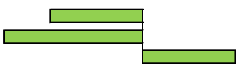
$f_k - \sum_{i=1}^{I} R_{ikt} \geq 0, \quad \forall t,k$ — Peak load

MTO planning optimization – free work content allocation within process step timing window

- Free work content allocation within process step timing window allows much better levelling than even allocation

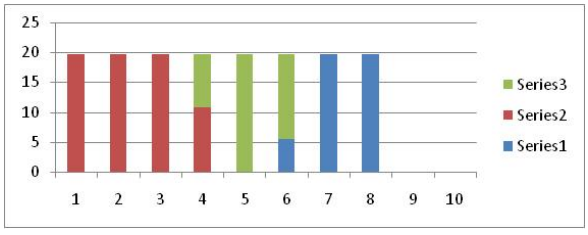
Schedule

Precedences:



Order #	Resource #	P	WC									
1	1	3	45	0	0	0	0	0	5.5	19.8	19.8	0	0
2	1	4	70	19.8	19.8	19.8	10.8	0	0	0	0	0	0
3	1	3	43	0	0	0	9.0	19.8	14.3	0	0	0	0
1	2	2	67	0	0	0	0	0	0	35.0	32.0	0	0
2	2	2	34	0	0	0	34.0	0	0	0	0	0	0
3	2	1	35	0	0	0	0	0	35.0	0	0	0	0
1	3	2	23	0	0	0	0	0	0	0	0	0	23.0
2	3	3	89	0	0	0	0	29.7	29.7	29.7	0	0	0
3	3	2	43	0	0	0	0	0	0	0	29.7	13.3	0

Resource 1



MTO planning optimization – free work content allocation within process step timing window

$$\text{Min} \sum_{\forall k} f_k$$

s.t.

$$C_{ik} = \sum_{t=1}^{HZ} tc_{ikt}, \quad \forall i$$

$$\sum_{t=1}^{HZ} c_{ikt} = 1, \quad \forall i,k$$

$$D_i \geq C_{ik} \geq C_{hk} + P_{ik}, \forall i,h \in U(i),k$$

$$C_{ik} - P_{ik} \geq A_{ik}, \quad \forall i,k$$

$$X_{ikt} = \sum_{u=t}^{t+P_{ik}-1} c_{iku}, \quad \forall i,t$$

X_{ikt} = 1, if t in allowed timing window, otherwise 0

$$R_{ikt} \leq MX_{ikt}, \quad \forall i,k,t$$

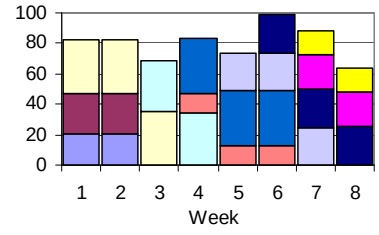
Work may only be allocated to timing window

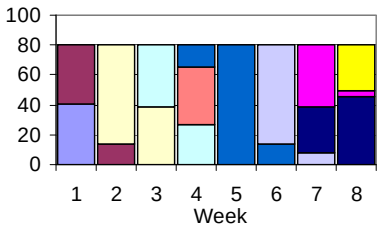
$$\sum_{t=1}^{HZ} R_{ikt} = WC_{ik}, \quad \forall i,k$$

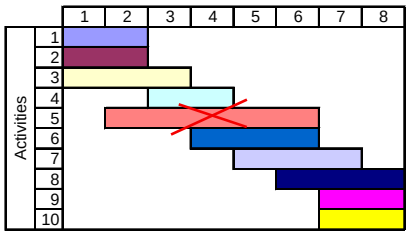
All work content must be allocated

$$f_k - \sum_{i=1}^{I} R_{ikt} \geq 0, \quad \forall t,k$$

From aggregate planning to fine scheduling

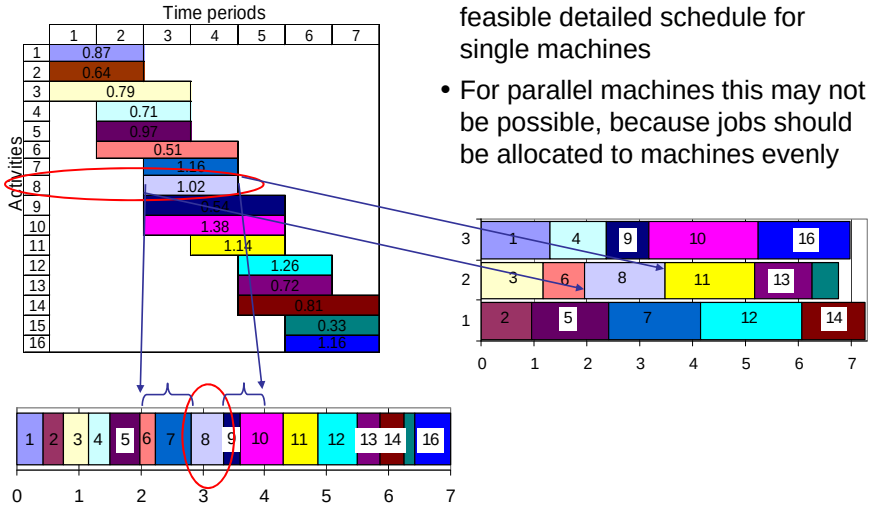






- A particularly good property for a schedule is “tiling” so that no step starts before and ends after another step
- With such timing work is never pre-empted

From aggregate planning to fine scheduling



- Such an aggregate schedule can always be transformed to a feasible detailed schedule for single machines
- For parallel machines this may not be possible, because jobs should be allocated to machines evenly

Sensitivity analysis

- How sensitive is the optimal result to the parameter values? In this case the demand values are uncertain, because the demands depend on our customers' needs, which they probably do not even know themselves very accurately beforehand. So, our values are only educated guesses by our sales department.
- Let us assume two scenarios:
    1) One of our products is a winner/looser in the market place. Therefore sales of that product at all markets increase/decrease in the same ratio.
    2) One of the market areas is experiencing an economic boom/recession. Therefore sales of all products at that market increase/decrease in the same ratio.
- For both scenarios:
    - How big a change can take place before the optimal solution (factory/product setup) changes?
    - How much does one lose by sticking to the original factory/product setup?
    - Analyse and evaluate the sensitivity and make conclusions!

Factory planning

- Factory location
- Manufacturing allocation to factories

Introduction

- Factory location is a major issue strongly dependent on markets, technology, transportation time and costs etc. and therefore it is a strategic decision
- As the am. factors change, the strategy and even factory locations and production allocation is sometimes changed
- Factories are not built or stopped lightly, but their capacity, product allocation and markets served are easier to change
- Mechanical manufacturing is usually not very strongly dependent on any single factor's, like energy, water, customers, proximity and it is generally environmentally problem-free
- Therefore, location decisions are affected by many factors. In the following, we focus on general cost issues

Factory location and production allocation

On costs and other issues

- Economies of scale favor concentration of production to few large factories
- On the other hand, transportation costs, delivery times, and local presence demand decentralization of production to many markets and locations
- Optimal plant location depends on market sizes, logistics costs, costs of factors of production, used technologies, and already existing resources. All these factors should be considered simultaneously
- In practice many other issues influence the decisions, like historical decisions and qualitative non-economical issues
- However the effects of purely technical and economical factors should be known and they can usually be estimated quantitatively using appropriate models

Factory location and production allocation

- **In the short run** production allocation is mainly determined by variable costs and capacities of the factories
- **In the long run**, factories can be started and ended
- The problem can be formulated as an IP and solved easily

We have only one average product and we assume that the transportation cost is included in the variable cost:

$$\text{Min} C = \sum_{\forall i}\sum_{\forall j} V_{ij} x_{ij} + \sum_{\forall i} F_i y_i$$

Variable cost V_{ij} for production of x_{ij} units at factory i for market j and fixed cost F_i

s.t.

$$\sum_{\forall i} x_{ij} \geq D_j, \quad \forall j$$

Demand D_j must be satisfied

$$\sum_{\forall j} x_{ij} \leq W_i, \quad \forall i$$

Capacities of factories W_i must not be exceeded

$$y_i M \geq \sum_j x_{ij} \quad \forall i$$

$$y_i \in \{0,1\}, \quad \forall i$$

Setting of binary auxiliary variables y_i are binary auxiliary variables

Factory location and production allocation

Excel model:

Variable cost		**Factories, *i***			
	Markkina, *j*	Europe	Americas	Far-East	**Demand**
	Europe	200	750	600	3000
	Americas	800	250	400	2000
	Developing c.	1000	1500	1200	1000
	Far-East	800	700	120	2000
Fixed cost, *Fi*		500000	650000	700000	
Capacity, *Wi*		3500	3500	3500	
Production, *xij*					**Total**
	Europe	3000	0	0	3000
	Americas	0	2000	0	2000
	Developing c.	500	0	500	1000
	Far-East	0	0	2000	2000
		3500	2000	2500	
	Factory produces, *yi* :	1	1	1	
9999	Big-M:	9999	9999	9999	
Total cost		1600000	1150000	1540000	4290000

Factory location and production allocation

Some observations about plant location models

- Optimal solutions have the following interesting properties, which also have some practical value:
  - If unit costs of transportation decrease as distance increases (normally valid), optimal production location is always in an end of a line segment connecting locations – “End point optimality”
  - If unit production costs decrease as production volume increases (also usually valid) and there are no capacity constraints, each market is fed by only one factory – “Single assignment property”
- With delivery time or capacity constraints a.m. properties are not often valid (example)
- Constraints and costs related to delivery time, local value added content, purchasing, etc. are easy to add to optimization models

Factory location and production allocation

Other factors

- In addition to cost factors mentioned many other indirect and often qualitative factors must be considered when making decisions. These may not be reasonably contained in optimization models. This type of factors are for example:
  - Cultural issues
  - Political risks
  - Local legislation
  - Natural environmental phenomena
  - Labor availability
  - Labor unions
  - All types of infrastructure
- These may be evaluated using AHP or SWOT analysis etc.
- Such analyses, although systematic, is largely based on subjective evaluation and is therefore approximate. Comparison to results of quantitative cost analyses is also difficult.

Factory location and production allocation

Exact location

The exact location of the factory in the chosen area is again a result of many rather technical and easily judged factors:

- Access to transportation: Roads, railways, harbor, airfield
- Utilities: Water, sewer, electricity district heating
- City planning
- Cost of land and room for extension
- Soil
- Building cost
- Available services
- Supplier closeness
- Labor availability