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,

  $h(x) = 0$

  $g(x) =< 0$

  $x >= 0$

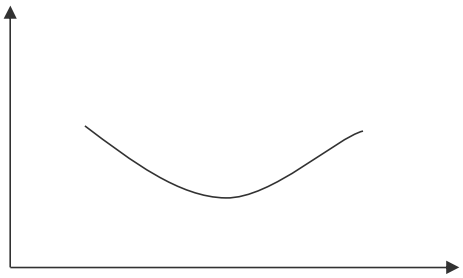
  $x \in R^n$
- In the model ”decision variables” (x) represent those inputs, the values of which are determined by solving the problem.

Optimization – introduction

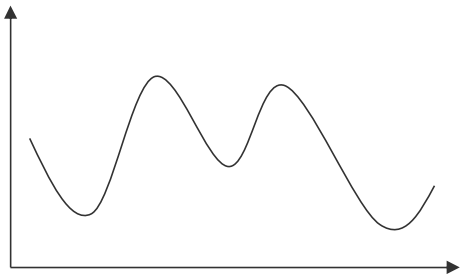
- The value (goodness) of the solution is a function of the decision variables. This function is called *cost function* or *objective function*: *f(x)*
- The variable values can be constrained by setting *constraints*, which are also functions of the decision variables: *h(x)* = 0, *g(x)* =< 0, or *bounds*: *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

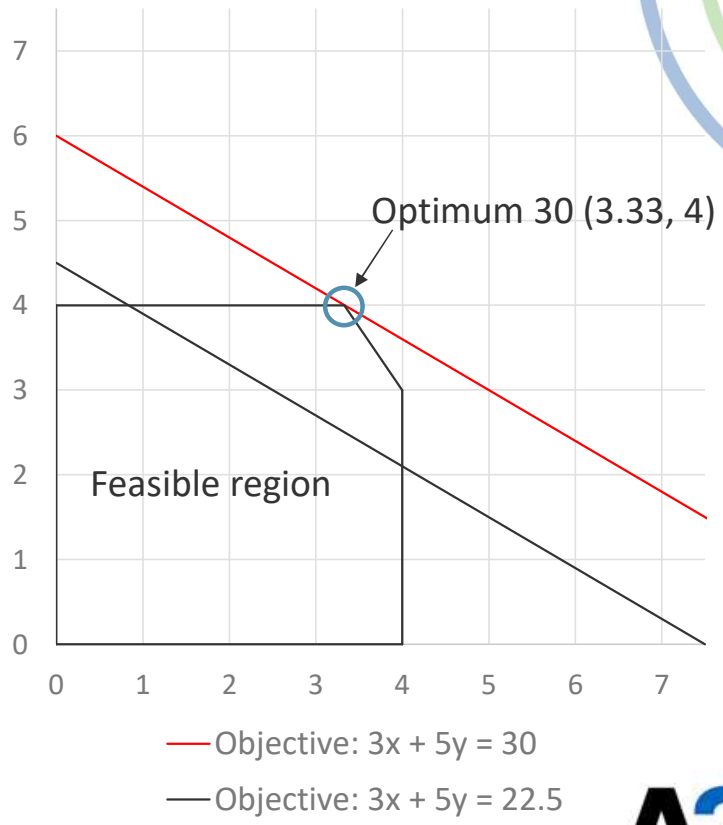
The following simple two-dimensional (two variable) problem is linear, convex, and the decision variables may take real values. To solve it, one must find such values for decision variables *x* and *y* that the value of objective function (*z*) is maximized. In addition, the solution (*x, y*) must satisfy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

Maximize $z = 3x + 5y$

so that,

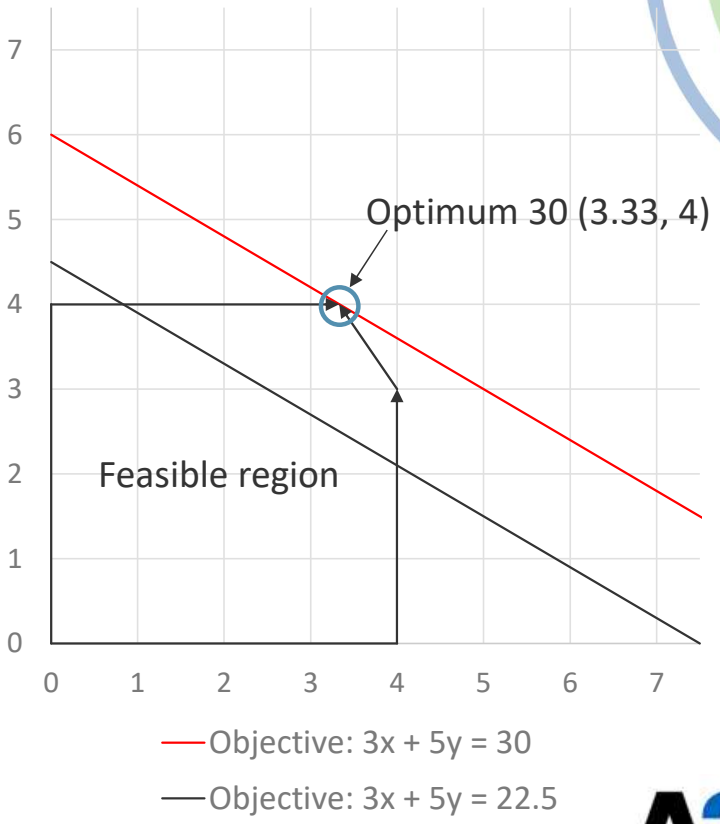
$x <= 4$
$y <= 4$
$3x + 2y <= 18$
$x >= 0, y >= 0$

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



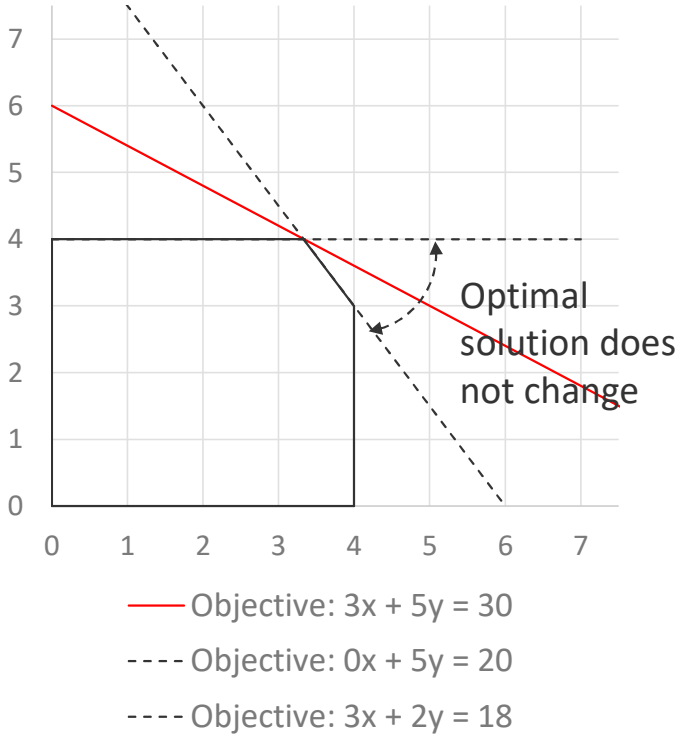
Solving of linear problems

- 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.

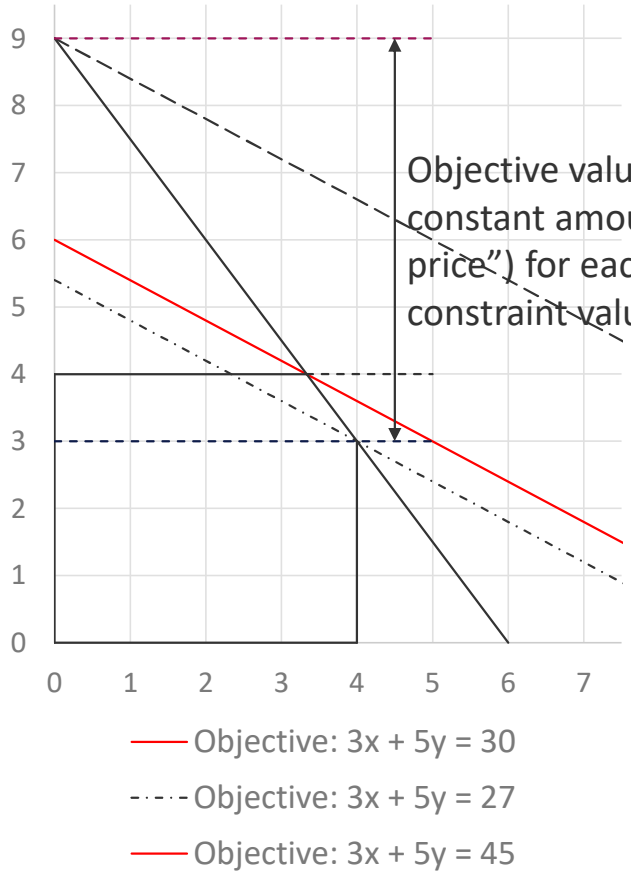


Solution sensitivity to parameter values

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:

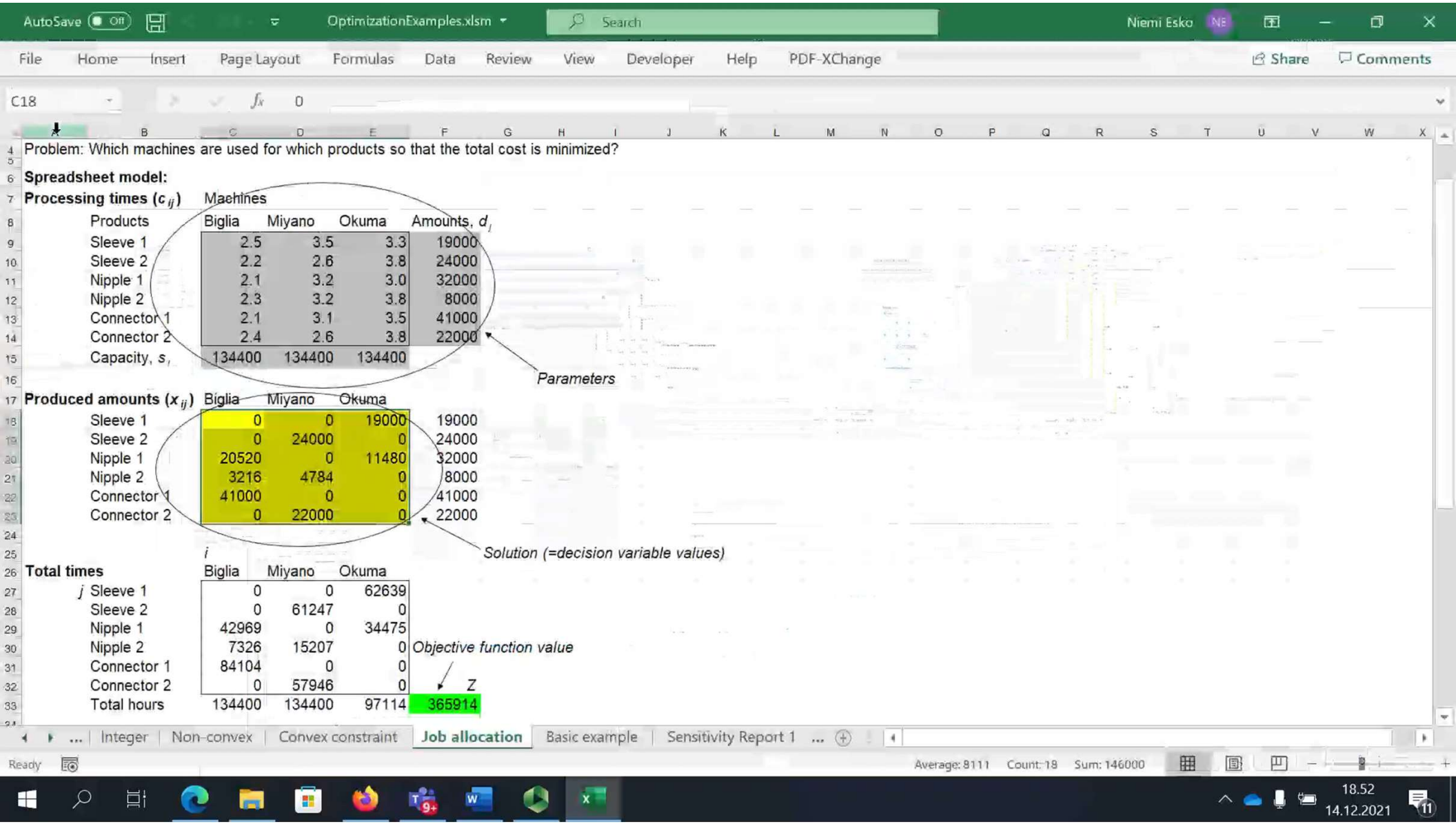
$$\text{Min} \quad Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij},$$

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

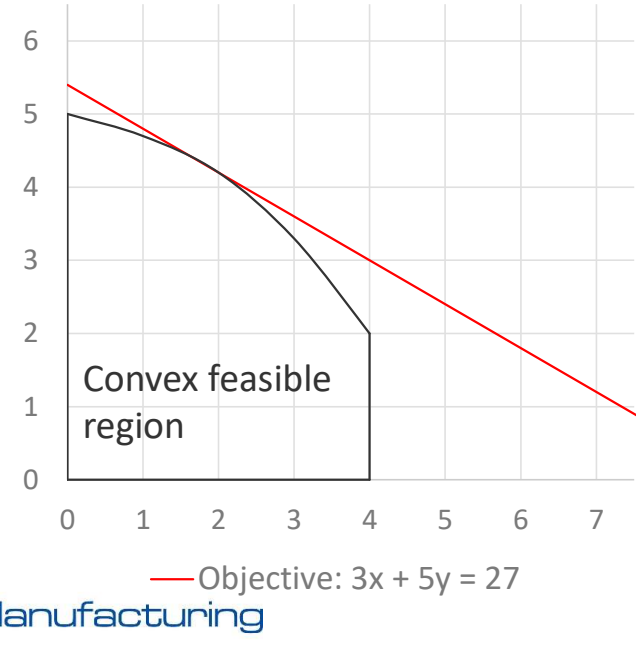
$$\sum_{j=1}^{n} c_{ij} x_{ij} \le s_i \quad \text{for } i = 1,..,m$$

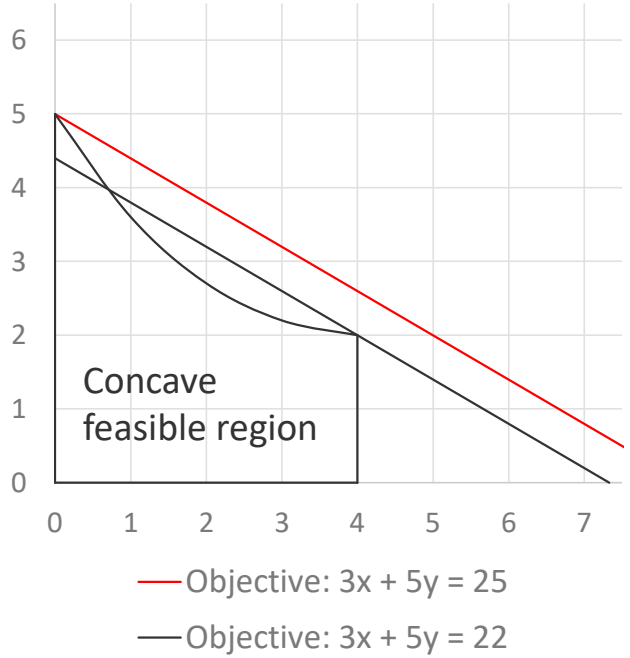
$$\sum_{i=1}^{m} x_{ij} = d_j \quad \text{for } j = 1,..,n$$

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

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

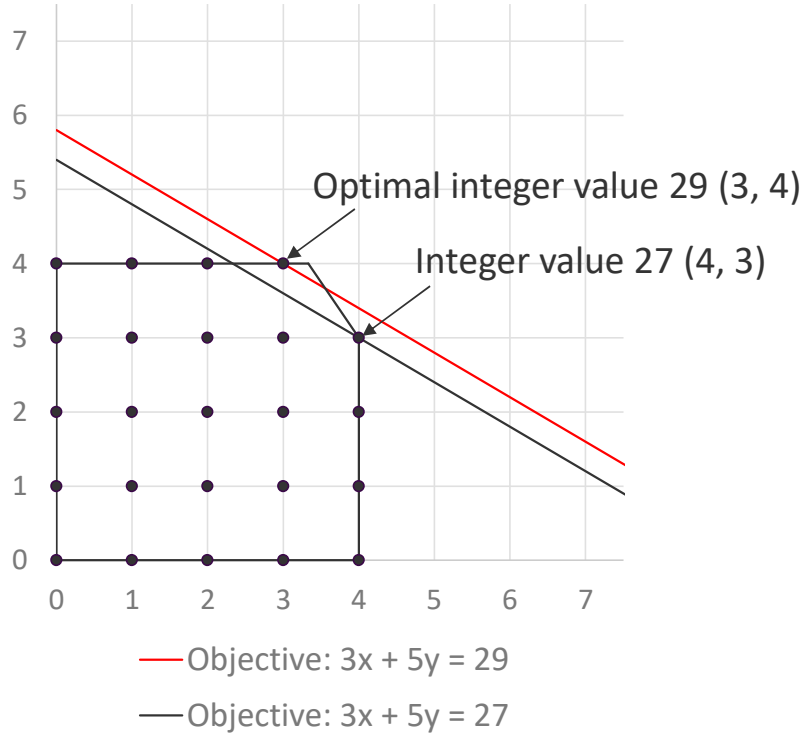
- In integer problems decision variables take integer values
- If some variables take real values, the problem is called a mixed integer programming, MIP, problem
- If variables take only values 1 or 0, the problem is called a binary programming, BIP, problem
- Default assumption is that the problem 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												

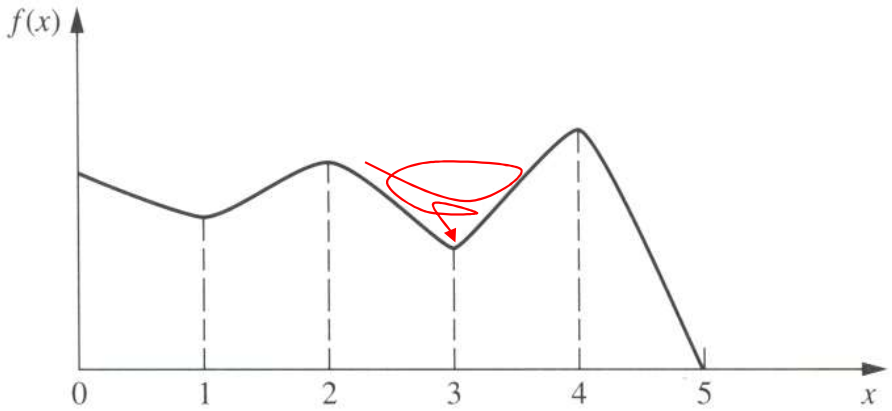
	f_i Finished	Difference	Diffeence^2
1	7	-1	1
1	11	2	4
1	18	1	1
1	15		
1	4	-2	4
1	9	1	1
			11

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