Quantitative Methods in Decision Making

Every manager is exposed to a number of decision situations and problems in his everyday work which he can analyze from two points of view:

- either based on knowledge and experience (qualitative analysis)
- or using data in numerical form and mathematically processing it (quantitative analysis),

Quantitative Methods in Decision Making

Every manager is exposed to a number of decision situations and problems in his everyday work which he can analyze from two points of view:

- either based on knowledge and experience (qualitative analysis)
- or using data in numerical form and mathematically processing it (quantitative analysis),

In certain situations, of course, only one of these analyzes can be performed. However, if the manager uses only the qualitative analysis without numerical calculations, the outcome depends to a large extent on the his good judgment. Conversely, too much confidence in numerical results can be misleading: each numerical solution is accurate only if the model was well constructed. In addition, a quantitative analysis of a problem can be lengthy and inefficient when a decision needs to be taken quickly.

Quantitative Methods in Decision Making

So in what situations are quantitative methods suitable?

Quantitative Methods in Decision Making

So in what situations are quantitative methods suitable? Specifically, if the problem is:

- complex and using simulating reality by a suitable model can help the manager
- very important or connected with very high costs
- new and the decision-maker is lacking experience of solving similar problems
- repeated; using good quantitative procedures saves both time and resources

The art of modeling

By a model we mean a certain representation of a real system. It is never a perfect picture of reality (it is not possible and neither is it desirable). A properly designed model should only capture those features that are important to solve the problem. If we include all the details in the model, it will be too complex, possibly unsolvable and confusing. On the other hand, if there is too much effort to simplify the model, some important facts and links may be omitted. In modeling it is crucial to choose the right relationship between the real world and the model.

The art of modeling

By a model we mean a certain representation of a real system. It is never a perfect picture of reality (it is not possible and neither is it desirable). A properly designed model should only capture those features that are important to solve the problem. If we include all the details in the model, it will be too complex, possibly unsolvable and confusing. On the other hand, if there is too much effort to simplify the model, some important facts and links may be omitted. In modeling it is crucial to choose the right relationship between the real world and the model. The problem should be well formulated so that quantitative methods can be used to solve it and its results should be interpreted and implemented. Experts or specialized software can help solve the mathematical model. Even with the use of computer technology, however, it is advisable to be familiar with the available methods so that in a particular situation we can choose the appropriate algorithm and set its parameters.

Phases of modeling

Most of the process is based on teamwork and good communication.

- Problem definition
- Construction of the model - expressing the defined problem by means of mathematical relationships
- Model solution - using exact or heuristic optimization algorithms or simulations. (plus sensitivity analysis if we are not sure about the exact values of the parameters involved in the model)
- Model validation (Does the solution make sense? Are the results acceptable?)
- Solution implementation - translation into operating instructions

Optimization

When optimizing, we try to choose the "best solution"among all "possible solutions". In a particular problem, these terms must be specified exactly. Futhermore we will describe all possible solutions using the set of M, which we call a feasible region or feasible set, and we will express the quality of the solution through the function $f : M \to \mathbb{R}$ which is called the goal or the objective function. The formulation of the optimization problem follows:
Find $x^* \in M$ such that: $f(x^*) \geq f(x), \forall x \in M,$

Optimization

When optimizing, we try to choose the "best solution"among all "possible solutions". In a particular problem, these terms must be specified exactly. Futhermore we will describe all possible solutions using the set of M, which we call a feasible region or feasible set, and we will express the quality of the solution through the function $f : M \to \mathbb{R}$ which is called the goal or the objective function. The formulation of the optimization problem follows:

Find $x^* \in M$ such that: $f(x^*) \geq f(x),\ \forall x \in M,$

Comment: A maximization problem "$f \to max$"is easily transformable into a minimization problem "$-f \to min$".

Optimization

When optimizing, we try to choose the "best solution"among all "possible solutions". In a particular problem, these terms must be specified exactly. Futhermore we will describe all possible solutions using the set of M, which we call a feasible region or feasible set, and we will express the quality of the solution through the function $f : M \to \mathbb{R}$ which is called the goal or the objective function. The formulation of the optimization problem follows:

Find $x^* \in M$ such that: $f(x^*) \geq f(x), \forall x \in M,$

Comment: A maximization problem "$f \to max$"is easily transformable into a minimization problem "$-f \to min$".

Some applications:

Optimization

When optimizing, we try to choose the "best solution"among all "possible solutions". In a particular problem, these terms must be specified exactly. Futhermore we will describe all possible solutions using the set of M, which we call a feasible region or feasible set, and we will express the quality of the solution through the function $f : M \to \mathbb{R}$ which is called the goal or the objective function. The formulation of the optimization problem follows:

Find $x^* \in M$ such that: $f(x^*) \geq f(x), \forall x \in M,$

Comment: A maximization problem "$f \to max$"is easily transformable into a minimization problem "$-f \to min$".

Some applications:

- Profit maximization (optimal product mix)
- Portfolio optimization
- Scheduling
- Minimization of transportation costs, optimal routes and location of distribution centers
- Project management (minimal time, costs,...)
- Inventory management

Classification of optimization problems

There are two types of optimization problems with respect to the feasible set:

- If any point x in Euclidean space $\mathbb{R}^n$ is feasible, i.e. $M = \mathbb{R}^n$, we use the term unconstrained optimization. The procedure of an analytical solution of such problems is tought on the basic mathematics course
- Otherwise, if $M \subset \mathbb{R}^n$, we talk about constrained optimization. The existence of the solution for optimizing continuous functions on a bounded closed set is guaranteed by the Weierstrass theorem.

An analytical solution is not always easy to find (for example, when there are too many variables or a complicated shape of the feasible set, or there are nonlinear functions involved, etc.). Therefore, special methods have been developed to solve certain types of optimization problems.

Mathematical programming

The term mathematical programming refers to a set of methods used to optimize a criterion expressed as a function of n variables while meeting the constraining conditions generally expressed in the form of equalities and inequalities. The mathematical programming can be divided into:

- linear programming (LP) for problems where both the objective function and the constraints are expressed by means of linear functions of the variables
- nonlinear programming (NLP) when the above mentioned conditions are not met. A special case of NLP is quadratic programming for problems with the objective function in the form of the second degree polynomial and linear constraints.

Mathematical programming

The term mathematical programming refers to a set of methods used to optimize a criterion expressed as a function of *n* variables while meeting the constraining conditions generally expressed in the form of equalities and inequalities. The mathematical programming can be divided into:

- linear programming (LP) for problems where both the objective function and the constraints are expressed by means of linear functions of the variables
- nonlinear programming (NLP) when the above mentioned conditions are not met. A special case of NLP is quadratic programming for problems with the objective function in the form of the second degree polynomial and linear constraints.

We will focus on LP models which are the most widespread. A lot of real problems can be well formulated as an LP problem and software is available for their effective solution. In practice, we can encounter non-linear relationships (e.g. non-proportionality: if the price is not constant, then the income is not proportional to the quantity sold, non-additivity: the synergy effect, etc.), but due to the enormously greater complexity of NLP procedures, it is often preferable to use an approximation by a linear model.

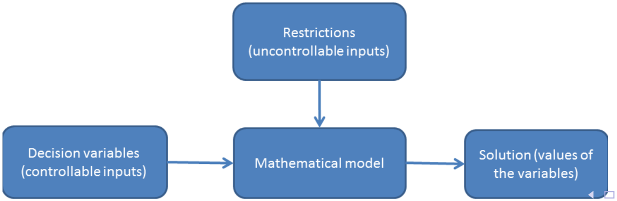
Linear programming

When formulating the mathematical programming problem it is necessary to start from a well-described economic model.

Linear programming

When formulating the mathematical programming problem it is necessary to start from a well-described economic model. It is therefore necessary to know:

- what we want to achieve (so we have to choose the criterion: profit or cost or the volume of production etc., and determine whether we will try to minimize or maximize it)
- controllable inputs, i.e. which variables we can influence in order to achieve the goal (the number of produced pieces of different types in production problems, the load of the vehicles in transportation problems, etc.)
- uncontrollable inputs or constraints that limit us (prices of purchased raw materials, disposition of resources, capacity of the device, etc.)


Restrictions (uncontrollable inputs)
Decision variables (controllable inputs)
Mathematical model
Solution (values of the variables)

LP problem - Optimization of production mix

Further explanations will be illustrated by the following example from Josef Jablonský: "Operations Research, Quantitative Models for Economic Decision Making":

Example : Management of a company for coffee roasting and packing plan the production of two blends - Mocca and Standard. Suppliers have three types of coffee beans K_1, K_2 and K_3 available at 40, 60 and 25 tons respectivelly. The technological procedure determining the mixture composition is summarized in the table.

	Mocca	*Standard*	Capacity [t]
K_1	0,5	0,25	40
K_2	0,5	0,5	60
K_3		0,25	25

Due to the production costs and the price of the blends, a profit of CZK 20,000 or CZK 14,000 per tonne of mixture it Mocca or it Standard respectivelly was calculated. The company's management looks for the production mix which will maximize the profit.

Optimization of the production mix - problem formulation

If we denote the amount of *Mocca* by x_1 blend and the amount of *Standard* blend x_2, we can formulate the problem mathematically as a maximization problem with an objective function:

$$z = 20000x_1 + 14000x_2$$

subject to the constraints

$$\begin{array}{cccc} 0,5x_1 & + & 0,25x_2 & \leq 40 \\ 0,5x_1 & + & 0,5x_2 & \leq 60 \\ & & 0,25x_2 & \leq 25 \\ & & x_1,\ x_2 & \geq 0 \end{array}$$

Optimization of the production mix - problem formulation

If we denote the amount of *Mocca* by x_1 blend and the amount of *Standard* blend x_2, we can formulate the problem mathematically as a maximization problem with an objective function:

$$z = 20000x_1 + 14000x_2$$

subject to the constraints

$$\begin{array}{rcrl} 0,5x_1 & + & 0,25x_2 & \leq 40 \\ 0,5x_1 & + & 0,5x_2 & \leq 60 \\ & & 0,25x_2 & \leq 25 \\ & & x_1,\ x_2 & \geq 0 \end{array}$$

We can also use a matrix formulation:

$z = \mathbf{c}^\top \cdot \mathbf{x} \rightarrow max$ subject to $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0},$

where $\mathbf{x} = (x_1,\ x_2)^\top$ is a vector of the decision variables, $\mathbf{c} = (20,\ 14)^\top$ is a price vector, $\mathbf{b} = (40, 60, 25)^\top$ is a vector of capacities and

$\mathbf{A} = \begin{pmatrix} 0{,}5 & 0{,}25 \\ 0{,}5 & 0{,}5 \\ 0 & 0{,}25 \end{pmatrix}$ is a technology matrix (structural coefficients).

Mathematical formulation of a generic LP problem

A generic LP problem with *n* variables and *m* constraints can be formulated as follows:
minimize (or maximize) function

$z = \sum_{j=1}^{n} c_j x_j$

subject to

$\sum_{j=1}^{n} a_{ij} x_j \; ? \; b_i, \; i = 1, \ldots m$

$x_j \geq 0, \; j = 1, \ldots n,$

where the symbol „? „ represents any of the relationships $\leq, =, \geq$. Restrictions are presented in such a way that the right sides of b_i are non-negative.

Mathematical formulation of a generic LP problem

A generic LP problem with n variables and m constraints can be formulated as follows:
minimize (or maximize) function

$z = \sum_{j=1}^{n} c_j x_j$

subject to

$\sum_{j=1}^{n} a_{ij} x_j \; ? \; b_i, \;\; i = 1, \ldots m$

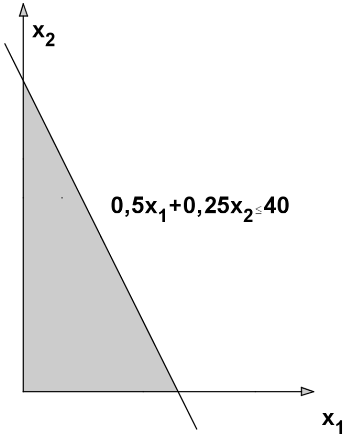
$x_j \geq 0, \;\; j = 1, \ldots n,$

where the symbol „? „ represents any of the relationships $\leq, =, \geq$. Restrictions are presented in such a way that the right sides of b_i are non-negative.

One problem can be formulated in many different ways. You can easily convert a minimization problem to a maximization problem with $-z = \sum_{j=1}^{n}(-c_j)x_j$. Equality restrictions can be rewritten as two inequalities of $\leq$ and $\geq$ with the same coefficients on both sides as in the original equation. Conversion of the inequality constraint to the equality is possible by introducing additional variables, as will be shown later.

Graphical solution of an LP problem

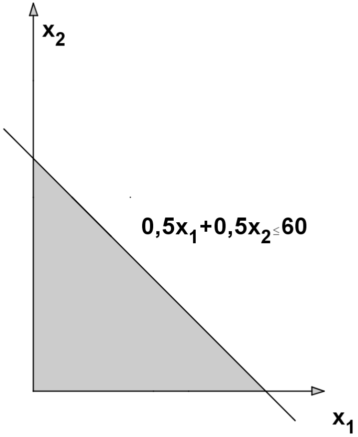
Two dimensional problems can be solved graphically. Let's show it on our coffee problem.



Because of the non-negativity of the variables, we are limited to the first quadrant only. Here we will illustrate the halfplane formed by the points fulfilling the first limiting condition.

Graphical solution of an LP problem

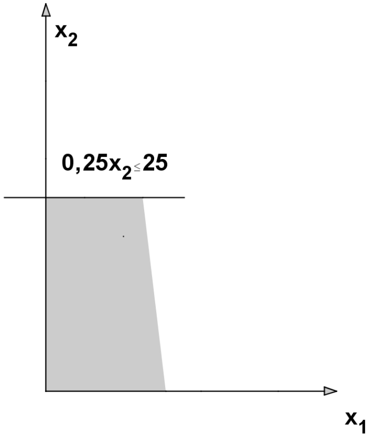
Two dimensional problems can be solved graphically. Let's show it on our coffee problem.



We also show a halfplane formed by the points fulfilling the second limiting condition.

Graphical solution of an LP problem

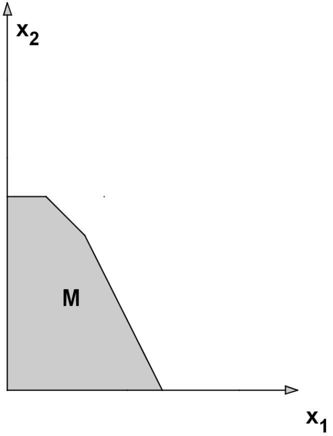
Two dimensional problems can be solved graphically. Let's show it on our coffee problem.



Next we show a halfplane formed by the points fulfilling the third limiting condition.

Graphical solution of an LP problem

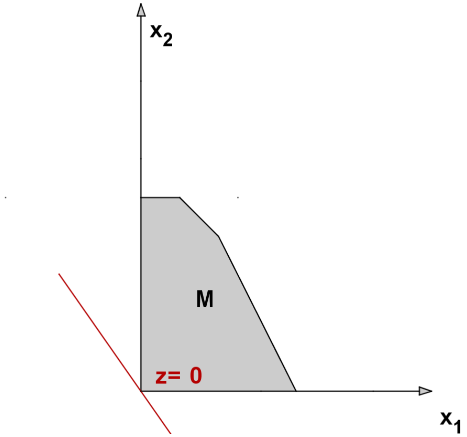
Two dimensional problems can be solved graphically. Let’s show it on our coffee problem.



Feasible region M consists of the points satisfying all three constraints.

Graphical solution of an LP problem

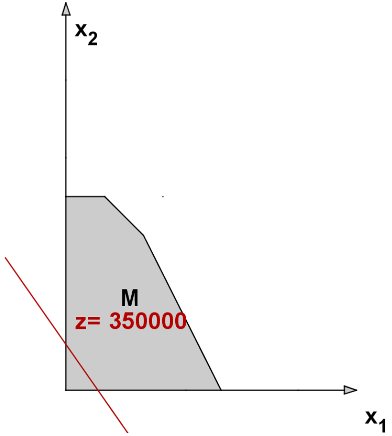
Two dimensional problems can be solved graphically. Let’s show it on our coffee problem.



Contour line of objective function $z = 20000x_1 + 14000x_2 = 0$

Graphical solution of an LP problem

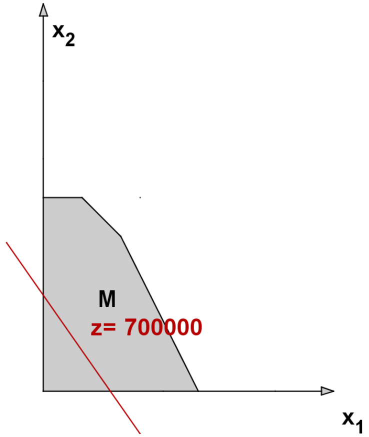
Two dimensional problems can be solved graphically. Let's show it on our coffee problem.



Contour line of objective function $z = 20000x_1 + 14000x_2 = 350000$

Graphical solution of an LP problem

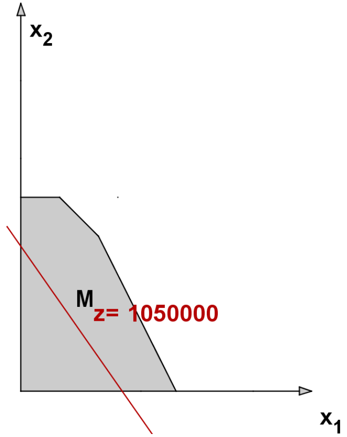
Two dimensional problems can be solved graphically. Let's show it on our coffee problem.



Contour line of objective function $z = 20000x_1 + 14000x_2 = 700000$

Graphical solution of an LP problem

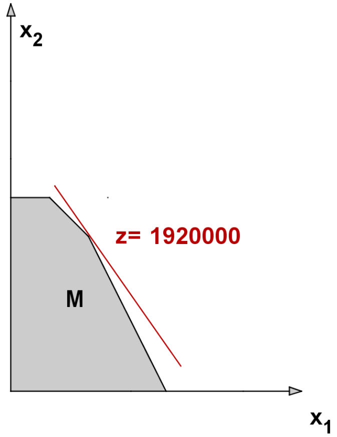
Two dimensional problems can be solved graphically. Let's show it on our coffee problem.



Contour line of objective function $z = 20000x_1 + 14000x_2 = 1050000$

Graphical solution of an LP problem

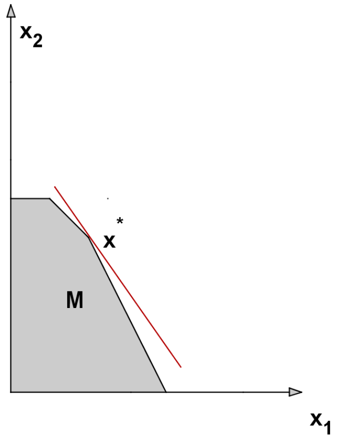
Two dimensional problems can be solved graphically. Let's show it on our coffee problem.



Contour line of objective function $z = 20000x_1 + 14000x_2 = 1920000$

Graphical solution of an LP problem

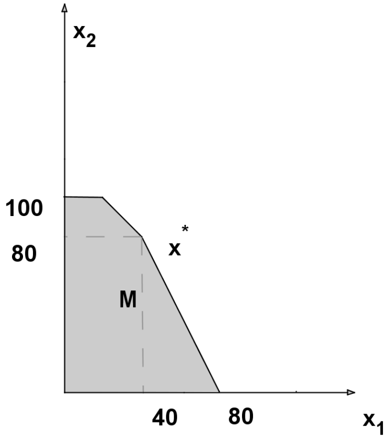
Two dimensional problems can be solved graphically. Let’s show it on our coffee problem.



The contour of the highest value touches *M* at x^*

Graphical solution of an LP problem

Two dimensional problems can be solved graphically. Let’s show it on our coffee problem.



The point $x^* = [40, 80]$ is an optimal solution.

Fundamental theorem of linear programming

Feasible region M is given by the obligatory (non-negativity) conditions and constraints $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$. We can formulate them as equalities:

$$\begin{array}{ccccccccc}
0,5x_1 & + & 0,25x_2 & + & x_3 & & & & = 40 \\
0,5x_1 & + & 0,5x_2 & & & + & x_4 & & = 60 \\
 & & 0,25x_2 & & & & & + \; x_5 & = 25
\end{array}$$

Fundamental theorem of linear programming

Feasible region M is given by the obligatory (non-negativity) conditions and constraints $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$. We can formulate them as equalities:

$$\begin{array}{ccccccccc} 0,5x_1 & + & 0,25x_2 & + & x_3 & & & & = 40 \\ 0,5x_1 & + & 0,5x_2 & & & + & x_4 & & = 60 \\ & & 0,25x_2 & & & & & + \; x_5 & = 25 \end{array}$$

Variables $x_3, \; x_4$ and x_5 are called slacks and can be interpreted economically as unused capacity of raw materials.

Fundamental theorem of linear programming

Feasible region M is given by the obligatory (non-negativity) conditions and constraints $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$. We can formulate them as equalities:

$$\begin{array}{ccccccccl}
0,5x_1 & + & 0,25x_2 & + & x_3 & & & & = 40 \\
0,5x_1 & + & 0,5x_2 & & & + & x_4 & & = 60 \\
 & & 0,25x_2 & & & & & + \quad x_5 & = 25
\end{array}$$

Variables $x_3,\ x_4$ and x_5 are called slacks and can be interpreted economically as unused capacity of raw materials. The system consists of m equations of $m + n$ variables and it can have an infinite number of solutions. The solution with n variables of zero value is called basic. (In 2D problems, the basic solutions correspond to the intersections of the boundary lines of the individual inequalities) We denote the non-zero variables as basic and zero as nonbasic.

Fundamental theorem of linear programming

Feasible region M is given by the obligatory (non-negativity) conditions and constraints $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$. We can formulate them as equalities:

$$\begin{array}{ccccccccccc} 0,5x_1 & + & 0,25x_2 & + & x_3 & & & & & = & 40 \\ 0,5x_1 & + & 0,5x_2 & & & + & x_4 & & & = & 60 \\ & & 0,25x_2 & & & & & + & x_5 & = & 25 \end{array}$$

Variables $x_3,\ x_4$ and x_5 are called slacks and can be interpreted economically as unused capacity of raw materials. The system consists of m equations of $m+n$ variables and it can have an infinite number of solutions. The solution with n variables of zero value is called basic. (In 2D problems, the basic solutions correspond to the intersections of the boundary lines of the individual inequalities) We denote the non-zero variables as basic and zero as nonbasic.

Caution! Not every basic solution is feasible. The feasible basic solutions correspond to the "corner points"of M. In our example, $m=3$, $n=2$, so we will get ? basic solutions and ? of them is feasible.

Fundamental theorem of linear programming

Feasible region M is given by the obligatory (non-negativity) conditions and constraints $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$. We can formulate them as equalities:

$$\begin{array}{ccccccccc}
0,5x_1 & + & 0,25x_2 & + & x_3 & & & & = 40 \\
0,5x_1 & + & 0,5x_2 & & & + & x_4 & & = 60 \\
 & & 0,25x_2 & & & & & + \; x_5 & = 25
\end{array}$$

Variables $x_3,\ x_4$ and x_5 are called slacks and can be interpreted economically as unused capacity of raw materials. The system consists of m equations of $m+n$ variables and it can have an infinite number of solutions. The solution with n variables of zero value is called basic. (In 2D problems, the basic solutions correspond to the intersections of the boundary lines of the individual inequalities) We denote the non-zero variables as basic and zero as nonbasic.

Caution! Not every basic solution is feasible. The feasible basic solutions correspond to the "corner points"of M. In our example, $m = 3$, $n = 2$, so we will get ? basic solutions and ? of them is feasible.

Fundamental theorem of linear programming: If the problem has an optimal solution, then it also has an optimal basic solution.

Simplex tableau

We can use a matrix notation of the system $(\mathbf{A},\ \mathbf{I}) \cdot (x_1,\ x_2,\ x_3,\ x_4,\ x_5)^\top = \mathbf{b}$, where $\mathbf{I}$ is an identity matrix of the size $m = 3$. Every such system of m equations of $m + n$ unknowns, where the matrix on the left side contains all the columns of the identity matrix, is called a system in the canonical form.

Simplex tableau

We can use a matrix notation of the system $(\mathbf{A},\ \mathbf{I}) \cdot (x_1,\ x_2,\ x_3,\ x_4,\ x_5)^\top = \mathbf{b}$, where $\mathbf{I}$ is an identity matrix of the size $m = 3$. Every such system of m equations of $m + n$ unknowns, where the matrix on the left side contains all the columns of the identity matrix, is called a system in the canonical form.

We can easily find at least one solution of such a system:
$x_1 = 0,\ x_2 = 0,\ x_3 = b_1 = 40,\ x_4 = b_2 = 60,\ x_5 = b_3 = 25$. This is even a basic solution. Let's summarize everything in a table:

basic var.	x_1	x_2	x_3	x_4	x_5	b_i
x_3	$\frac{1}{2}$	$\frac{1}{4}$	1	0	0	40
x_4	$\frac{1}{2}$	$\frac{1}{2}$	0	1	0	60
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	-20	-14	0	0	0	0

Simplex tableau

We can use a matrix notation of the system $(\mathbf{A},\ \mathbf{I}) \cdot (x_1,\ x_2,\ x_3,\ x_4,\ x_5)^\top = \mathbf{b}$, where $\mathbf{I}$ is an identity matrix of the size $m = 3$. Every such system of m equations of $m + n$ unknowns, where the matrix on the left side contains all the columns of the identity matrix, is called a system in the canonical form.

We can easily find at least one solution of such a system: $x_1 = 0,\ x_2 = 0,\ x_3 = b_1 = 40,\ x_4 = b_2 = 60,\ x_5 = b_3 = 25$. This is even a basic solution. Let's summarize everything in a table:

basic var.	x_1	x_2	x_3	x_4	x_5	b_i
x_3	$\frac{1}{2}$	$\frac{1}{4}$	1	0	0	40
x_4	$\frac{1}{2}$	$\frac{1}{2}$	0	1	0	60
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	-20	-14	0	0	0	0

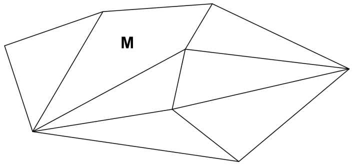
The last row (z_j) corresponds to the coefficients of the objective function in an annulated shape. We converted the equation $z = 20x_1 + 14x_2$ [in thousands of CZK] into $z - 20x_1 - 14x_2 - 0x_3 - 0x_4 - 0x_5 = 0$. For the initial basic solution, we get a zero value of the objective function $z = 0$ (see the lower right corner of the table). We'll call this scheme the initial simplex tableau of the problem.

Simplex method

The Simplex method is an iterative procedure to find the optimal solution of an LP problem. The first step is to find the initial basic solution. This step is simple for problems containing only "$\leq$“ inequalities because of the slacks. For other types of problems we can start by solving the initial problem of minimizing the auxiliary variables expressing the violation of the constraints (the procedure is called the two-phase simplex method). In the next steps the method iteratively calculates new basic solutions with better values for the objective function. After a finite number of steps, a basic solution with the best value of the objective function is found (according to the fundamental LP theorem it is the optimal solution of the whole problem) or it is determined that such a solution does not exist.

Simplex method

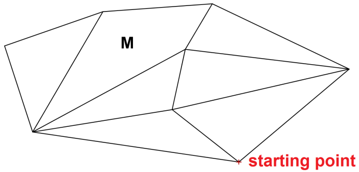
The Simplex method is an iterative procedure to find the optimal solution of an LP problem. The first step is to find the initial basic solution. This step is simple for problems containing only "$\leq$" inequalities because of the slacks. For other types of problems we can start by solving the initial problem of minimizing the auxiliary variables expressing the violation of the constraints (the procedure is called the two-phase simplex method). In the next steps the method iteratively calculates new basic solutions with better values for the objective function. After a finite number of steps, a basic solution with the best value of the objective function is found (according to the fundamental LP theorem it is the optimal solution of the whole problem) or it is determined that such a solution does not exist. In the figure there is a schematic representation of the procedure in 3D.



The feasible region

Simplex method

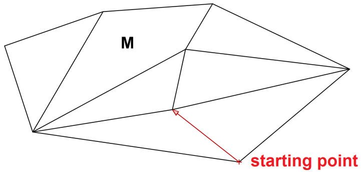
The Simplex method is an iterative procedure to find the optimal solution of an LP problem. The first step is to find the initial basic solution. This step is simple for problems containing only "$\leq$“ inequalities because of the slacks. For other types of problems we can start by solving the initial problem of minimizing the auxiliary variables expressing the violation of the constraints (the procedure is called the two-phase simplex method). In the next steps the method iteratively calculates new basic solutions with better values for the objective function. After a finite number of steps, a basic solution with the best value of the objective function is found (according to the fundamental LP theorem it is the optimal solution of the whole problem) or it is determined that such a solution does not exist. In the figure there is a schematic representation of the procedure in 3D.



Initial basic solution

Simplex method

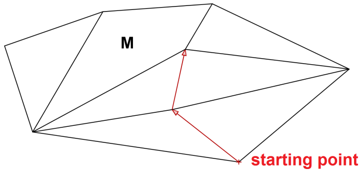
The Simplex method is an iterative procedure to find the optimal solution of an LP problem. The first step is to find the initial basic solution. This step is simple for problems containing only "$\leq$“ inequalities because of the slacks. For other types of problems we can start by solving the initial problem of minimizing the auxiliary variables expressing the violation of the constraints (the procedure is called the two-phase simplex method). In the next steps the method iteratively calculates new basic solutions with better values for the objective function. After a finite number of steps, a basic solution with the best value of the objective function is found (according to the fundamental LP theorem it is the optimal solution of the whole problem) or it is determined that such a solution does not exist. In the figure there is a schematic representation of the procedure in 3D.



We move to an adjacent corner point with a better objective value.

Simplex method

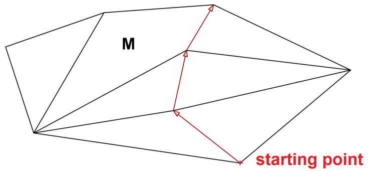
The Simplex method is an iterative procedure to find the optimal solution of an LP problem. The first step is to find the initial basic solution. This step is simple for problems containing only "$\leq$“ inequalities because of the slacks. For other types of problems we can start by solving the initial problem of minimizing the auxiliary variables expressing the violation of the constraints (the procedure is called the two-phase simplex method). In the next steps the method iteratively calculates new basic solutions with better values for the objective function. After a finite number of steps, a basic solution with the best value of the objective function is found (according to the fundamental LP theorem it is the optimal solution of the whole problem) or it is determined that such a solution does not exist. In the figure there is a schematic representation of the procedure in 3D.



We move to an adjacent corner point with a better objective value next time.

Simplex method

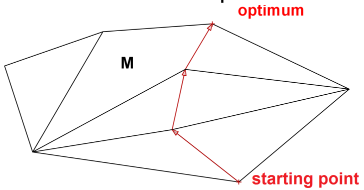
The Simplex method is an iterative procedure to find the optimal solution of an LP problem. The first step is to find the initial basic solution. This step is simple for problems containing only "≤" inequalities because of the slacks. For other types of problems we can start by solving the initial problem of minimizing the auxiliary variables expressing the violation of the constraints (the procedure is called the two-phase simplex method). In the next steps the method iteratively calculates new basic solutions with better values for the objective function. After a finite number of steps, a basic solution with the best value of the objective function is found (according to the fundamental LP theorem it is the optimal solution of the whole problem) or it is determined that such a solution does not exist. In the figure there is a schematic representation of the procedure in 3D.



We move to an adjacent corner point with a better objective value again.

Simplex method

The Simplex method is an iterative procedure to find the optimal solution of an LP problem. The first step is to find the initial basic solution. This step is simple for problems containing only "$\leq$“ inequalities because of the slacks. For other types of problems we can start by solving the initial problem of minimizing the auxiliary variables expressing the violation of the constraints (the procedure is called the two-phase simplex method). In the next steps the method iteratively calculates new basic solutions with better values for the objective function. After a finite number of steps, a basic solution with the best value of the objective function is found (according to the fundamental LP theorem it is the optimal solution of the whole problem) or it is determined that such a solution does not exist. In the figure there is a schematic representation of the procedure in 3D.



There is no better adjacent vertex, we have achieved the optimum.

Iteration step of the simplex method

Numbers z_j in the bottom row of the simplex tableau are called reduced prices. They express the rate of change in an objective function when switching to a new basic solution. If a variable x_k enters the base (so its value grows from 0 to $t > 0$), the objective function value increases by $\Delta z = -t \cdot z_k$.

Iteration step of the simplex method

Numbers z_j in the bottom row of the simplex tableau are called reduced prices. They express the rate of change in an objective function when switching to a new basic solution. If a variable x_k enters the base (so its value grows from 0 to $t > 0$), the objective function value increases by $\Delta z = -t \cdot z_k$. We want this Δz to be positive when maximizing, which means that z_k has to be negative. If all reduced prices are non-negative, the objective value cannot be increased and the solution is optimal.

Iteration step of the simplex method

Numbers z_j in the bottom row of the simplex tableau are called reduced prices. They express the rate of change in an objective function when switching to a new basic solution. If a variable x_k enters the base (so its value grows from 0 to $t > 0$), the objective function value increases by $\Delta z = -t \cdot z_k$. We want this Δz to be positive when maximizing, which means that z_k has to be negative. If all reduced prices are non-negative, the objective value cannot be increased and the solution is optimal. Else we choose x_k with the least value of z_k (this x_k is called the entering variable) and substitute a basic (leaving) variable with it.

Iteration step of the simplex method

Numbers z_j in the bottom row of the simplex tableau are called reduced prices. They express the rate of change in an objective function when switching to a new basic solution. If a variable x_k enters the base (so its value grows from 0 to $t > 0$), the objective function value increases by $\Delta z = -t \cdot z_k$. We want this Δz to be positive when maximizing, which means that z_k has to be negative. If all reduced prices are non-negative, the objective value cannot be increased and the solution is optimal. Else we choose x_k with the least value of z_k (this x_k is called the entering variable) and substitute a basic (leaving) variable with it. For the simplex tableau

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_3	$\frac{1}{2}$	$\frac{1}{4}$	1	0	0	40
x_4	$\frac{1}{2}$	$\frac{1}{2}$	0	1	0	60
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	-20	-14	0	0	0	0

we have an entering variable x_1 because -20 is the lowest reduced price.

Iteration step of the simplex method

The choice of the leaving variable is based on the need to maintain the feasibility of the solution, i.e. the non-negativity of all the basic variables. Let us write this condition for the new values of the original basic variables $x_3, \ x_4, \ x_5$. If we set $x_1 = t$, we get:
$x_3 = 40 - \frac{1}{2}t \geq 0$
$x_4 = 60 - \frac{1}{2}t \geq 0$
$x_5 = 25 - 0 \geq 0$
Obviously the largest such t is $t = 80$, for which we get $x_3 = 0$. It now becomes the leaving variable. We can upgrade the table using elementary row operations to get number one in the upper left corner and zeros below it.

Iteration step of the simplex method

The choice of the leaving variable is based on the need to maintain the feasibility of the solution, i.e. the non-negativity of all the basic variables. Let us write this condition for the new values of the original basic variables $x_3,\ x_4,\ x_5$. If we set $x_1 = t$, we get:
$x_3 = 40 - \frac{1}{2}t \geq 0$
$x_4 = 60 - \frac{1}{2}t \geq 0$
$x_5 = 25 - 0 \geq 0$
Obviously the largest such t is $t = 80$, for which we get $x_3 = 0$. It now becomes the leaving variable. We can upgrade the table using elementary row operations to get number one in the upper left corner and zeros below it.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_3	$\frac{1}{2}$	$\frac{1}{4}$	1	0	0	40
x_4	$\frac{1}{2}$	$\frac{1}{2}$	0	1	0	60
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	-20	-14	0	0	0	0

We multiply the first row by two.

Iteration step of the simplex method

The choice of the leaving variable is based on the need to maintain the feasibility of the solution, i.e. the non-negativity of all the basic variables. Let us write this condition for the new values of the original basic variables $x_3,\ x_4,\ x_5$. If we set $x_1 = t$, we get:
$x_3 = 40 - \frac{1}{2}t \geq 0$
$x_4 = 60 - \frac{1}{2}t \geq 0$
$x_5 = 25 - 0 \geq 0$
Obviously the largest such t is $t = 80$, for which we get $x_3 = 0$. It now becomes the leaving variable. We can upgrade the table using elementary row operations to get number one in the upper left corner and zeros below it.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_4	$\frac{1}{2}$	$\frac{1}{2}$	0	1	0	60
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	-20	-14	0	0	0	0

Subtract a 0.5 multiplier of the first row from the second row.

Iteration step of the simplex method

The choice of the leaving variable is based on the need to maintain the feasibility of the solution, i.e. the non-negativity of all the basic variables. Let us write this condition for the new values of the original basic variables $x_3,\ x_4,\ x_5$. If we set $x_1 = t$, we get:
$x_3 = 40 - \frac{1}{2}t \geq 0$
$x_4 = 60 - \frac{1}{2}t \geq 0$
$x_5 = 25 - 0 \geq 0$
Obviously the largest such t is $t = 80$, for which we get $x_3 = 0$. It now becomes the leaving variable. We can upgrade the table using elementary row operations to get number one in the upper left corner and zeros below it.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_4	0	$\frac{1}{4}$	-1	1	0	20
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	-20	-14	0	0	0	0

Finally we add a 20 multiplier of the first row to the last one.

Iteration step of the simplex method

The choice of the leaving variable is based on the need to maintain the feasibility of the solution, i.e. the non-negativity of all the basic variables. Let us write this condition for the new values of the original basic variables $x_3,\ x_4,\ x_5$. If we set $x_1 = t$, we get:
$x_3 = 40 - \frac{1}{2}t \geq 0$
$x_4 = 60 - \frac{1}{2}t \geq 0$
$x_5 = 25 - 0 \geq 0$
Obviously the largest such t is $t = 80$, for which we get $x_3 = 0$. It now becomes the leaving variable. We can upgrade the table using elementary row operations to get number one in the upper left corner and zeros below it.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_4	0	$\frac{1}{4}$	-1	1	0	20
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	0	-4	40	0	0	1600

We have a new table.

The second iteration

Negative reduced price $z_2 = -4$ indicates that we can increase the objective value by choosing x_2 as the entering variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_4	0	$\frac{1}{4}$	-1	1	0	20
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	0	-4	40	0	0	1600

The second iteration

Negative reduced price $z_2 = -4$ indicates that we can increase the objective value by choosing x_2 as the entering variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_4	0	$\frac{1}{4}$	-1	1	0	20
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	0	-4	40	0	0	1600

When $x_2 = t$, the largest value of t can be $min\{2 \times 80,\ 4 \times 20,\ 4 \times 25\} = 80$. We get new values of basic variables,
$x_1 = 80 - \frac{1}{2}t = 40,\ x_4 = 20 - \frac{1}{4}t = 0,\ x_5 = 25 - \frac{1}{4}t = 5$. So x_4 becomes nonbasic; it is a leaving variable.

The second iteration

Negative reduced price $z_2 = -4$ indicates that we can increase the objective value by choosing x_2 as the entering variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_4	0	$\frac{1}{4}$	-1	1	0	20
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	0	-4	40	0	0	1600

When $x_2 = t$, the largest value of t can be $min\{2 \times 80,\ 4 \times 20,\ 4 \times 25\} = 80$.
We get new values of basic variables,
$x_1 = 80 - \frac{1}{2}t = 40,\ x_4 = 20 - \frac{1}{4}t = 0,\ x_5 = 25 - \frac{1}{4}t = 5$. So x_4 becomes nonbasic; it is a leaving variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_4	0	$\frac{1}{4}$	-1	1	0	20
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	0	-4	40	0	0	1600

We multiply the second row by 4.

The second iteration

Negative reduced price $z_2 = -4$ indicates that we can increase the objective value by choosing x_2 as the entering variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_4	0	$\frac{1}{4}$	-1	1	0	20
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	0	-4	40	0	0	1600

When $x_2 = t$, the largest value of t can be $min\{2 \times 80,\ 4 \times 20,\ 4 \times 25\} = 80$.
We get new values of basic variables,
$x_1 = 80 - \frac{1}{2}t = 40,\ x_4 = 20 - \frac{1}{4}t = 0,\ x_5 = 25 - \frac{1}{4}t = 5$. So x_4 becomes nonbasic; it is a leaving variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_2	0	1	-4	4	0	80
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	0	-4	40	0	0	1600

We subtract its 0.5 multiplier from the first row.

The second iteration

Negative reduced price $z_2 = -4$ indicates that we can increase the objective value by choosing x_2 as the entering variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_4	0	$\frac{1}{4}$	-1	1	0	20
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	0	-4	40	0	0	1600

When $x_2 = t$, the largest value of t can be $min\{2 \times 80,\ 4 \times 20,\ 4 \times 25\} = 80$. We get new values of basic variables,
$x_1 = 80 - \frac{1}{2}t = 40,\ x_4 = 20 - \frac{1}{4}t = 0,\ x_5 = 25 - \frac{1}{4}t = 5$. So x_4 becomes nonbasic; it is a leaving variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	0	4	0	0	40
x_2	0	1	-4	4	0	80
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	0	-4	40	0	0	1600

We subtract its 0.25 multiplier from the third row.

The second iteration

Negative reduced price $z_2 = -4$ indicates that we can increase the objective value by choosing x_2 as the entering variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_4	0	$\frac{1}{4}$	-1	1	0	20
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	0	-4	40	0	0	1600

When $x_2 = t$, the largest value of t can be $min\{2 \times 80,\ 4 \times 20,\ 4 \times 25\} = 80$. We get new values of basic variables,
$x_1 = 80 - \frac{1}{2}t = 40,\ x_4 = 20 - \frac{1}{4}t = 0,\ x_5 = 25 - \frac{1}{4}t = 5$. So x_4 becomes nonbasic; it is a leaving variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	0	4	0	0	40
x_2	0	1	-4	4	0	80
x_5	0	0	1	-1	1	5
z_j	0	-4	40	0	0	1600

Finally we add its 4 multiplier to the last one.

The second iteration

Negative reduced price $z_2 = -4$ indicates that we can increase the objective value by choosing x_2 as the entering variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	$\frac{1}{2}$	2	0	0	80
x_4	0	$\frac{1}{4}$	-1	1	0	20
x_5	0	$\frac{1}{4}$	0	0	1	25
z_j	0	-4	40	0	0	1600

When $x_2 = t$, the largest value of t can be $min\{2 \times 80,\ 4 \times 20,\ 4 \times 25\} = 80$.
We get new values of basic variables,
$x_1 = 80 - \frac{1}{2}t = 40,\ x_4 = 20 - \frac{1}{4}t = 0,\ x_5 = 25 - \frac{1}{4}t = 5$. So x_4 becomes nonbasic; it is a leaving variable.

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	0	4	0	0	40
x_2	0	1	-4	4	0	80
x_5	0	0	1	-1	1	5
z_j	0	0	24	16	0	1920

We have got a new simplex table.

The termination of the computation

All reduced prices are non-negative in the final table:

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	0	4	0	0	40
x_2	0	1	-4	4	0	80
x_5	0	0	1	-1	1	5
z_j	0	0	24	16	0	1920

It is not possible to increase the objective value; the maximal profit is 1.920 millions of CZK. The variables $x_3,\ x_4$ are nonbasic so they are of zero value. The values of basic variables can be determined from the table: $x_1 = 40,\ x_2 = 80,\ x_5 = 5$. This means that in order to maximize the profit we should produce 40 tons of the blend *Mocca* and 80 tons of the blend *Standard*. The first two raw materials would be used at their maximum available amount, but five tons of the third raw material would be redundant.

The termination of the computation

All reduced prices are non-negative in the final table:

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	0	4	0	0	40
x_2	0	1	-4	4	0	80
x_5	0	0	1	-1	1	5
z_j	0	0	24	16	0	1920

It is not possible to increase the objective value; the maximal profit is 1.920 millions of CZK. The variables $x_3, \ x_4$ are nonbasic so they are of zero value. The values of basic variables can be determined from the table: $x_1 = 40, \ x_2 = 80, \ x_5 = 5$. This means that in order to maximize the profit we should produce 40 tons of the blend *Mocca* and 80 tons of the blend *Standard*. The first two raw materials would be used at their maximum available amount, but five tons of the third raw material would be redundant.

Comment: The procedure is similar also for minimization problems, but we treat the signs in the last row in an opposite way (we choose the entering variable according to the largest reduced price and terminate the computations when all of them are negative.)

Two-phase simplex method

If there are also constraints of other types than "$\leq$"in the problem, we have to find the initial feasible solution first. To do this, we use the first phase of a simplex method. Let's show it on an illustrative example.

$z = x_1 - x_2 \to \min$

subject to

$$\begin{array}{rcrl} & & x_1 & \geq 2 \\ & & x_2 & \geq 2 \\ 5x_1 & + & 10x_2 & \leq 50 \end{array}$$

Two-phase simplex method

If there are also constraints of other types than "$\leq$"in the problem, we have to find the initial feasible solution first. To do this, we use the first phase of a simplex method. Let's show it on an illustrative example.

$z = x_1 - x_2 \to \min$

subject to

$$\begin{array}{rcrl} & & x_1 & \geq 2 \\ & & x_2 & \geq 2 \\ 5x_1 & + & 10x_2 & \leq 50 \end{array}$$

Constraints can be converted to equalities using slack variables:

$$\begin{array}{rcrrrrl} & & x_1 & -x_3 & & & = 2 \\ & & x_2 & & -x_4 & & = 2 \\ 5x_1 & + & 10x_2 & & & +x_5 & = 50 \end{array}$$

Two-phase simplex method

If there are also constraints of other types than "$\leq$"in the problem, we have to find the initial feasible solution first. To do this, we use the first phase of a simplex method. Let's show it on an illustrative example.

$z = x_1 - x_2 \to \min$

subject to

$$\begin{array}{rcrl} & & x_1 & \geq 2 \\ & & x_2 & \geq 2 \\ 5x_1 & + & 10x_2 & \leq 50 \end{array}$$

Constraints can be converted to equalities using slack variables:

$$\begin{array}{rcrrrrl} & & x_1 & -x_3 & & & = 2 \\ & & x_2 & & -x_4 & & = 2 \\ 5x_1 & + & 10x_2 & & & +x_5 & = 50 \end{array}$$

Unfortunately, we don't get a system in a canonical form because the coefficients at x_3 and x_4 are not $= 1$. Therefore, we add the non-negative auxiliary variables y_1, y_2 to the left side of the appropriate constraints and these variables will already be basic (together with x_5). For the starting point we have values $y_1 = y_2 = 2$, $x_5 = 50$, but this doesn't give us a feasible solution for the original problem.

Two-phase simplex method

To obtain a feasible solution of the original problem, we have to ensure that $y_1 = y_2 = 0$. This can be done by minimizing the auxiliary objective function $z' = y_1 + y_2$ (if this function has a minimum > 0, then the original problem has no feasible solution). We express z' using non-basic variables and write down the resulting reduced prices into the simplex tableau:
$z' = (2 - x_1 + x_3) + (2 - x_2 + x_4) = 4 - x_1 - x_2 + x_3 + x_4$

Two-phase simplex method

To obtain a feasible solution of the original problem, we have to ensure that $y_1 = y_2 = 0$. This can be done by minimizing the auxiliary objective function $z' = y_1 + y_2$ (if this function has a minimum > 0, then the original problem has no feasible solution). We express z' using non-basic variables and write down the resulting reduced prices into the simplex tableau:
$z' = (2 - x_1 + x_3) + (2 - x_2 + x_4) = 4 - x_1 - x_2 + x_3 + x_4$

basic var.	x_1	x_2	x_3	x_4	x_5	y_1	y_2	β_i
y_1	1	0	-1	0	0	1	0	2
y_2	0	1	0	-1	0	0	1	2
x_5	5	10	0	0	1	0	0	50
z'_j	1	1	-1	-1	0	0	0	4

We may choose as a leaving variable either x_1 or x_2. Let us choose the second one.

Two-phase simplex method

To obtain a feasible solution of the original problem, we have to ensure that $y_1 = y_2 = 0$. This can be done by minimizing the auxiliary objective function $z' = y_1 + y_2$ (if this function has a minimum > 0, then the original problem has no feasible solution). We express z' using non-basic variables and write down the resulting reduced prices into the simplex tableau:
$z' = (2 - x_1 + x_3) + (2 - x_2 + x_4) = 4 - x_1 - x_2 + x_3 + x_4$

basic var.	x_1	x_2	x_3	x_4	x_5	y_1	y_2	β_i
y_1	1	0	-1	0	0	1	0	2
y_2	0	1	0	-1	0	0	1	2
x_5	5	10	0	0	1	0	0	50
z'_j	1	1	-1	-1	0	0	0	4

We subtract 10 times the second row from the third one.

Two-phase simplex method

To obtain a feasible solution of the original problem, we have to ensure that $y_1 = y_2 = 0$. This can be done by minimizing the auxiliary objective function $z' = y_1 + y_2$ (if this function has a minimum > 0, then the original problem has no feasible solution). We express z' using non-basic variables and write down the resulting reduced prices into the simplex tableau:
$z' = (2 - x_1 + x_3) + (2 - x_2 + x_4) = 4 - x_1 - x_2 + x_3 + x_4$

basic var.	x_1	x_2	x_3	x_4	x_5	y_1	y_2	β_i
y_1	1	0	-1	0	0	1	0	2
y_2	0	1	0	-1	0	0	1	2
x_5	5	0	0	10	1	0	-10	30
z'_j	1	1	-1	-1	0	0	0	4

We subtract the second row from the fourth.

Two-phase simplex method

To obtain a feasible solution of the original problem, we have to ensure that $y_1 = y_2 = 0$. This can be done by minimizing the auxiliary objective function $z' = y_1 + y_2$ (if this function has a minimum > 0, then the original problem has no feasible solution). We express z' using non-basic variables and write down the resulting reduced prices into the simplex tableau:
$z' = (2 - x_1 + x_3) + (2 - x_2 + x_4) = 4 - x_1 - x_2 + x_3 + x_4$

basic var.	x_1	x_2	x_3	x_4	x_5	y_1	y_2	β_i
y_1	1	0	-1	0	0	1	0	2
x_2	0	1	0	-1	0	0	1	2
x_5	5	0	0	10	1	0	-10	30
z'_j	1	0	-1	0	0	0	-1	2

We have a new table; the entering variable is x_1. We subtract five times the first row from the third one.

Two-phase simplex method

To obtain a feasible solution of the original problem, we have to ensure that $y_1 = y_2 = 0$. This can be done by minimizing the auxiliary objective function $z' = y_1 + y_2$ (if this function has a minimum > 0, then the original problem has no feasible solution). We express z' using non-basic variables and write down the resulting reduced prices into the simplex tableau:
$z' = (2 - x_1 + x_3) + (2 - x_2 + x_4) = 4 - x_1 - x_2 + x_3 + x_4$

basic var.	x_1	x_2	x_3	x_4	x_5	y_1	y_2	β_i
y_1	1	0	-1	0	0	1	0	2
x_2	0	1	0	-1	0	0	1	2
x_5	0	0	5	10	1	-5	-10	20
z'_j	1	0	-1	0	0	0	-1	2

We subtract the first row from the last one.

Two-phase simplex method

To obtain a feasible solution of the original problem, we have to ensure that $y_1 = y_2 = 0$. This can be done by minimizing the auxiliary objective function $z' = y_1 + y_2$ (if this function has a minimum > 0, then the original problem has no feasible solution). We express z' using non-basic variables and write down the resulting reduced prices into the simplex tableau:
$z' = (2 - x_1 + x_3) + (2 - x_2 + x_4) = 4 - x_1 - x_2 + x_3 + x_4$

basic var.	x_1	x_2	x_3	x_4	x_5	y_1	y_2	β_i
x_1	1	0	-1	0	0	1	0	2
x_2	0	1	0	-1	0	0	1	2
x_5	0	0	5	10	1	-5	-10	20
z'_j	0	0	0	0	0	-1	-1	0

We have found a minimum of an auxiliary function $z'_{opt} = 0$, so we can start phase 2: we omit y_1, y_2 and minimize function $z = x_1 - x_2 = (2 - x_3) - (2 - x_4) = -x_3 + x_4$ whose reduced prices we add to the table. We start from the point $[2, 2, 0, 0, 20]$.

Note: If we obtain $z'_{opt} \neq 0$, there would be no feasible solution to the original problem.

Two-phase simplex method

The second phase can be completed as usual:

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	0	-1	0	0	2
x_2	0	1	0	-1	0	2
x_5	0	0	5	10	1	20
z_j	0	0	-1	1	0	0

The entering variable is x_4.

Two-phase simplex method

The second phase can be completed as usual:

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	0	-1	0	0	2
x_2	0	1	0	-1	0	2
x_5	0	0	5	10	1	20
z_j	0	0	-1	1	0	0

We divide the third row by 10.

Two-phase simplex method

The second phase can be completed as usual:

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	0	-1	0	0	2
x_2	0	1	0	-1	0	2
x_5	0	0	$\frac{1}{2}$	1	$\frac{1}{10}$	2
z_j	0	0	-1	1	0	0

We add the third row to the second one.

Two-phase simplex method

The second phase can be completed as usual:

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	0	-1	0	0	2
x_2	0	1	$\frac{1}{2}$	0	$\frac{1}{10}$	4
x_5	0	0	$\frac{1}{2}$	1	$\frac{1}{10}$	2
z_j	0	0	-1	1	0	0

We subtract the third row from the last one.

Two-phase simplex method

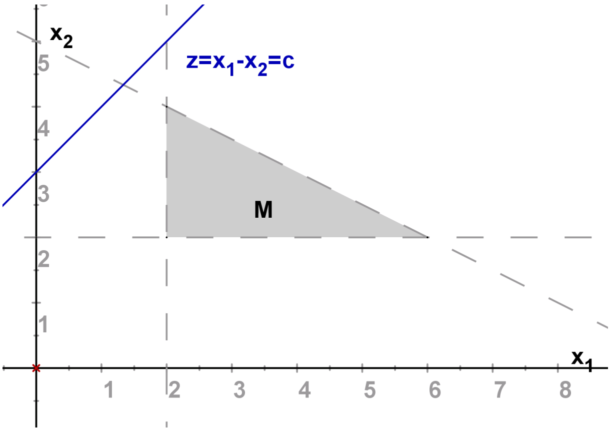
The second phase can be completed as usual:

basic var.	x_1	x_2	x_3	x_4	x_5	β_i
x_1	1	0	-1	0	0	2
x_2	0	1	$\frac{1}{2}$	0	$\frac{1}{10}$	4
x_4	0	0	$\frac{1}{2}$	1	$\frac{1}{10}$	2
z_j	0	0	$-\frac{3}{2}$	0	$-\frac{1}{10}$	-2

We have obtained the optimal table, so we have $x_1 = 2,\ x_2 = 4,\ z_{opt} = -2$.

Two-phase simplex method

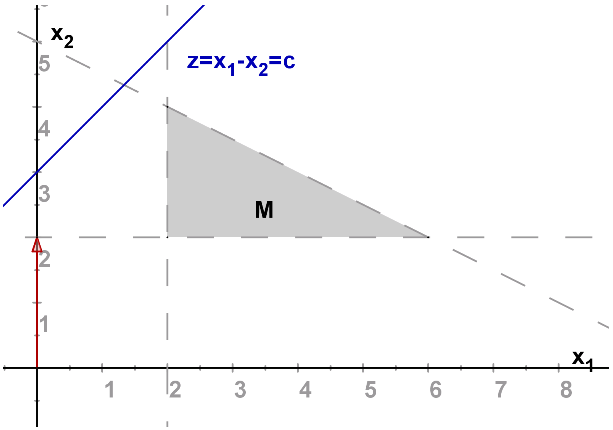
The whole process in a graphical representation:



Feasible region M. Both variables x_1, x_2 are non-basic at the beginning of the first phase, so we start from the origin.

Two-phase simplex method

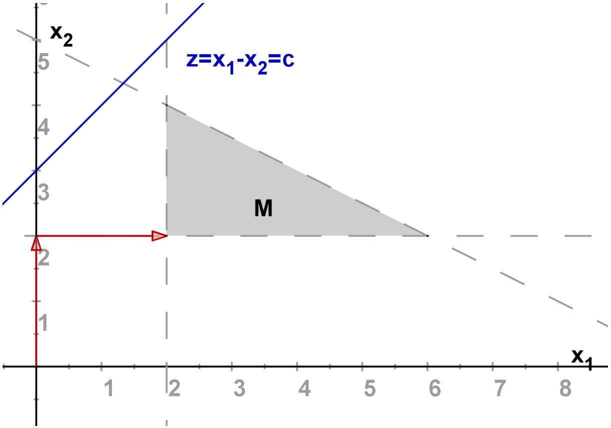
The whole process in a graphical representation:



Variable x_2 becomes basic after the first step with the value $x_2 = 2$.

Two-phase simplex method

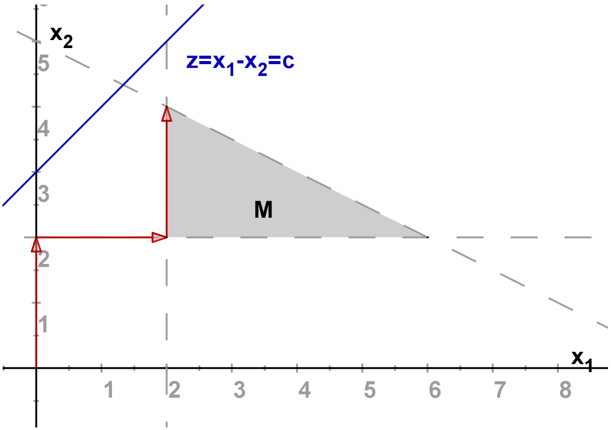
The whole process in a graphical representation:



In the next step also variable x_1 becomes basic with the value $x_1 = 2$, we achieved the feasible region and the second phase can start.

Two-phase simplex method

The whole process in a graphical representation:



Optimum is achieved at the point $[x_1, x_2] = [2, 4]$ after one step.

Pitfalls of the simplex method - degeneration

If the variable is basic and at the same time it is $= 0$ (i.e. the corresponding right side is zero), we say that degeneration has occurred. There is a risk that procedure would get to an infinite loop (after several steps we would return to the same corner point of the feasible set).

Pitfalls of the simplex method - degeneration

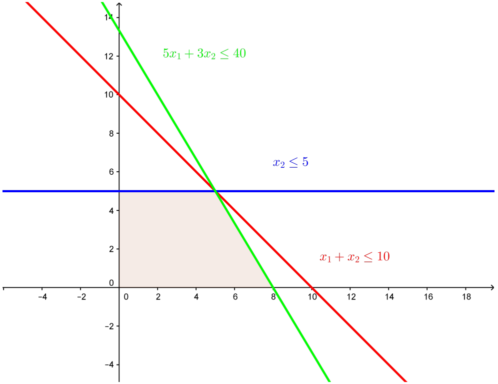
If the variable is basic and at the same time it is $= 0$ (i.e. the corresponding right side is zero), we say that degeneration has occurred. There is a risk that procedure would get to an infinite loop (after several steps we would return to the same corner point of the feasible set). Degeneration is caused by the fact that some constraints are redundant and can be eliminated in several ways:

- By modifying the test of optimality
- Charnes method - adjusts the zero right sides to positive values
- Bland rule - modifies the determination of a key column and a key row

Pitfalls of the simplex method - degeneration

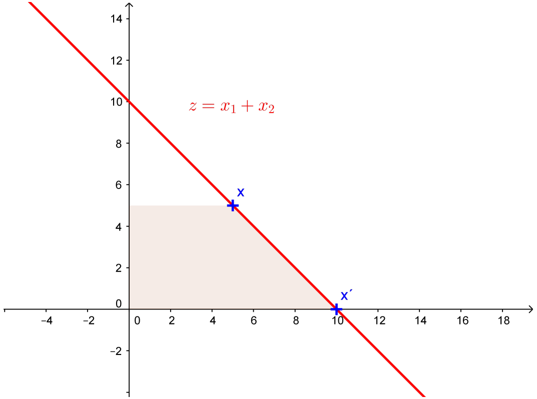
If the variable is basic and at the same time it is $= 0$ (i.e. the corresponding right side is zero), we say that degeneration has occurred. There is a risk that procedure would get to an infinite loop (after several steps we would return to the same corner point of the feasible set). Degeneration is caused by the fact that some constraints are redundant and can be eliminated in several ways:

- By modifying the test of optimality
- Charnes method - adjusts the zero right sides to positive values
- Bland rule - modifies the determination of a key column and a key row



Pitfalls of the simplex method - non-uniqueness of the optimal solution

If, in the output table corresponding to the optimal solution $\mathbf{x}$, a non-basic coefficient in the z row is $= 0$, then if the variable corresponding to this coefficient enters the base, we get $\mathbf{x}'$ which will also be optimal. The optimum does not only occur in both vertices $\mathbf{x}$ and $\mathbf{x}'$, but also at each point of the edge between them (in the case of three optimal neighbours, even the entire wall is optimized by them), etc.

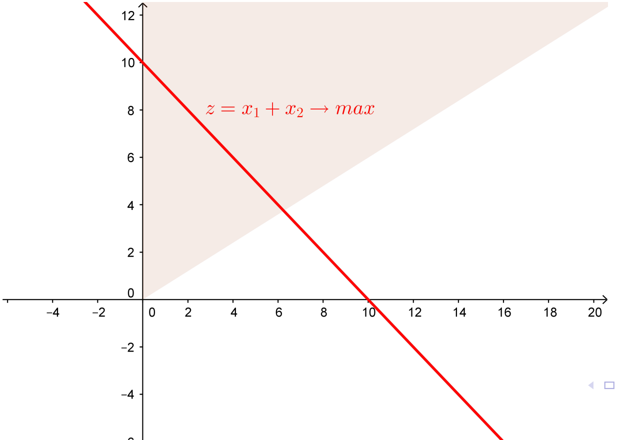


Pitfalls of the simplex method - unbounded feasible region

If all the values in the entering column are ≤ 0 in some iteration step, i.e. no positive change is possible for that column, it means that the set of acceptable solutions is unlimited (in the optimization direction). In practice, this is mostly due to a wrong formulation of the problem (a constraint is missing, or a parameter is not well estimated).

Pitfalls of the simplex method - unbounded feasible region

If all the values in the entering column are ≤ 0 in some iteration step, i.e. no positive change is possible for that column, it means that the set of acceptable solutions is unlimited (in the optimization direction). In practice, this is mostly due to a wrong formulation of the problem (a constraint is missing, or a parameter is not well estimated).

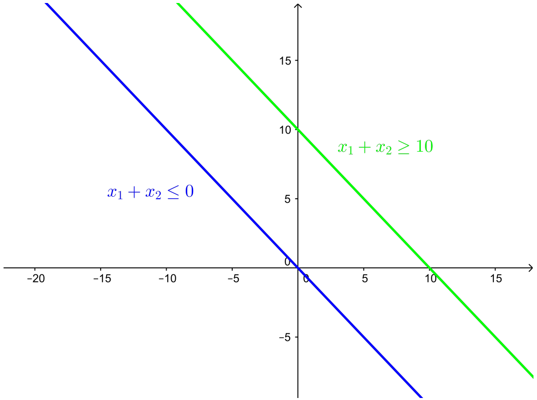


Pitfalls of the simplex method - empty feasible set

The absence of a feasible solution is recognized by the fact that, in the first phase of the two-phase method a positive optimal value is obtained. This is due to the constraints being contradictory. In such situations, we can use goal programming or reformulate some constraints.

Pitfalls of the simplex method - empty feasible set

The absence of a feasible solution is recognized by the fact that, in the first phase of the two-phase method a positive optimal value is obtained. This is due to the constraints being contradictory. In such situations, we can use goal programming or reformulate some constraints.



Duality of LP problems

The original problem can be seen in another way. Suppose we did not process the raw materials but sold them straight away. The question is, when will this direct sale of resources be better than production. This will, of course, depend on the profit from the sale of individual resources - we will use the so-called dual variables denoted by w_i (we have three types of coffee beans, i.e. $i = 1, 2, 3$). We can then formulate the dual problem to the primary problem: What is the minimum profit from the sale of resources when no production is profitable?

Duality of LP problems

The original problem can be seen in another way. Suppose we did not process the raw materials but sold them straight away. The question is, when will this direct sale of resources be better than production. This will, of course, depend on the profit from the sale of individual resources - we will use the so-called dual variables denoted by w_i (we have three types of coffee beans, i.e. $i = 1, 2, 3$). We can then formulate the dual problem to the primary problem: What is the minimum profit from the sale of resources when no production is profitable? Thus, we minimize the profit from the sale of resources $g(\mathbf{w}) = 40w_1 + 60w_2 + 25w_3$ subject to the condition that it is not worthwhile to produce either Mocca or Standard coffee, $0.5w_1 + 0.5w_2 \geq 20$, $0.5w_1 + 0.25w_2 + 0.5w_3 \geq 14$.

Duality of LP problems

The original problem can be seen in another way. Suppose we did not process the raw materials but sold them straight away. The question is, when will this direct sale of resources be better than production. This will, of course, depend on the profit from the sale of individual resources - we will use the so-called dual variables denoted by w_i (we have three types of coffee beans, i.e. $i = 1, 2, 3$). We can then formulate the dual problem to the primary problem: What is the minimum profit from the sale of resources when no production is profitable? Thus, we minimize the profit from the sale of resources $g(\mathbf{w}) = 40w_1 + 60w_2 + 25w_3$ subject to the condition that it is not worthwhile to produce either Mocca or Standard coffee, $0.5w_1 + 0.5w_2 \geq 20,$ $0.5w_1 + 0.25w_2 + 0.5w_3 \geq 14.$ When using the notation introduced above, where $\mathbf{c}$ is the vector of profits from the sale of mixtures, $\mathbf{b} = (40, 60, 25)$ and $\mathbf{A}$ is a structural matrix; we can compare the matrix formulation of the original, called primary, and the dual problem:

primary problem		dual problem	
maximize	$z = \mathbf{c}^\top \cdot \mathbf{x}$	minimize	$g(\mathbf{w}) = \mathbf{b}^\top \cdot \mathbf{w}$
s. t.	$\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$,	s. t.	$\mathbf{A}^\top \cdot \mathbf{w} \geq \mathbf{c}$, $\mathbf{w} \geq 0$

Duality of LP problems

The following rules can be applied for the construction of dual LP problems in a general case:

Maximization problem	$\leftrightarrow$	Minimization problem
primary	$\leftrightarrow$	dual
dual	$\leftrightarrow$	primary
constraint of the type $\leq$	$\leftrightarrow$	non-negative variable
constraint of the type $\geq$	$\leftrightarrow$	non-positive variable
constraint of the type $=$	$\leftrightarrow$	unbounded variable
non-negative variable	$\leftrightarrow$	constraint of the type $\geq$
non-positive variable	$\leftrightarrow$	constraint of the type $\leq$
unbounded variable	$\leftrightarrow$	constraint of the type $=$

Duality of LP problems

The mutual relationship between primary-dual problem can be described in The duality theorem:

If there an optimal solution to one of the primary-dual problems exists,
then there is also an optimal solution to the second problem and
they have the same optimal values !

Duality of LP problems

The mutual relationship between primary-dual problem can be described in The duality theorem:

If there an optimal solution to one of the primary-dual problems exists, then there is also an optimal solution to the second problem and they have the same optimal values !

As a consequence, if one of the problems has no optimal solution, neither does the other problem. It can be shown that if the feasible set is empty for one problem, the other problem is unbounded and vice versa. The next corollary is called The week duality theorem:

The objective value of the maximization problem is always less or equal to the objective value of the minimization problem.

Duality of LP problems

The mutual relationship between primary-dual problem can be described in The duality theorem:

If there an optimal solution to one of the primary-dual problems exists, then there is also an optimal solution to the second problem and they have the same optimal values !

As a consequence, if one of the problems has no optimal solution, neither does the other problem. It can be shown that if the feasible set is empty for one problem, the other problem is unbounded and vice versa. The next corollary is called The week duality theorem:

The objective value of the maximization problem is always less or equal to the objective value of the minimization problem.

We can also formulate a theorem concerning Complementary slackness:

If the k-th variable of the primary problem is positive (non-zero), then the k-th condition of the dual problem is satisfied as an equality. It is said that the k-th constraint is binding.

Sensitivity analysis

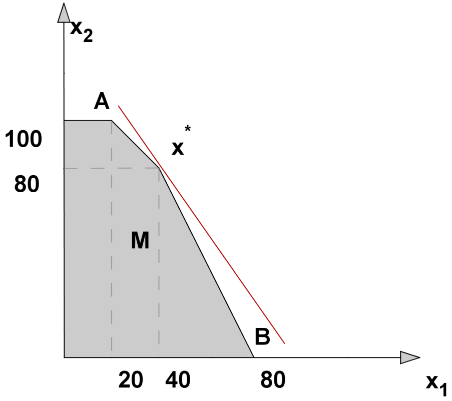
Sensitivity analysis examines the extent to which the possible changes to the input data affect the original optimal solution. In particular, we are interested in the effect of changing a single product profit or changing the resources capacities. This can be determined without recalculation of the whole task. We define the so-called stability intervals for:

- the coefficients c_k of the objective function; we determine the range of values of c_k (while preserving values of the other coefficients), for which the optimal solution would lie in the same point,
- the capacity limits b_i, when we determine the range of b_i (while preserving the values of the other capacities), that would lead to the same set of basic variables, i.e. the same set of active constraints.

  It is important for managerial decision-making to find out what is the effect of changing the capacities on the optimal value of the objective function. This can be determined from the optimal values of the dual variables w_i. These values are called shadow prices and they express the value by which the objective function changes if we increase the capacity of the i-th source b_i by a unit (assuming that this change does not get out of the stability interval)

Sensitivity analysis - the intervals for product prices

The determination of stability intervals is not difficult and it is often an integral part of software outputs. We will show graphical interpretation and derivation of stability intervals for the objective function coefficients in our simple example of optimizing coffee production. The figure shows how much the optimal level curve of the objective function can be tilted not to affect the optimal solution x^*.



Sensitivity analysis - the intervals for product prices

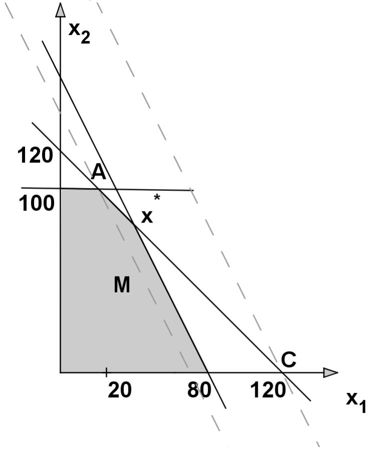
We define the bounds for a tilt so that the level line will cross the points $x^* = [40, 80]$,$A = [20, 100]$ or $x^* = [40, 80]$, $B = [80, 0]$ respectively. Thus the slope q should satisfy the conditions

$$-2 = \frac{80 - 0}{40 - 80} \leq q \leq \frac{80 - 100}{40 - 20} = -1.$$

The slope of the original level line $z = c_1 x_1 + c_2 x_2$ can be expressed as $q = \frac{-c_1}{c_2}$; for our problem we have $c_1 = 20,\ c_2 = 14$. Stability intervals for c_1 can be determined by substituting $q = \frac{-c_1}{14}$ into inequalities: $-2 \leq \frac{-c_1}{14} \leq -1$, i.e. $c_1 \in \langle 14, 28\rangle$. Analogically for c_2 we get stability intervals by substituting $q = \frac{-20}{c_2}$ into inequalities: $-2 \leq \frac{-20}{c_2} \leq -1$ and we get $c_2 \in \langle 10, 20\rangle$.

Sensitivity analysis - the intervals for capacities

Let's show similar graphical derivation of stability intervals for the right-hand sides of constraints. The figure shows how we can move the boundary of the first constraint so that the optimal solution stays at the intersection of the boundary lines of the first and second constraint.



Sensitivity analysis - the intervals for capacities

The original equation of the boundary line for the first constraint was $0,5x_1 + 0,25x_2 = 40$. Its right-hand side b_1 can be changed to move the line at most to the point A or C respectively. By substituting the coordinates of the point $A = [20, 100]$ into the left side of the constraint we get $0,5 \cdot 20 + 0,25 \cdot 100 = 35$ which is the lower bound for b_1.
By substituting the coordinates of the point $C = [120, 0]$ into the left side of the constraint we get $0,5 \cdot 120 + 0,25 \cdot 0 = 60$ which is the upper bound for b_1.

Sensitivity analysis - the intervals for capacities

The original equation of the boundary line for the first constraint was $0,5x_1 + 0,25x_2 = 40$. Its right-hand side b_1 can be changed to move the line at most to the point A or C respectively. By substituting the coordinates of the point $A = [20, 100]$ into the left side of the constraint we get $0,5 \cdot 20 + 0,25 \cdot 100 = 35$ which is the lower bound for b_1.
By substituting the coordinates of the point $C = [120, 0]$ into the left side of the constraint we get $0,5 \cdot 120 + 0,25 \cdot 0 = 60$ which is the upper bound for b_1.
So we obtained the stability interval for $b_1 : \in \langle 35, 60 \rangle$. Similarly, we can obtain stability intervals for other constraints. These intervals are important when the managers decide to purchase additional resources: if the shadow price of the constraint is greater than the purchase price of the resource, it is worthwhile to increase capacity within the stability interval. And how do we find shadow price for b_1? By changing its value to $b_1 + \Delta$ we get a new optimal point as the intersection of lines $0,5x_1 + 0,25x_2 = 40 + \Delta,\ 0,5x_1 + 0,5x_2 = 60$ which is the point $[40 + 4\Delta, 80 - 4\Delta]$. At this point, the objective function attains the value of $z = 20(40 + 4\Delta) + 14(80 - 4\Delta) = 1920 + 24\Delta$. The shadow price is $w_1 = 24$.

Sensitivity analysis - the intervals for capacities

The original equation of the boundary line for the first constraint was $0,5x_1 + 0,25x_2 = 40$. Its right-hand side b_1 can be changed to move the line at most to the point A or C respectively. By substituting the coordinates of the point $A = [20, 100]$ into the left side of the constraint we get $0,5 \cdot 20 + 0,25 \cdot 100 = 35$ which is the lower bound for b_1.
By substituting the coordinates of the point $C = [120, 0]$ into the left side of the constraint we get $0,5 \cdot 120 + 0,25 \cdot 0 = 60$ which is the upper bound for b_1.
So we obtained the stability interval for $b_1 : \in \langle 35, 60 \rangle$. Similarly, we can obtain stability intervals for other constraints. These intervals are important when the managers decide to purchase additional resources: if the shadow price of the constraint is greater than the purchase price of the resource, it is worthwhile to increase capacity within the stability interval. And how do we find shadow price for b_1? By changing its value to $b_1 + \Delta$ we get a new optimal point as the intersection of lines $0,5x_1 + 0,25x_2 = 40 + \Delta,\ 0,5x_1 + 0,5x_2 = 60$ which is the point $[40 + 4\Delta, 80 - 4\Delta]$. At this point, the objective function attains the value of $z = 20(40 + 4\Delta) + 14(80 - 4\Delta) = 1920 + 24\Delta$. The shadow price is $w_1 = 24$.

Shadow prices can be found in the last row of an optimal table under the colum

Special problems of linear programming

Some LP problems have special properties. These properties can relate to the structure of the model, the type of variables, methods of solving, etc. An important class of such special problems is the class of distribution problems. We will explore some representatives of this group, particularly a transportation problem, an assignment problem, and a travelling salesman problem. Other problems (container or multi-stage transportation problem, coverage problem, cutting plans, etc.) can be found in the literature.

Special problems of linear programming

Some LP problems have special properties. These properties can relate to the structure of the model, the type of variables, methods of solving, etc. An important class of such special problems is the class of distribution problems. We will explore some representatives of this group, particularly a transportation problem, an assignment problem, and a travelling salesman problem. Other problems (container or multi-stage transportation problem, coverage problem, cutting plans, etc.) can be found in the literature.

We will also deal with the specifics of integer problems and basic approaches to their solution. Problems in which some variables have range restricted to a set of integer numbers are called integer programming problems. Integer variables usually represent the number of indivisible pieces. In some problems, there are binary variables where values 0 and 1 encode the absence or presence of a certain connection between the specified objects.

Transportation problem

Typically, the transportation problem deals with cost minimizing associated with the delivery of goods from the suppliers to customers. We define m vendors - $V_1, V_2, \dots, V_m$ with limited capacities $a_1, a_2, \dots, a_m$, and n target locations - subscribers $S_1, S_2, \dots, S_n$ with requests $b_1, b_2, \dots, b_n$. Each source-target connection is associated with a value, typically, for example, the cost of transporting the unit of goods. We denote these costs by $c_{ij},\ i = 1, \dots, m,\ j = 1, \dots, n$. The goal is to schedule transport volumes between sources and targets (denoted by $x_{ij},\ i = 1, \dots, m,\ j = 1, \dots, n$) in such way that customer requirements are satisfied and resource capacities are not exceeded.

Transportation problem

Typically, the transportation problem deals with cost minimizing associated with the delivery of goods from the suppliers to customers. We define m vendors - $V_1, V_2, \ldots, V_m$ with limited capacities $a_1, a_2, \ldots, a_m$, and n target locations - subscribers $S_1, S_2, \ldots, S_n$ with requests $b_1, b_2, \ldots, b_n$. Each source-target connection is associated with a value, typically, for example, the cost of transporting the unit of goods. We denote these costs by $c_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n$. The goal is to schedule transport volumes between sources and targets (denoted by $x_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n$) in such way that customer requirements are satisfied and resource capacities are not exceeded. Therefore, the problem consists of $m \cdot n$ variables x_{ij}, for which we minimize the objective function

$z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$ subject to

$\sum_{j=1}^{n} x_{ij} \leq a_i,\ i = 1, \ldots, m,$

$\sum_{i=1}^{m} x_{ij} = b_j,\ j = 1, \ldots, n,$

$x_{ij} \geq 0,\ i = 1, \ldots, m,\ j = 1, \ldots, n$

Both objective function and constraints are linear, so it is a LP problem.

Transportation problem - solution methods

Even though transportation problem is as an LP problem solvable by a standard simplex method, special procedures are more suitable due to a large number of variables and special sparse structure of the matrix (there are many zeros and the other numbers are positioned in a block scheme).

Transportation problem - solution methods

Even though transportation problem is as an LP problem solvable by a standard simplex method, special procedures are more suitable due to a large number of variables and special sparse structure of the matrix (there are many zeros and the other numbers are positioned in a block scheme).

Example: For a problem with 20 sources and 300 customers there is ? variables and ? constraints. The simplex tableau of such problem has 1920000 cells, which means 11 MB of operational memory.

Transportation problem - solution methods

Even though transportation problem is as an LP problem solvable by a standard simplex method, special procedures are more suitable due to a large number of variables and special sparse structure of the matrix (there are many zeros and the other numbers are positioned in a block scheme).

Example: For a problem with 20 sources and 300 customers there is ? variables and ? constraints. The simplex tableau of such problem has 1920000 cells, which means 11 MB of operational memory.

An exact solution can be achieved by Modified distribution method (MODI), which is based on the simplex method. We can also quickly obtain an approximate solution by some heuristics; we will mention three of them: North-western corner method, Index method and Vogel approximate method (VAM).

Transportation problem - solution methods

It is obviously not possible to supply all customers if total demand $\sum_{j=1}^{n} b_j$ exceeds total capacity $\sum_{i=1}^{n} a_i$, then there is no feasible solution to the problem. The problem fulfilling equality $\sum_{j=1}^{n} b_j = \sum_{i=1}^{n} a_i$ is called balanced transportation problem. The balanced problem has a feasible solution even if the capacities constraints are stated as equalities; let's assume such formulation of the problem for the explanation of solution methods.

Transportation problem - solution methods

It is obviously not possible to supply all customers if total demand $\sum_{j=1}^{n} b_j$ exceeds total capacity $\sum_{i=1}^{n} a_i$, then there is no feasible solution to the problem. The problem fulfilling equality $\sum_{j=1}^{n} b_j = \sum_{i=1}^{n} a_i$ is called balanced transportation problem. The balanced problem has a feasible solution even if the capacities constraints are stated as equalities; let's assume such formulation of the problem for the explanation of solution methods.

Unbalanced problem with the surplus of demand can be adjusted by introducing a dummy source with the capacity $\sum_{j=1}^{n} b_j - \sum_{i=1}^{n} a_i$. In the case of supply surplus, there is introduced a dummy customer with the request $\sum_{i=1}^{n} a_i - \sum_{j=1}^{n} b_j$.

Remark: Transportation costs to dummy places are always zero!

Transportation problem - NW corner method

North-western corner method is the simplest way how to obtain a feasible solution of the transportation problem. We begin to match requests with the sources in the upper left corner of the table and proceed to the right (if the request is completed) or down (if the capacity of the source is reached). We finish in the right bottom corner.

Transportation problem - NW corner method

North-western corner method is the simplest way how to obtain a feasible solution of the transportation problem. We begin to match requests with the sources in the upper left corner of the table and proceed to the right (if the request is completed) or down (if the capacity of the source is reached). We finish in the right bottom corner.
We will demonstrate the method on the following problem (M. Plevný, M. Žižka: Modelování a optimalizace v manažerském rozhodování): Find a feasible solution to the transportation problem with four customers S_1, S_2, S_3 and S_4, requesting 3, 6, 4 and 5 units of product and three sources V_1, V_2 and V_3 with capacities 5, 7 and 6 units respectively.

		requests			
	x_{ij}	3	6	4	5
cap. left	5				
	7				
	6				

The S_1 will get 3 units from the first source, then 2 units will be left in the V_1.

Transportation problem - NW corner method

North-western corner method is the simplest way how to obtain a feasible solution of the transportation problem. We begin to match requests with the sources in the upper left corner of the table and proceed to the right (if the request is completed) or down (if the capacity of the source is reached). We finish in the right bottom corner.
We will demonstrate the method on the following problem (M. Plevný, M. Žižka: Modelování a optimalizace v manažerském rozhodování): Find a feasible solution to the transportation problem with four customers S_1, S_2, S_3 and S_4, requesting 3, 6, 4 and 5 units of product and three sources V_1, V_2 and V_3 with capacities 5, 7 and 6 units respectively.

		requests			
	x_{ij}	0	6	4	5
cap. left	2	3			
	7				
	6				

Two units left in V_1 will be transported to S_2 and we move to another source.

Transportation problem - NW corner method

North-western corner method is the simplest way how to obtain a feasible solution of the transportation problem. We begin to match requests with the sources in the upper left corner of the table and proceed to the right (if the request is completed) or down (if the capacity of the source is reached). We finish in the right bottom corner.
We will demonstrate the method on the following problem (M. Plevný, M. Žižka: Modelování a optimalizace v manažerském rozhodování): Find a feasible solution to the transportation problem with four customers S_1, S_2, S_3 and S_4, requesting 3, 6, 4 and 5 units of product and three sources V_1, V_2 and V_3 with capacities 5, 7 and 6 units respectively.

		requests			
	x_{ij}	0	4	4	5
cap. left	0	3	2		
	7				
	6				

Four remaining units requested by the S_2 will be supplied by the V_2, three units will be left.

Transportation problem - NW corner method

North-western corner method is the simplest way how to obtain a feasible solution of the transportation problem. We begin to match requests with the sources in the upper left corner of the table and proceed to the right (if the request is completed) or down (if the capacity of the source is reached). We finish in the right bottom corner.
We will demonstrate the method on the following problem (M. Plevný, M. Žižka: Modelování a optimalizace v manažerském rozhodování): Find a feasible solution to the transportation problem with four customers S_1, S_2, S_3 and S_4, requesting 3, 6, 4 and 5 units of product and three sources V_1, V_2 and V_3 with capacities 5, 7 and 6 units respectively.

		requests			
	x_{ij}	0	0	4	5
cap. left	0	3	2		
	3		4		
	6				

Three remaining units from the V_2 are delivered to the S_3.

Transportation problem - NW corner method

North-western corner method is the simplest way how to obtain a feasible solution of the transportation problem. We begin to match requests with the sources in the upper left corner of the table and proceed to the right (if the request is completed) or down (if the capacity of the source is reached). We finish in the right bottom corner.
We will demonstrate the method on the following problem (M. Plevný, M. Žižka: Modelování a optimalizace v manažerském rozhodování): Find a feasible solution to the transportation problem with four customers S_1, S_2, S_3 and S_4, requesting 3, 6, 4 and 5 units of product and three sources V_1, V_2 and V_3 with capacities 5, 7 and 6 units respectively.

		requests			
	x_{ij}	0	0	1	5
cap. left	0	3	2		
	0		4	3	
	6				

A last unit requested by customer S_3 will be transported from the source V_3.

Transportation problem - NW corner method

North-western corner method is the simplest way how to obtain a feasible solution of the transportation problem. We begin to match requests with the sources in the upper left corner of the table and proceed to the right (if the request is completed) or down (if the capacity of the source is reached). We finish in the right bottom corner.
We will demonstrate the method on the following problem (M. Plevný, M. Žižka: Modelování a optimalizace v manažerském rozhodování): Find a feasible solution to the transportation problem with four customers S_1, S_2, S_3 and S_4, requesting 3, 6, 4 and 5 units of product and three sources V_1, V_2 and V_3 with capacities 5, 7 and 6 units respectively.

		requests			
	x_{ij}	0	0	0	5
cap. left	0	3	2		
	0		4	3	
	5			1	

Remaining five units of the V_3 will be left for the customer S_4.

Transportation problem - NW corner method

North-western corner method is the simplest way how to obtain a feasible solution of the transportation problem. We begin to match requests with the sources in the upper left corner of the table and proceed to the right (if the request is completed) or down (if the capacity of the source is reached). We finish in the right bottom corner.
We will demonstrate the method on the following problem (M. Plevný, M. Žižka: Modelování a optimalizace v manažerském rozhodování): Find a feasible solution to the transportation problem with four customers S_1, S_2, S_3 and S_4, requesting 3, 6, 4 and 5 units of product and three sources V_1, V_2 and V_3 with capacities 5, 7 and 6 units respectively.

		requests			
	x_{ij}	0	0	0	0
cap. left	0	3	2		
	0		4	3	
	0			1	5

Now we have a feasible solution.

Transportation problem - the index method

The index method usually gives a better feasible solution with lower costs than the previous method. It is based on the so-called greedy algorithm, the preferred connections are those with the lowest costs.

Transportation problem - the index method

The index method usually gives a better feasible solution with lower costs than the previous method. It is based on the so-called greedy algorithm, the preferred connections are those with the lowest costs. Let's demonstrate the method on the previously solved problem. We add the table with unit costs of transportation $c_{ij},\ i = 1, \dots 3,\ j = 1, \dots 4$.

		requests			
	x_{ij}	3	6	4	5
cap. left	5				
	7				
	6				

transportation costs

c_{ij}	S_1	S_2	S_3	S_4
V_1	2	1	3	4
V_2	6	2	6	1
V_3	7	3	3	3

The lowest costs are $c_{12} = c_{24} = 1$. We choose for example the first connection, the V_1 can supply to the S_2 at most 5 units. Its capacity will be reached and that is why we don't use the first row of the cost matrix anymore.

Transportation problem - the index method

The index method usually gives a better feasible solution with lower costs than the previous method. It is based on the so-called greedy algorithm, the preferred connections are those with the lowest costs. Let's demonstrate the method on the previously solved problem. We add the table with unit costs of transportation $c_{ij},\ i = 1, \dots 3,\ j = 1, \dots 4$.

		requests			
	x_{ij}	3	1	4	5
cap. left	0		5		
	7				
	6				

transportation costs

c_{ij}	S_1	S_2	S_3	S_4
V_1				
V_2	6	2	6	1
V_3	7	3	3	3

Now the lowest costs are $c_{24} = 1$. Customer S_4 can get from the V_2 at most 5 units. He has no other requests and we ignore the fourth column from now on.

Transportation problem - the index method

The index method usually gives a better feasible solution with lower costs than the previous method. It is based on the so-called greedy algorithm, the preferred connections are those with the lowest costs. Let's demonstrate the method on the previously solved problem. We add the table with unit costs of transportation c_{ij}, $i = 1, \dots 3$, $j = 1, \dots 4$.

		requests			
	x_{ij}	3	1	4	0
cap. left	0		5		
	2				5
	6				

transportation costs

c_{ij}	S_1	S_2	S_3	S_4
V_1				
V_2	6	2	6	
V_3	7	3	3	

The lowest costs are $c_{22} = 2$. The S_2 can take from the V_2 only 1 unit, the second column is completed.

Transportation problem - the index method

The index method usually gives a better feasible solution with lower costs than the previous method. It is based on the so-called greedy algorithm, the preferred connections are those with the lowest costs. Let's demonstrate the method on the previously solved problem. We add the table with unit costs of transportation c_{ij}, $i = 1, \dots 3$, $j = 1, \dots 4$.

		requests			
	x_{ij}	3	0	4	0
cap. left	0		5		
	1		1		5
	6				

transportation costs

c_{ij}	S_1	S_2	S_3	S_4
V_1				
V_2	6		6	
V_3	7		3	

The lowest costs are $c_{33} = 3$. Customer S_3 can get from the V_3 4 units, the third column is completed.

Transportation problem - the index method

The index method usually gives a better feasible solution with lower costs than the previous method. It is based on the so-called greedy algorithm, the preferred connections are those with the lowest costs. Let's demonstrate the method on the previously solved problem. We add the table with unit costs of transportation c_{ij}, $i = 1, \dots 3$, $j = 1, \dots 4$.

		requests			
	x_{ij}	3	0	0	0
cap. left	0		5		
	1		1		5
	2			4	

transportation costs

c_{ij}	S_1	S_2	S_3	S_4
V_1				
V_2	6			
V_3	7			

The lowest costs are $c_{21} = 6$. V_2 can supply S_1 with 1 unit, the third row is completed.

Transportation problem - the index method

The index method usually gives a better feasible solution with lower costs than the previous method. It is based on the so-called greedy algorithm, the preferred connections are those with the lowest costs. Let's demonstrate the method on the previously solved problem. We add the table with unit costs of transportation $c_{ij},\ i = 1, \dots 3,\ j = 1, \dots 4$.

Transportation problem - the index method

The index method usually gives a better feasible solution with lower costs than the previous method. It is based on the so-called greedy algorithm, the preferred connections are those with the lowest costs. Let's demonstrate the method on the previously solved problem. We add the table with unit costs of transportation c_{ij}, $i = 1, \dots 3$, $j = 1, \dots 4$.

		requests			
	x_{ij}	2	0	0	0
cap. left	0		5		
	0	1	1		5
	2			4	

transportation costs

c_{ij}	S_1	S_2	S_3	S_4
V_1				
V_2				
V_3	7			

In the last step we transport from V_3 to S_1 remaining two units.

Transportation problem - the index method

The index method usually gives a better feasible solution with lower costs than the previous method. It is based on the so-called greedy algorithm, the preferred connections are those with the lowest costs. Let's demonstrate the method on the previously solved problem. We add the table with unit costs of transportation $c_{ij},\ i = 1, \dots 3,\ j = 1, \dots 4$.

		requests			
	x_{ij}	0	0	0	0
cap. left	0		5		
	0	1	1		5
	0	2		4	

transportation costs

c_{ij}	S_1	S_2	S_3	S_4
V_1	2	1	3	4
V_2	6	2	6	1
V_3	7	3	3	3

The cost of the transportation will be

$x_{12} \cdot c_{12} + x_{21} \cdot c_{21} + x_{22} \cdot c_{22} + x_{24} \cdot c_{24} + x_{31} \cdot c_{31} + x_{33} \cdot c_{33} = 5 + 6 + 2 + 5 + 14 + 12 = 44.$

Previous method would cost $6 + 2 + 6 + 2 + 5 + 14 + 12 = 47$.

Transportation problem - method VAM

Vogel approximate method is the third heuristics and its advantage lies in considering alternative costs of the cheapest connection. It uses differences (we compute them by subtracting row minima from the second lowest price in every row and column minima from the second cheapest connection in every column). In every step we choose the cheapest connection in the line (row or column) with the highest difference.

Transportation problem - method VAM

Vogel approximate method is the third heuristics and its advantage lies in considering alternative costs of the cheapest connection. It uses differences (we compute them by subtracting row minima from the second lowest price in every row and column minima from the second cheapest connection in every column). In every step we choose the cheapest connection in the line (row or column) with the highest difference.
Lets apply the method on our example. We attach to the cost table row and column differences $d_{i.},\ i = 1, \dots 3$ a $d_{.j},\ j = 1, \dots 4$.

		requests			
	x_{ij}	3	6	4	5
cap. left	5				
	7				
	6				

c_{ij}	S_1	S_2	S_3	S_4	$d_{i.}$
V_1	2	1	3	4	1
V_2	6	2	6	1	1
V_3	7	3	3	3	0
$d_{.j}$	4	1	0	2	

The highest difference is in the first column, where is the lowest cost $c_{11} = 2$.

Transportation problem - method VAM

Vogel approximate method is the third heuristics and its advantage lies in considering alternative costs of the cheapest connection. It uses differences (we compute them by subtracting row minima from the second lowest price in every row and column minima from the second cheapest connection in every column). In every step we choose the cheapest connection in the line (row or column) with the highest difference.
Lets apply the method on our example. We attach to the cost table row and column differences $d_{i.}, \; i = 1, \dots 3$ a $d_{.j}, \; j = 1, \dots 4$.

		requests			
	x_{ij}	0	6	4	5
cap. left	2	3			
	7				
	6				

c_{ij}	S_1	S_2	S_3	S_4	$d_{i.}$
V_1		1	3	4	2
V_2		2	6	1	1
V_3		3	3	3	0
$d_{.j}$		1	0	2	

The highest difference is in the first row, where is the lowest cost $c_{12} = 1$.

Transportation problem - method VAM

Vogel approximate method is the third heuristics and its advantage lies in considering alternative costs of the cheapest connection. It uses differences (we compute them by subtracting row minima from the second lowest price in every row and column minima from the second cheapest connection in every column). In every step we choose the cheapest connection in the line (row or column) with the highest difference.
Lets apply the method on our example. We attach to the cost table row and column differences $d_{i.}, \; i = 1, \ldots 3$ a $d_{.j}, \; j = 1, \ldots 4$.

		requests			
	x_{ij}	0	4	4	5
cap. left	0	3	2		
	7				
	6				

c_{ij}	S_1	S_2	S_3	S_4	$d_{i.}$
V_1					
V_2		2	6	1	1
V_3		3	3	3	0
$d_{.j}$		1	3	2	

The highest difference is in the third column, we choose its lowest cost $c_{33} = 3$.

Transportation problem - method VAM

Vogel approximate method is the third heuristics and its advantage lies in considering alternative costs of the cheapest connection. It uses differences (we compute them by subtracting row minima from the second lowest price in every row and column minima from the second cheapest connection in every column). In every step we choose the cheapest connection in the line (row or column) with the highest difference.
Lets apply the method on our example. We attach to the cost table row and column differences $d_{i.},\ i = 1, \dots 3$ a $d_{.j},\ j = 1, \dots 4$.

		requests			
	x_{ij}	0	4	0	5
cap. left	0	3	2		
	7				
	2			4	

c_{ij}	S_1	S_2	S_3	S_4	$d_{i.}$
V_1					
V_2		2		1	1
V_3		3		3	0
$d_{.j}$		1		2	

The highest difference is in the fourth column, where is the lowest cost $c_{24} = 1$.

Transportation problem - method VAM

Vogel approximate method is the third heuristics and its advantage lies in considering alternative costs of the cheapest connection. It uses differences (we compute them by subtracting row minima from the second lowest price in every row and column minima from the second cheapest connection in every column). In every step we choose the cheapest connection in the line (row or column) with the highest difference.
Lets apply the method on our example. We attach to the cost table row and column differences $d_{i.},\ i = 1, \ldots 3$ a $d_{.j},\ j = 1, \ldots 4$.

		requests			
	x_{ij}	0	4	0	0
cap. left	0	3	2		
	2				5
	2			4	

c_{ij}	S_1	S_2	S_3	S_4	$d_{i.}$
V_1					
V_2		2			
V_3		3			
$d_{.j}$		1			

The highest difference is in the fourth column, where is the lowest cost $c_{22} = 2$.

Transportation problem - method VAM

Vogel approximate method is the third heuristics and its advantage lies in considering alternative costs of the cheapest connection. It uses differences (we compute them by subtracting row minima from the second lowest price in every row and column minima from the second cheapest connection in every column). In every step we choose the cheapest connection in the line (row or column) with the highest difference.
Lets apply the method on our example. We attach to the cost table row and column differences $d_{i.},\ i = 1, \dots 3$ a $d_{.j},\ j = 1, \dots 4$.

		requests			
	x_{ij}	0	2	0	0
cap. left	0	3	2		
	0		2		5
	2			4	

c_{ij}	S_1	S_2	S_3	S_4	$d_{i.}$
V_1					
V_2					
V_3					
$d_{.j}$					

It remains to transport two units from the source V_3 to the customer S_2.

Transportation problem - method VAM

Vogel approximate method is the third heuristics and its advantage lies in considering alternative costs of the cheapest connection. It uses differences (we compute them by subtracting row minima from the second lowest price in every row and column minima from the second cheapest connection in every column). In every step we choose the cheapest connection in the line (row or column) with the highest difference.
Lets apply the method on our example. We attach to the cost table row and column differences $d_{i.}$, $i = 1, \dots 3$ a $d_{.j}$, $j = 1, \dots 4$.

		requests			
	x_{ij}	0	0	0	0
cap. left	0	3	2		
	0		2		5
	0		2	4	

c_{ij}	S_1	S_2	S_3	S_4	$d_{i.}$
V_1	2	1	3	4	1
V_2	6	2	6	1	1
V_3	7	3	3	3	0
$d_{.j}$	4	1	0	2	

Final transport costs are remarkably lower than those obtained by previous methods: $x_{11} \cdot c_{11} + x_{12} \cdot c_{12} + x_{22} \cdot c_{22} + x_{24} \cdot c_{24} + x_{32} \cdot c_{32} + x_{33} \cdot c_{33} = 35$.

Transportation problem - method MODI

The modified distribution method for the solution of balanced problems is based on theorems of duality in linear programming. A dual problem to our task

$z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \to min$ subject to

$\sum_{j=1}^{n} x_{ij} = a_i,\ i = 1, \dots, m, \sum_{i=1}^{m} x_{ij} = b_j,\ j = 1, \dots, n,$

$x_{ij} \geq 0,\ i = 1, \dots, m,\ j = 1, \dots, n$

is formulated by means of dual variables $u_i,\ v_j,\ i = 1, \dots, m,\ j = 1, \dots, n$:

$\sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j \to max$

subject to $u_i + v_j \leq c_{ij},\ i = 1, \dots, m,\ j = 1, \dots, n$

Transportation problem - method MODI

The modified distribution method for the solution of balanced problems is based on theorems of duality in linear programming. A dual problem to our task
$z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \rightarrow min$ subject to
$\sum_{j=1}^{n} x_{ij} = a_i,\ i = 1, \dots, m, \sum_{i=1}^{m} x_{ij} = b_j,\ j = 1, \dots, n,$
$x_{ij} \geq 0,\ i = 1, \dots, m,\ j = 1, \dots, n$
is formulated by means of dual variables $u_i,\ v_j,\ i = 1, \dots, m,\ j = 1, \dots, n$:

$\sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j \rightarrow max$

subject to $u_i + v_j \leq c_{ij},\ i = 1, \dots, m,\ j = 1, \dots, n$

The complementary slackness condition of LP duality implies that all primal basic variables ($x_{ij} > 0$) have the corresponding constraints satisfied as equality. ($u_i + v_j = c_{ij}$). If there is no degeneration, the basis contains exactly $m + n - 1$ variables. So we have the system of $m + n - 1$ equations for $m + n$ dual variables. That means that we can choose one value arbitrarily and solve the system for the rest of the dual variables. By checking the dual feasibility condition

$u_i + v_j - c_{ij} \leq 0,\ i = 1, \dots, m,\ j = 1, \dots, n$

we find out whether is our solution optimal

Transportation problem - method MODI

We can determine the values of dual variables and perform the optimality test right in the table; let's show it on the feasible table obtained by the method VAM:

		requests			
	x_{ij}	3	6	4	5
capacity	5	3	2		
	7		2		5
	6		2	4	

c_{ij}	S_1	S_2	S_3	S_4	u_i
V_1	2	1	3	4	
V_2	6	2	6	1	
V_3	7	3	3	3	
v_j					

We choose such values of u_i, v_j, so that the sums of individual pairs are equal to the numbers in the red cells. We can choose one value arbitrarily, let's say $v_2 = 0$. Remaining values will be computed so that the above mentioned condition is satisfied.

Transportation problem - method MODI

We can determine the values of dual variables and perform the optimality test right in the table; let's show it on the feasible table obtained by the method VAM:

		requests			
	x_{ij}	3	6	4	5
capacity	5	3	2		
	7		2		5
	6		2	4	

c_{ij}	S_1	S_2	S_3	S_4	u_i
V_1	2	1	3	4	1
V_2	6	2	6	1	2
V_3	7	3	3	3	3
v_j	1	0	0	-1	

We choose such values of u_i, v_j, so that the sums of individual pairs are equal to the numbers in the red cells. We can choose one value arbitrarily, lets say $v_2 = 0$. Remaining values will be computed so that the above mentioned condition is satisfied. The table is optimal, the sums of dual variables are not higher than the numbers inside the table.

Transportation problem - method MODI

If we begin from a different starting point, the situation would not be that easy. The test of optimality is not satisfied for a feasible solution determined by the index method:

		requests			
	x_{ij}	3	6	4	5
capacity	5		5		
	7	1	1		5
	6	2		4	

c_{ij}	S_1	S_2	S_3	S_4	u_i
V_1	2	1	3	4	
V_2	6	2	6	1	
V_3	7	3	3	3	
v_j					

Transportation problem - method MODI

If we begin from a different starting point, the situation would not be that easy. The test of optimality is not satisfied for a feasible solution determined by the index method:

		requests			
	x_{ij}	3	6	4	5
capacity	5		5		
	7	1	1		5
	6	2		4	

c_{ij}	S_1	S_2	S_3	S_4	u_i
V_1	2	1	3	4	-1
V_2	6	2	6	1	0
V_3	7	3	3	3	1
v_j	6	2	2	1	

We choose u_i, v_j so that the sums of individual pairs are equal to the numbers in the red cells. We can start by choosing $u_2 = 0$, the rest has to be computed again.

Transportation problem - method MODI

If we begin from a different starting point, the situation would not be that easy. The test of optimality is not satisfied for a feasible solution determined by the index method:

		requests			
	x_{ij}	3	6	4	5
capacity	5		5		
	7	1	1		5
	6	2		4	

c_{ij}	S_1	S_2	S_3	S_4	u_i
V_1	2	1	3	4	-1
V_2	6	2	6	1	0
V_3	7	3	3	3	1
v_j	6	2	2	1	

We can see that the optimality condition is violated in the upper left corner: $u_1 + v_1 = 5 > c_{11} = 2$.

Transportation problem - method MODI

If we begin from a different starting point, the situation would not be that easy. The test of optimality is not satisfied for a feasible solution determined by the index method:

		requests			
	x_{ij}	3	6	4	5
capacity	5	+	5 -		
	7	1 -	1 +		5
	6	2		4	

c_{ij}	S_1	S_2	S_3	S_4	u_i
V_1	2	1	3	4	-1
V_2	6	2	6	1	0
V_3	7	3	3	3	1
v_j	6	2	2	1	

We can see that the optimality condition is violated in the upper left corner: $u_1 + v_1 = 5 > c_{11} = 2$. We can do the iteration step and proceed to another basic solution with a better value of the primal objective function by choosing the entering variable x_{11}, which will replace one of variables x_{12}, x_{22}, x_{21} forming the so-called Dantzig closed loop going through x_{11}. The feasibility of the primal problem must not be violated. We can see that the variable x_{21} can be decreased by 1 and therefore be excluded from the basis. At the same time, we have to increase x_{22} and decrease x_{12} (see the signs + and - in the left table). We get a new basic solution.

Transportation problem - method MODI

If we begin from a different starting point, the situation would not be that easy. The test of optimality is not satisfied for a feasible solution determined by the index method:

		requests			
	x_{ij}	3	6	4	5
capacity	5	1	4		
	7	0	2		5
	6	2		4	

c_{ij}	S_1	S_2	S_3	S_4	u_i
V_1	2	1	3	4	-1
V_2	6	2	6	1	0
V_3	7	3	3	3	4
v_j	3	2	-1	1	

We compute the values of u_i, v_j again and check the optimality. The condition is violated at c_{32} and c_{34}.

Transportation problem - method MODI

If we begin from a different starting point, the situation would not be that easy. The test of optimality is not satisfied for a feasible solution determined by the index method:

		requests			
	x_{ij}	3	6	4	5
capacity	5	1 +	4 -		
	7	0	2		5
	6	2 -	+	4	

c_{ij}	S_1	S_2	S_3	S_4	u_i
V_1	2	1	3	4	-1
V_2	6	2	6	1	0
V_3	7	3	3	3	4
v_j	3	2	-1	1	

We compute the values of u_i, v_j again and check for the optimality. The condition is violated at c_{32} and c_{34}.
We compare the differences $(u_3 + v_2) - c_{32} = 6 - 3 = 3$ and $(u_3 + v_4) - c_{34} = 5 - 3 = 2$. The first one is higher, so by choosing x_{32} as entering variable, we improve the objective value more. We have to complete Dantzig loop by the basic variables x_{31}, x_{11} and x_{12}. The leaving variable will be x_{31}, its 2 units will be moved to x_{32}. We also have to change other variables in the loop - we increase x_{11} by 2 and lower x_{12} by the same amount. We get a new table.

Transportation problem - method MODI

If we begin from a different starting point, the situation would not be that easy. The test of optimality is not satisfied for a feasible solution determined by the index method:

		requests			
	x_{ij}	3	6	4	5
capacity	5	3	2		
	7		2		5
	6	0	2	4	

c_{ij}	S_1	S_2	S_3	S_4	u_i
V_1	2	1	3	4	1
V_2	6	2	6	1	2
V_3	7	3	3	3	3
v_j	1	0	0	-1	

Resulting solution is already optimal as we could see before (when starting from VAM initial solution). The optimal value $z = 35$ cannot be improved. In MODI computations we sometimes face degeneracy, when there are less than $m + n - 1$ non-negative variables. We can get rid of it by letting some zero basis variable x_{ij} have small value $\varepsilon > 0$.

Application area of a transportation problem

Examples of possible applications of a transportation problem are illustrated in the following overview.

activity type	**source**	**destination**
fuel distribution	refinery, warehouse	gas station
mail distribution	sorting centre	post office
distribution of pharmaceuticals	warehouses of distribution companies	pharmacies, hospitals
beet processing	production centres	sugar factories

Exceptionally, we can deal with transportation problems with maximization type of objective function. When?

Assignment problem

In an assignment problem, we try to make pairs of objects from two different groups so that the benefit of the matching is as high as possible. Typically, the task is to assign jobs to machines or projects to employees in order to minimize costs or maximize profit, etc. It is similar to the transportation problem, but the variables are binary and the matching is one to one.

We will demonstrate solution of such a problem on the following example (M. Kavan: Výrobní a provozní management): Find optimal matching of jobs 1, 2, 3 to machines A, B, C, D for production costs given by the table:

		machines			
	c_{ij}	A	B	C	D
jobs	1	15	19	17	12
	2	12	10	15	9
	3	18	14	11	14

We have to choose one number in each row, so that their sum is minimal and their columns are unique.

Assignment problem - the mathematical formulation

We can represent assigning the i-th job to the j-th machine by the equality $x_{ij} = 1$ (zero otherwise). If there are more jobs than machines ($m > n$), the problem is unsolvable. In the case $n > m$ we can balance the problem by introducing dummy jobs with zero production costs so that $m = n$. Let's assume that the problem is balanced.
Mathematical formulation of the assignment problem assumes binary variables $x_{ij},\ i, j = 1 \dots n$ and conditions that all row and column sums are equal to 1. If we denote production cost by $c_{ij},\ i, j = 1 \dots n$, we get the following formulation:
Minimize objective function

$$z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1, \dots, n,$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1, \dots, n,$$

$$x_{ij} \in \{0,\ 1\},\ i = 1,\ j = 1, \dots, n$$

Assignment problem - solution methods

Assignment problem can be solved by the Hungarian method. We perform the row and column reduction of the cost table in order to highlight row and column minima (they are zeros after the reduction). If there exist n independent zeros (not sharing any row or column), the table is optimal. The zeros are independent if it is not possible to cover them all by less than n vertical or horizontal lines. Let's demonstrate the method on the previous example.

Assignment problem - solution methods

Assignment problem can be solved by the Hungarian method. We perform the row and column reduction of the cost table in order to highlight row and column minima (they are zeros after the reduction). If there exist *n* independent zeros (not sharing any row or column), the table is optimal. The zeros are independent if it is not possible to cover them all by less than *n* vertical or horizontal lines. Let's demonstrate the method on the previous example. We have to balance the problem because there are more machines than jobs.

		machines			
	c_{ij}	A	B	C	D
jobs	1	15	19	17	12
	2	12	10	15	9
	3	18	14	11	14

We introduce one dummy job with zero production costs.

Assignment problem - solution methods

Assignment problem can be solved by the Hungarian method. We perform the row and column reduction of the cost table in order to highlight row and column minima (they are zeros after the reduction). If there exist *n* independent zeros (not sharing any row or column), the table is optimal. The zeros are independent if it is not possible to cover them all by less than *n* vertical or horizontal lines. Let's demonstrate the method on the previous example. We have to balance the problem because there are more machines than jobs.

		machines			
	c_{ij}	A	B	C	D
jobs	1	15	19	17	12
	2	12	10	15	9
	3	18	14	11	14
	4	0	0	0	0

We do primary reduction by subtracting row minima, there will be at least one zero in every row.

Assignment problem - solution methods

Assignment problem can be solved by the Hungarian method. We perform the row and column reduction of the cost table in order to highlight row and column minima (they are zeros after the reduction). If there exist n independent zeros (not sharing any row or column), the table is optimal. The zeros are independent if it is not possible to cover them all by less than n vertical or horizontal lines. Let's demonstrate the method on the previous example. We have to balance the problem because there are more machines than jobs.

		machines			
	c_{ij}	A	B	C	D
jobs	1	3	7	5	0
	2	3	1	6	0
	3	7	3	0	3
	4	0	0	0	0

The same should be done for columns, but it is not necessary due to a dummy job - there is already a zero in every column.

Assignment problem - solution methods

Assignment problem can be solved by the Hungarian method. We perform the row and column reduction of the cost table in order to highlight row and column minima (they are zeros after the reduction). If there exist n independent zeros (not sharing any row or column), the table is optimal. The zeros are independent if it is not possible to cover them all by less than n vertical or horizontal lines. Let's demonstrate the method on the previous example. We have to balance the problem because there are more machines than jobs.

		machines			
	c_{ij}	A	B	C	D
jobs	1	3	7	5	0
	2	3	1	6	0
	3	7	3	0	3
	4	0	0	0	0

We need only three lines to cover all zeros, so the table is not optimal. In the iteration step we add the smallest uncovered element to the numbers in the intersections of covering lines and at the same time, we subtract it from every uncovered element (the elements covered by only one line remain unchanged).

Assignment problem - solution methods

Assignment problem can be solved by the Hungarian method. We perform the row and column reduction of the cost table in order to highlight row and column minima (they are zeros after the reduction). If there exist *n* independent zeros (not sharing any row or column), the table is optimal. The zeros are independent if it is not possible to cover them all by less than *n* vertical or horizontal lines. Let's demonstrate the method on the previous example. We have to balance the problem because there are more machines than jobs.

		machines			
	c_{ij}	A	B	C	D
jobs	1	2	6	5	0
	2	2	0	6	0
	3	6	2	0	3
	4	0	0	1	1

After the secondary reduction we have a new uncovered zero and there are no zeros covered by two lines. We have to construct a new covering.

Assignment problem - solution methods

Assignment problem can be solved by the Hungarian method. We perform the row and column reduction of the cost table in order to highlight row and column minima (they are zeros after the reduction). If there exist n independent zeros (not sharing any row or column), the table is optimal. The zeros are independent if it is not possible to cover them all by less than n vertical or horizontal lines. Let's demonstrate the method on the previous example. We have to balance the problem because there are more machines than jobs.

		machines			
	c_{ij}	A	B	C	D
jobs	1	2	6	5	0
	2	2	0	6	0
	3	6	2	0	3
	4	0	0	1	1

Let's check the optimality of a new table. It is not possible to cover all zeros by less than four lines, we have an optimal solution and the algorithm terminates (otherwise we would perform the secondary reduction again).

Assignment problem - solution methods

Assignment problem can be solved by the Hungarian method. We perform the row and column reduction of the cost table in order to highlight row and column minima (they are zeros after the reduction). If there exist *n* independent zeros (not sharing any row or column), the table is optimal. The zeros are independent if it is not possible to cover them all by less than *n* vertical or horizontal lines. Let's demonstrate the method on the previous example. We have to balance the problem because there are more machines than jobs.

		machines			
	c_{ij}	A	B	C	D
jobs	1	2	6	5	0
	2	2	0	6	0
	3	6	2	0	3
	4	0	0	1	1

We can find independent zeros among those covered by only one line. Their positions determine the optimal matching of jobs and machines: 1-D, 2-B, 3-C. A dummy job is assigned to machine A, so it is not used.

Assignment problem - solution methods

Assignment problem can be solved by the Hungarian method. We perform the row and column reduction of the cost table in order to highlight row and column minima (they are zeros after the reduction). If there exist *n* independent zeros (not sharing any row or column), the table is optimal. The zeros are independent if it is not possible to cover them all by less than *n* vertical or horizontal lines. Let's demonstrate the method on the previous example. We have to balance the problem because there are more machines than jobs.

		machines			
	c_{ij}	A	B	C	D
jobs	1	15	19	17	12
	2	12	10	15	9
	3	18	14	11	14

The optimal solution can be highlighted in an original table. The total costs of matching are $12 + 10 + 11 = 33$. The solution is not always unique - covering lines and independent zeros can be chosen in different ways - but all optimal solution have the same objective value.

travelling salesman problem (TSP)

The TSP is a special case of the vehicle routing problem. The objective is to start from an initial place (labelled by A_1), visit each of the places $A_2, \ldots, A_n$ exactly once and return to the place A_1 so that the total route is as short as possible. There is a lot of real TSP applications - e.g. the distribution of products, waste collection, etc.

travelling salesman problem (TSP)

The TSP is a special case of the vehicle routing problem. The objective is to start from an initial place (labelled by A_1), visit each of the places $A_2, \dots, A_n$ exactly once and return to the place A_1 so that the total route is as short as possible. There is a lot of real TSP applications - e.g. the distribution of products, waste collection, etc.

The mathematical model of TSP is similar to an assignment problem, it deals with binary variables x_{ij}, that have value 1 if the connection $A_i,\ A_j$ is on the route and the value 0 if it is not. The route goes through every place just once, so we can match the starting and ending points of each route segment in the table with the constraint of row and column sums equal to 1. However, TSP is much more difficult, because we have to add other constraints preventing the route from decomposing to separate cycles (see example).

travelling salesman problem (TSP)

The TSP is a special case of the vehicle routing problem. The objective is to start from an initial place (labelled by A_1), visit each of the places $A_2, \ldots, A_n$ exactly once and return to the place A_1 so that the total route is as short as possible. There is a lot of real TSP applications - e.g. the distribution of products, waste collection, etc.

The mathematical model of TSP is similar to an assignment problem, it deals with binary variables x_{ij}, that have value 1 if the connection $A_i,\ A_j$ is on the route and the value 0 if it is not. The route goes through every place just once, so we can match the starting and ending points of each route segment in the table with the constraint of row and column sums equal to 1. However, TSP is much more difficult, because we have to add other constraints preventing the route from decomposing to separate cycles (see example).

For a large n it is practically impossible to use standard optimization algorithms (we can hardly find solutions for problems up to $n = 30$ nodes using common computers).

travelling salesman problem - feasibility of the solution

We can demonstrate the difference between the TSP and an assignment problem on the following example:
If we try to find any closed route going through cities $A_1, \ldots, A_5$ (without taking into account the costs), we have to choose a pair of leaving and entering city for every segment of the route. Every city is just once leaving and just once entering. Let's depict some solution in the table:

	A_1	A_2	A_3	A_4	A_5
A_1	0	0	1	0	0
A_2	0	0	0	1	0
A_3	0	1	0	0	0
A_4	0	0	0	0	1
A_5	1	0	0	0	0

It is a feasible solution of TSP, because it represents the route $A_1 - A_3 - A_2 - A_4 - A_5 - A_1$.

travelling salesman problem - feasibility of the solution

We can demonstrate the difference between the TSP and an assignment problem on the following example:
If we try to find any closed route going through cities $A_1, \ldots, A_5$ (without taking into account the costs), we have to choose a pair of leaving and entering city for every segment of the route. Every city is just once leaving and just once entering. Let's depict some solution in the table:

	A_1	A_2	A_3	A_4	A_5
A_1	0	1	0	0	0
A_2	0	0	1	0	0
A_3	1	0	0	0	0
A_4	0	0	0	0	1
A_5	0	0	0	1	0

This is not a feasible solution of TSP, because it represents two cycles $A_1 - A_2 - A_3 - A_1$ and $A_4 - A_5 - A_4$.

travelling salesman problem - approximate solution

There are many heuristics for finding an approximate solution to TSP, we will show the greedy algorithm of the nearest neighbour on an example from the book J. Pelikán, V. Chýna: Kvantitativní management.
The bakery in Kralupy supplies grocery shops in surrounding cities every day. Find the order of stops of bakery car minimizing the total length of the route. Distances between the cities are given by the table:

travelling salesman problem - approximate solution

There are many heuristics for finding an approximate solution to TSP, we will show the greedy algorithm of the nearest neighbour on an example from the book J. Pelikán, V. Chýna: Kvantitativní management.
The bakery in Kralupy supplies grocery shops in surrounding cities every day. Find the order of stops of bakery car minimizing the total length of the route. Distances between the cities are given by the table:

	Kralupy	Veltrusy	Slaný	Velvary	Zlonice	Vraný	Bříza
Kralupy	0	4	16	8	18	25	17
Veltrusy	4	0	20	12	22	28	13
Slaný	16	20	0	12	7	14	17
Velvary	8	12	12	0	10	17	10
Zlonice	18	22	7	10	0	7	10
Vraný	25	28	14	17	7	0	15
Bříza	17	13	17	10	10	15	0

travelling salesman problem - approximate solution

There are many heuristics for finding an approximate solution to TSP, we will show the greedy algorithm of the nearest neighbour on an example from the book J. Pelikán, V. Chýna: Kvantitativní management.
The bakery in Kralupy supplies grocery shops in surrounding cities every day. Find the order of stops of bakery car minimizing the total length of the route. Distances between the cities are given by the table:

	Kralupy	Veltrusy	Slaný	Velvary	Zlonice	Vraný	Bříza
Kralupy	0	4	16	8	18	25	17
Veltrusy	4	0	20	12	22	28	13
Slaný	16	20	0	12	7	14	17
Velvary	8	12	12	0	10	17	10
Zlonice	18	22	7	10	0	7	10
Vraný	25	28	14	17	7	0	15
Bříza	17	13	17	10	10	15	0

The nearest neighbour algorithm chooses the first segment of the route according to the shortest distance from the starting point. Next, we continue to the next closest place and control for not closing the loop before we go through all the places. We will get the route Kralupy - Veltrusy - Velvary - Zlonice - Slaný - Vraný - Bříza - Kralupy with the total distance 4+12+10+7+

travelling salesman problem - approximate solution

There are many heuristics for finding an approximate solution to TSP, we will show the greedy algorithm of the nearest neighbour on an example from the book J. Pelikán, V. Chýna: Kvantitativní management.
The bakery in Kralupy supplies grocery shops in surrounding cities every day. Find the order of stops of bakery car minimizing the total length of the route. Distances between the cities are given by the table:

	Kralupy	Veltrusy	Slaný	Velvary	Zlonice	Vraný	Bříza
Kralupy	0	4	16	8	18	25	17
Veltrusy	4	0	20	12	22	28	13
Slaný	16	20	0	12	7	14	17
Velvary	8	12	12	0	10	17	10
Zlonice	18	22	7	10	0	7	10
Vraný	25	28	14	17	7	0	15
Bříza	17	13	17	10	10	15	0

The algorithm can give a better solution if we try to start the route from another city. Initialization from Veltrusy leads to the route Veltrusy - Kralupy - Velvary - Zlonice - Slaný - Vraný - Bříza - Veltrusy with the total distance 4+8+10+ 7+ 14+15 +13 = 71 km.

travelling salesman problem - approximate solution

There are many heuristics for finding an approximate solution to TSP, we will show the greedy algorithm of the nearest neighbour on an example from the book J. Pelikán, V. Chýna: Kvantitativní management.
The bakery in Kralupy supplies grocery shops in surrounding cities every day. Find the order of stops of bakery car minimizing the total length of the route. Distances between the cities are given by the table:

	Kralupy	Veltrusy	Slaný	Velvary	Zlonice	Vraný	Bříza
Kralupy	0	4	16	8	18	25	17
Veltrusy	4	0	20	12	22	28	13
Slaný	16	20	0	12	7	14	17
Velvary	8	12	12	0	10	17	10
Zlonice	18	22	7	10	0	7	10
Vraný	25	28	14	17	7	0	15
Bříza	17	13	17	10	10	15	0

Initialization from Veltrusy leads to the route Vraný -Zlonice - Slaný - Velvary - Kralupy - Veltrusy - Bříza -Vraný with the total distance 7+7+12+8+4+13 +15 = 66 km. It can be shown that this is even the shortest route.

travelling salesman problem - approximate solution

Another heuristics for TSP is the method of advantage numbers. It is based on comparing the length of the route $A_1 - A_i - A_j - A_1$ and the sum of distances $A_1 - A_i - A_1$, $A_1 - A_j - A_1$. The advantage number S_{ij} is defined as their difference, it expresses the advantage of connecting A_i with A_j comparing to two separate journeys from A_1: $S_{ij} = c_{1i} + c_{1j} - c_{ij}$.

We can compute the advantage matrix for a bakery example:

travelling salesman problem - approximate solution

Another heuristics for TSP is the method of advantage numbers. It is based on comparing the length of the route $A_1 - A_i - A_j - A_1$ and the sum of distances $A_1 - A_i - A_1$, $A_1 - A_j - A_1$. The advantage number S_{ij} is defined as their difference, it expresses the advantage of connecting A_i with A_j comparing to two separate journeys from A_1: $S_{ij} = c_{1i} + c_{1j} - c_{ij}$.

We can compute the advantage matrix for a bakery example:

	Veltrusy	Slaný	Velvary	Zlonice	Vraný	Bříza
Veltrusy	X	0	0	0	1	8
Slaný	0	X	12	27	27	16
Velvary	0	12	X	16	16	15
Zlonice	0	27	16	X	36	25
Vraný	1	27	16	36	X	27
Bříza	8	16	15	25	27	X

The route is constructed sequentially starting from the largest advantage numbers, controlling for not completing the circle too early. We got the route Slaný - Zlonice - Vraný - Bříza - Velvary - Veltrusy. Kralupy are at the same time starting and ending point, so the total distance is 16+7+7+15+10+12+4 = 71 km.

Integer programming

Special cases of LP include also integer programming problems. They are standard LP problems with an additional condition that some variables or all variables are integers (the first case is referred to as mixed integer programming). These conditions usually follow directly from the economic model, if the variables represent pieces of goods, number of employees, number of repeating certain operation, etc. The variables may often have range restricted to $\{0, 1\}$ if they represent a decision, status, presence of st., etc. We refer to those problems as binary programming problems

Integer programming

Special cases of LP include also integer programming problems. They are standard LP problems with an additional condition that some variables or all variables are integers (the first case is referred to as mixed integer programming). These conditions usually follow directly from the economic model, if the variables represent pieces of goods, number of employees, number of repeating certain operation, etc. The variables may often have range restricted to $\{0, 1\}$ if they represent a decision, status, presence of st., etc. We refer to those problems as binary programming problems and among their typical representatives can be found an assignment problem, the TSP or the so-called knapsack problem:

Example: There is *n* indivisible items of different value and a knapsack of limited capacity. The objective is to choose items of maximal total value, which do not exceed the capacity of the knapsack.

Integer programming

Special cases of LP include also integer programming problems. They are standard LP problems with an additional condition that some variables or all variables are integers (the first case is referred to as mixed integer programming). These conditions usually follow directly from the economic model, if the variables represent pieces of goods, number of employees, number of repeating certain operation, etc. The variables may often have range restricted to $\{0,\ 1\}$ if they represent a decision, status, presence of st., etc. We refer to those problems as binary programming problems and among their typical representatives can be found an assignment problem, the TSP or the so-called knapsack problem:

Example: There is *n* indivisible items of different value and a knapsack of limited capacity. The objective is to choose items of maximal total value, which do not exceed the capacity of the knapsack.
Remark: Without the indivisibility assumption the problem would not be a binary problem and its solution would be trivial: order items according to their value/weight ratio and load the knapsack in that order until the capacity is not full. If the last item cannot be loaded whole, cut it and load only part of it, so that the weight limit is achieved.

Example of integer programming

Example: Textile company Style, l.t.d. is the producer of men's fashion. Let's denote by x_1 its week production of a suit "Marcel"and by x_2 the same for a suit "Filip". The time needed for producing one suit of type "Marcel"is 10 hours and material costs are 400 CZK. The production of suit "Filip"takes 20 hours and the material costs are 300 CZK. The company has been offered to supply fashion for an international fair which takes place next week. Expected profit from selling respective suits is 20 EUR for "Marcel"and 30 EUR for "Filip". Design an optimal production plan, if there are one full-time employee and one half-time employee (that means 60 hours) available and on the stock, there is a material of the value 1300 CZK.

Example of integer programming

Example: Textile company Style, l.t.d. is the producer of men's fashion. Let's denote by x_1 its week production of a suit "Marcel"and by x_2 the same for a suit "Filip". The time needed for producing one suit of type "Marcel"is 10 hours and material costs are 400 CZK. The production of suit "Filip"takes 20 hours and the material costs are 300 CZK. The company has been offered to supply fashion for an international fair which takes place next week. Expected profit from selling respective suits is 20 EUR for "Marcel"and 30 EUR for "Filip". Design an optimal production plan, if there are one full-time employee and one half-time employee (that means 60 hours) available and on the stock, there is a material of the value 1300 CZK.

Mathematical formulation of the problem is following: Maximize objective function

$z = 20x_1 + 30x_2$

subject to

$400x_1 + 300x_2 \leq 1300$

$10x_1 + 20x_2 \leq 60$

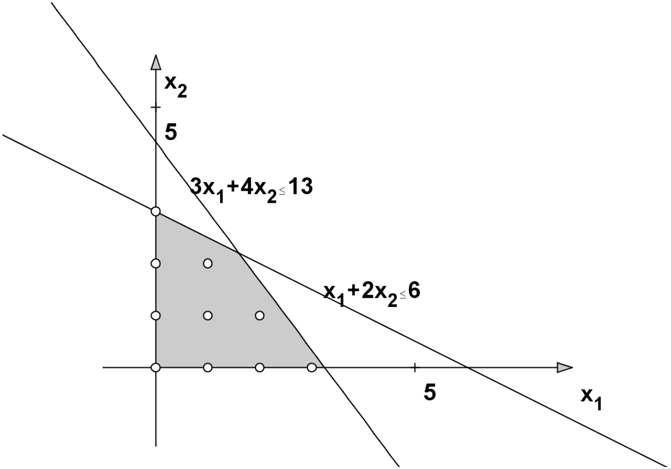
$x_1,\ x_2 \in \{0, 1, 2, 3, \dots\}$

Integer programming- graphical representation

The model contains only two variables, so it is possible to solve it graphically.

Integer programming- graphical representation

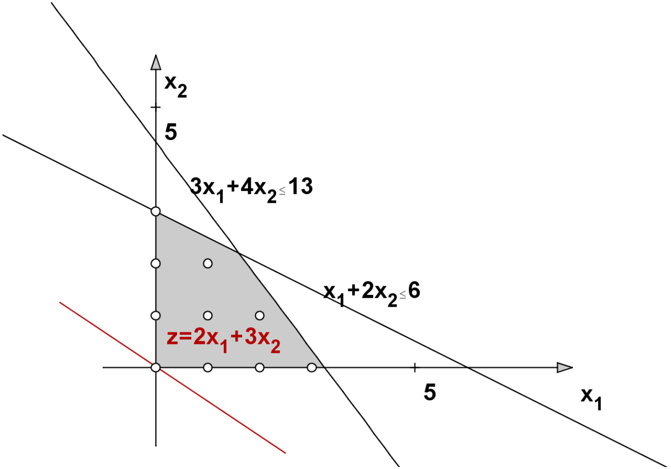
The model contains only two variables, so it is possible to solve it graphically.



The feasible set consists of points with integer coordinates lying in the gray polygon.

Integer programming- graphical representation

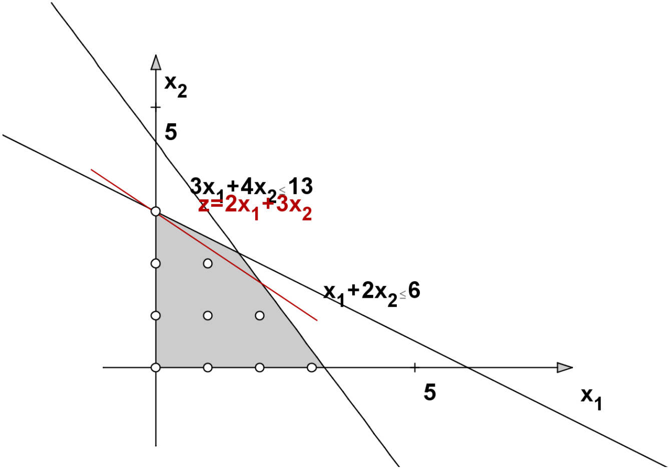
The model contains only two variables, so it is possible to solve it graphically.



The zero level curve of the profit function.

Integer programming- graphical representation

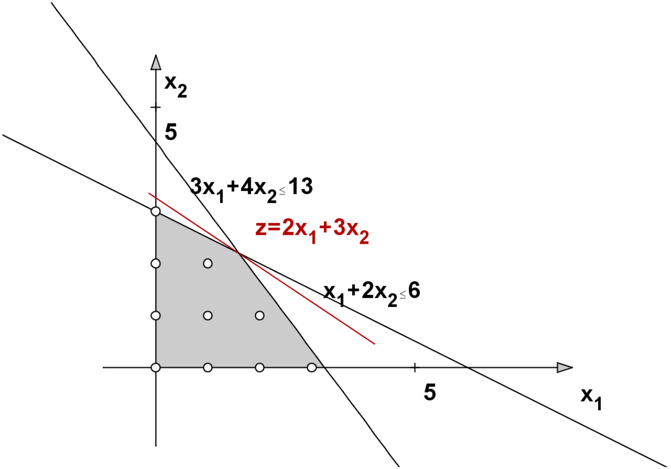
The model contains only two variables, so it is possible to solve it graphically.



The highest level curve goes through the point $[0, 3]$ which is the optimal solution x^*. We gain $3 \cdot 30 = 90$ EUR.

Integer programming- graphical representation

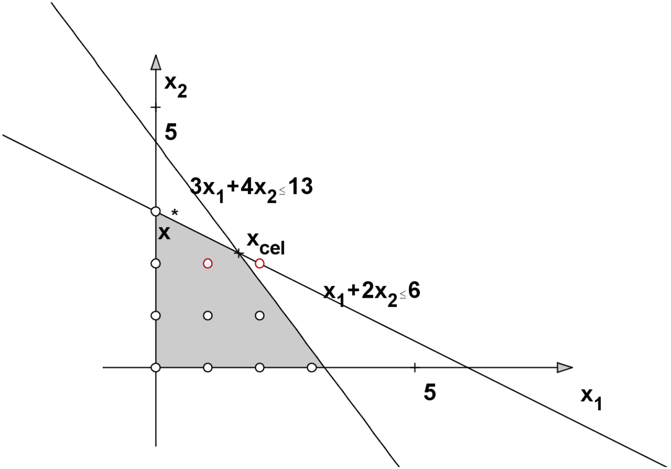
The model contains only two variables, so it is possible to solve it graphically.



Optimal solution of the relaxed problem (loosing the assumption on integer range of variables) is $x_{cel} = [1,6; 2,2]$, corresponding optimal profit is $z_{cel} = 98$ EUR.

Integer programming- graphical representation

The model contains only two variables, so it is possible to solve it graphically.



By rounding coordinates of $x_{cel} = [1,6; 2,2]$ we get the point $[2,2]$ that is not feasible. If we round the first coordinate down to $[1,2]$ instead, we get a feasible point, but its profit is only $20 + 2 \cdot 30 = 80$ EUR, which is significantly less than 90 EUR for x^*.

Integer programming- methods

We could see on a previous example that the intuitive approach of neglecting the integer type of variables in the model (this is called relaxed model) and rounding the relaxed solution is not always suitable. That is why different special procedures designed for integer problems are used. However, their computations are more complicated and time-consuming, so common computers can collapse even by solving relatively small tasks (tens of variables and constraints).

Integer programming- methods

We could see on a previous example that the intuitive approach of neglecting the integer type of variables in the model (this is called relaxed model) and rounding the relaxed solution is not always suitable. That is why different special procedures designed for integer problems are used. However, their computations are more complicated and time-consuming, so common computers can collapse even by solving relatively small tasks (tens of variables and constraints).

Cutting Hyperplane Methods (such as Gomory algorithm) search for the optimum of the relaxed problem and then add another restriction that cuts off the non-integer optimal point. These methods are quite old and less used nowadays.

pause Combinatorial methods are based on an effective scan of the feasible set, which, for purely integer problems, usually contains finite but great number of elements. In program systems, the branch and bound method (ie B & B) is most often used.

pause Special methods are used for problems with a special structure, including different heuristics for a TSP or the Hungarian method for the

Integer programming- method B & B

The feasible set is sequentially divided into smaller parts (branching), where we find the upper bound (the lower bound when minimizing) of the objective function (bounding). This allows estimating subsets where the optimum probably could be and subsets where it cannot occur.

Integer programming- method B & B

The feasible set is sequentially divided into smaller parts (branching), where we find the upper bound (the lower bound when minimizing) of the objective function (bounding). This allows estimating subsets where the optimum probably could be and subsets where it cannot occur.

Branching is done by solving a relaxed linear program - its feasible set is labelled by M_0. If the optimal solution x^0 of this relaxation is an integer, it is also a solution to the original integer problem. Else we choose an index i, for which i-th coordinate of the solution x_i^0 is not an integer and we construct additional constraints $x_i \leq a$ or $x_i \geq b$, where $a,\ b$ are integers surrounding x_i. By that is the set M_0 divided into subsets M_1 and M_2.

Integer programming- method B & B

The feasible set is sequentially divided into smaller parts (branching), where we find the upper bound (the lower bound when minimizing) of the objective function (bounding). This allows estimating subsets where the optimum probably could be and subsets where it cannot occur.

Branching is done by solving a relaxed linear program - its feasible set is labelled by M_0. If the optimal solution x^0 of this relaxation is an integer, it is also a solution to the original integer problem. Else we choose an index i, for which i-th coordinate of the solution x_i^0 is not an integer and we construct additional constraints $x_i \leq a$ or $x_i \geq b$, where $a, \ b$ are integers surrounding x_i. By that is the set M_0 divided into subsets M_1 and M_2.

On each of subsets M_1 and M_2 we find optima x^1 and x^2 and compute their objective values z^1 and z^2. By rounding downward these values we get upper bounds of objective functions on the sets M_1 and M_2. The whole algorithm proceeds until its termination in one of the following cases:

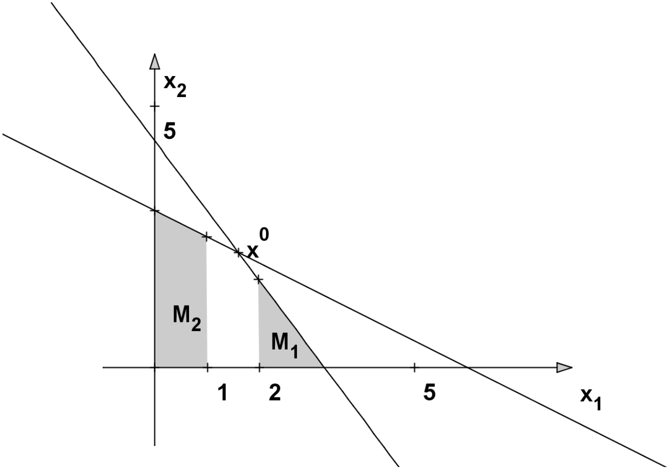
- integer solution is found in a branch or
- there is no feasible solution in a branch or
- non-integer optimal solution is found in a branch whose objective value is worse than the value of an integer point in a different branch.

Integer programming - method B & B

Lets demonstrate the B & B method for a textile company example.

Integer programming - method B & B

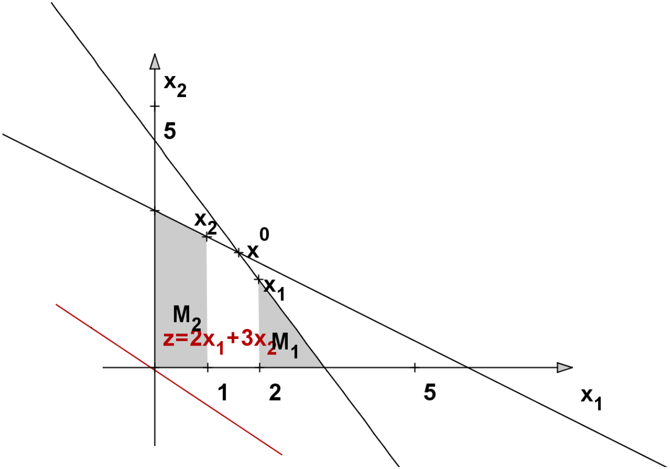
Lets demonstrate the B & B method for a textile company example.



We have a solution of the relaxed problem, $x^0 = x_{cel} = [1,6;2,2]$. Its first coordinate is not integer, so we introduce the constraints $x_1 \geq 2$ and $x_1 \leq 1$ respectively, thus dividing the feasible set into the M_1 and the M_2.

Integer programming - method B & B

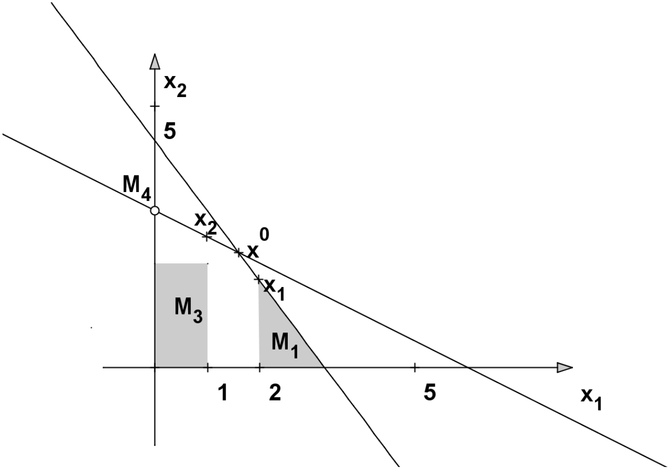
Lets demonstrate the B & B method for a textile company example.



On the feasible set M_1, the optimum is achieved at $x^1 = [2; \frac{5}{3}]$ and on the M_2, it is $x^2 = [1; \frac{5}{2}]$. Their values are $z^1 = 90$ and $z^2 = 95$, which determines the upper bounds for M_1 and M_2.

Integer programming - method B & B

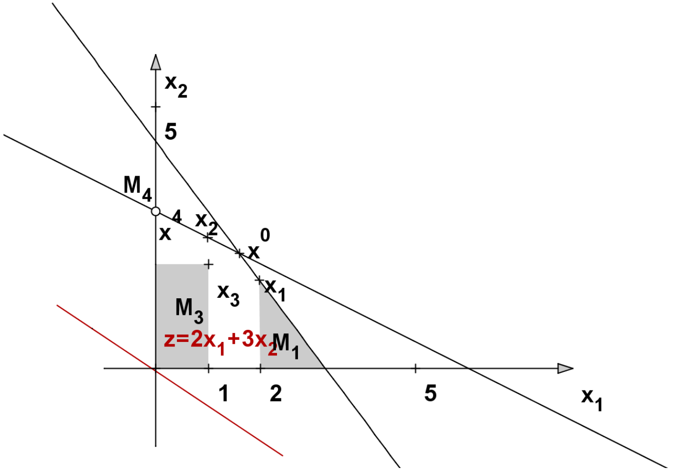
Lets demonstrate the B & B method for a textile company example.



The upper bound of M_2 is higher, so we continue by dividing this set. The second coordinate of x^2 is equal to $\frac{5}{2}$, so we introduce the constraint $x_2 \leq 2$ or $x_2 \geq 3$ into the model, thus we get new feasible sets M_3 and M_4 (which consists of only one element).

Integer programming - method B & B

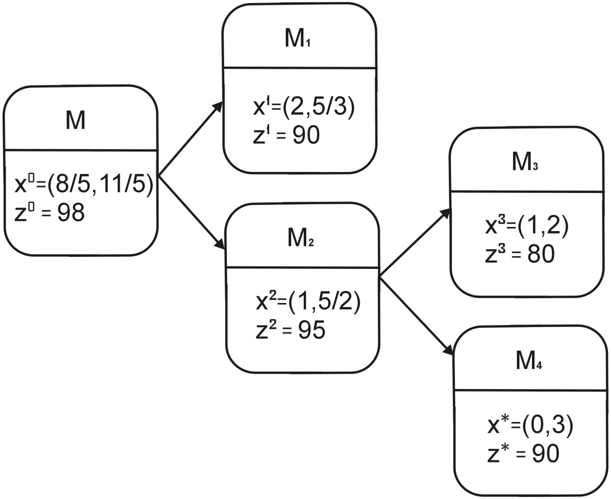
Lets demonstrate the B & B method for a textile company example.



Solutions x^3 and x^4 are both integers, their objective values are $z^3 = 80$ and $z^4 = 90$. The algorithm terminates, in no branch better value than 90 EUR (that is gained for $x^4 = [0,3]$) cannot be achieved.

Integer programming - method B & B

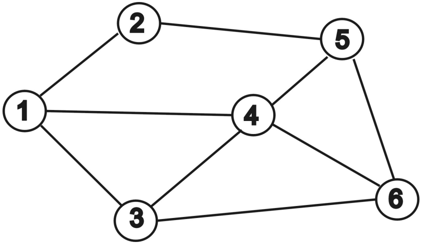
Lets demonstrate the B & B method for a textile company example.

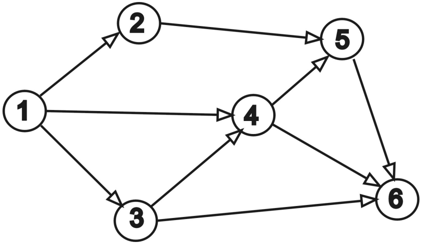


We can draw a scheme of the solution procedure.

Optimization on graphs

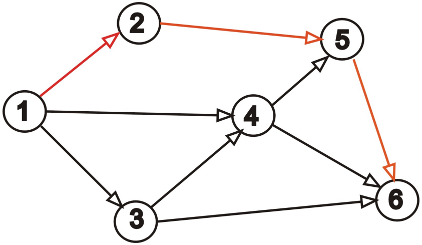
A lot of real-world systems can be represented by planar graphs (e.g. distribution network, project activities, etc.). A graph consists of the nodes, labelled by $u_1, \ldots, u_n$ and the edges, the edge connecting u_i and u_j is denoted by h_{ij}. We can represent a graph visually using points (circles) for the nodes and edges as lines connecting them. We distinguish between undirected edges allowing movement in both directions and directed edges, where the orientation is depicted by an arrow. If there are no directed edges in the graph, it is called undirected, else it is directed. Both above-mentioned types of graphs are shown in the following picture.





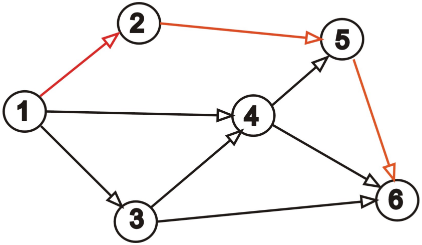
Optimization on graphs

The path from the node u_i to u_j is the sequence of successive edges, where the first one is starting in u_i and the last one is ending in u_j. If the orientation of the edges is respected, then the path is directed (else it is undirected). We can see a directed path from the node 1 to the node 6. There are only undirected paths going from 6 to 1.



Optimization on graphs

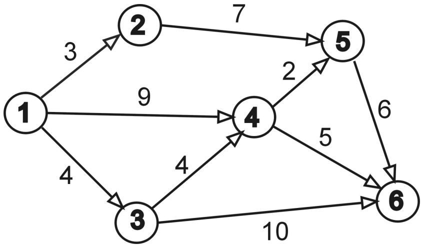
The path from the node u_i to u_j is the sequence of successive edges, where the first one is starting in u_i and the last one is ending in u_j. If the orientation of the edges is respected, then the path is directed (else it is undirected). We can see a directed path from the node 1 to the node 6. There are only undirected paths going from 6 to 1.



The path that has the same starting node as the end point $u_i = u_j$ is called a cycle, or in the case of an undirected graph, a circle. The graph in the picture contains circles, for example, $1 - 3 - 4 - 1$. If there is at least one undirected path between every two nodes, we call the graph connected. Every connected undirected graph that does not contain a circle is called the tree.

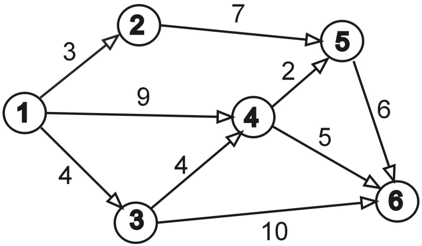
Optimization on graphs

Optimization on graphs involves weighted graphs. We assign edges weights according to their economic meaning (distances of distribution centres, costs on the transport of goods between them, etc.). A connected directed weighted graph with two special nodes (the source and the target) is called a network. If we assign weights to edges of the graph from our example, we get a network with the source in the node 1 and the target in the node 6.



Optimization on graphs

Optimization on graphs involves weighted graphs. We assign edges weights according to their economic meaning (distances of distribution centres, costs on the transport of goods between them, etc.). A connected directed weighted graph with two special nodes (the source and the target) is called a network. If we assign weights to edges of the graph from our example, we get a network with the source in the node 1 and the target in the node 6.



The path length is the sum of its edges weights. For example, the length of the directed path 1-2-5-6 is 3+7+6=16. Caution! A weighted graph is defined by the set of its nodes, edges and their weights, not by its graphical representation. Distances between the nodes do not need to correspond to weights of edges connecting them.

Optimization on graphs - problems

We can define many optimization problems on graphs:

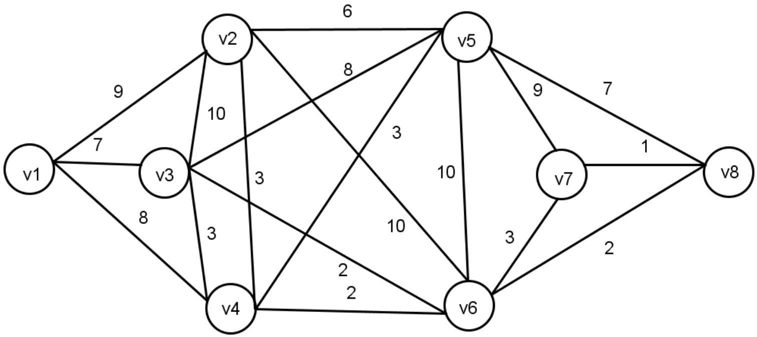
- A standard problem is represented by looking for the shortest path between two nodes. The problem is defined on directed as well as undirected graphs. There are many algorithms for solving this problem, one of the best known is the Dijkstra algorithm.
- Looking for a minimal spanning tree of the graph - the task is to choose such subset of edges, that each pair of nodes is connected and their total weight is minimal (so the spanning tree cannot contain cycles).
- Determining maximal flow in a network: If the weights of edges in a network represent capacities, we try to find a maximal number of units that can be transported from a source to a final node.
- Other problems, as colouring the graph, Chinese postman problem, travelling salesman problem, finding the median of a graph and the centre of a graph, etc.

Optimization on graphs - minimal spanning tree

For determining minimal spanning tree of the graph we can use a Kruskal algorithm: We form a subgraph by sequentially including edges with minimal weights until all the nodes are connected. We have to avoid forming a cycle: those edges that would complete the cycle are not included.
Example: Find a minimal spanning tree of a graph from T. Šubrt: Ekonomicko-matematické metody:

Optimization on graphs - minimal spanning tree

For determining minimal spanning tree of the graph we can use a Kruskal algorithm: We form a subgraph by sequentially including edges with minimal weights until all the nodes are connected. We have to avoid forming a cycle: those edges that would complete the cycle are not included.
Example: Find a minimal spanning tree of a graph from T. Šubrt: Ekonomicko-matematické metody:

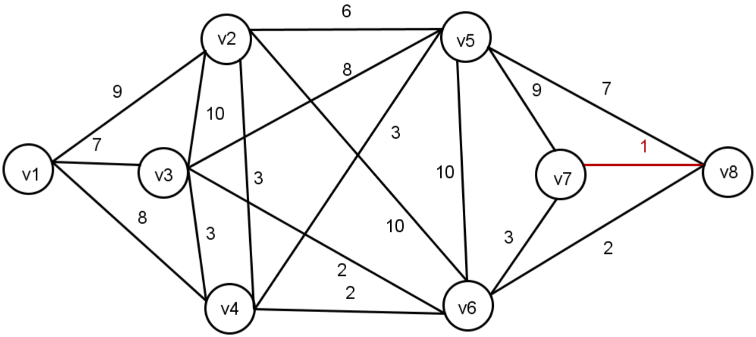


First we choose the edge of weight 1.

Optimization on graphs - minimal spanning tree

For determining minimal spanning tree of the graph we can use a Kruskal algorithm: We form a subgraph by sequentially including edges with minimal weights until all the nodes are connected. We have to avoid forming a cycle: those edges that would complete the cycle are not included.

Example: Find a minimal spanning tree of a graph from T. Šubrt: Ekonomicko-matematické metody:

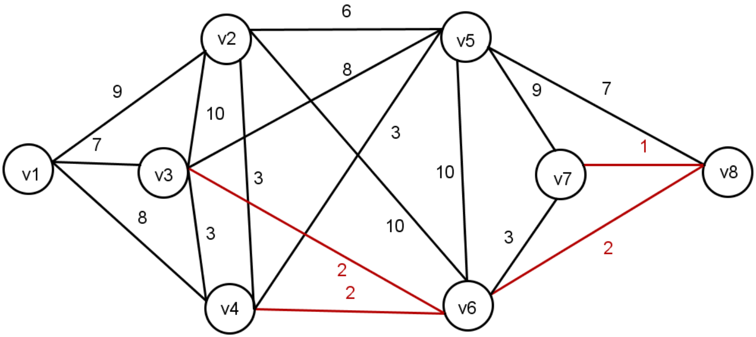


Then we add all edges with weight 2.

Optimization on graphs - minimal spanning tree

For determining minimal spanning tree of the graph we can use a Kruskal algorithm: We form a subgraph by sequentially including edges with minimal weights until all the nodes are connected. We have to avoid forming a cycle: those edges that would complete the cycle are not included.

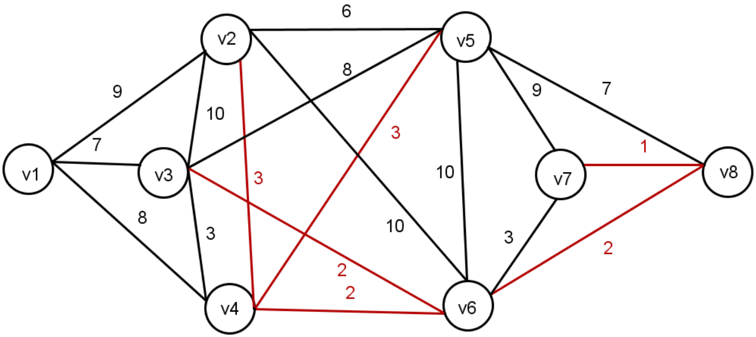
Example: Find a minimal spanning tree of a graph from T. Šubrt: Ekonomicko-matematické metody:



We add edges of weight 3 with the exception of v3-v4 which would close the circle.

Optimization on graphs - minimal spanning tree

For determining minimal spanning tree of the graph we can use a Kruskal algorithm: We form a subgraph by sequentially including edges with minimal weights until all the nodes are connected. We have to avoid forming a cycle: those edges that would complete the cycle are not included.
Example: Find a minimal spanning tree of a graph from T. Šubrt: Ekonomicko-matematické metody:

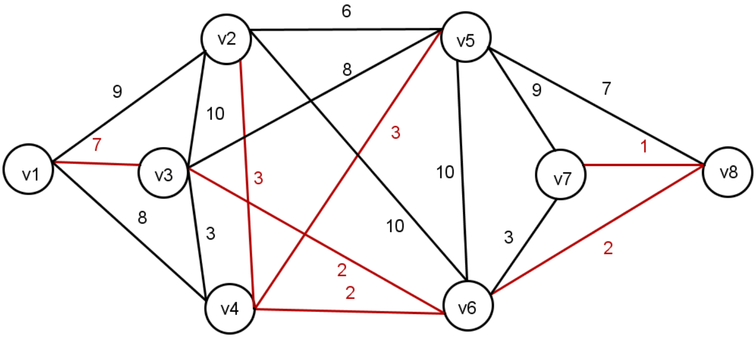


The edge weighted 6 cannot be included in the spanning tree, so we choose the edge weighted 7 instead.

Optimization on graphs - minimal spanning tree

For determining minimal spanning tree of the graph we can use a Kruskal algorithm: We form a subgraph by sequentially including edges with minimal weights until all the nodes are connected. We have to avoid forming a cycle: those edges that would complete the cycle are not included.

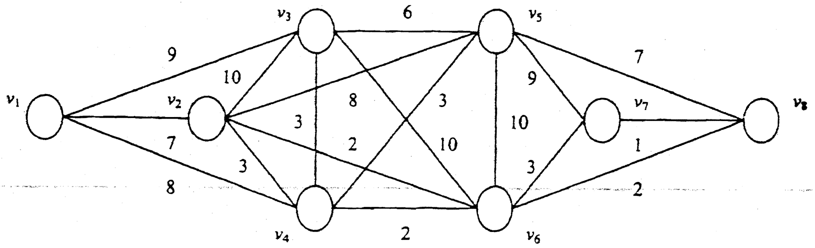
Example: Find a minimal spanning tree of a graph from T. Šubrt: Ekonomicko-matematické metody:



We have obtained a connected subgraph, so it is the sought minimal spanning tree (of total value 20).

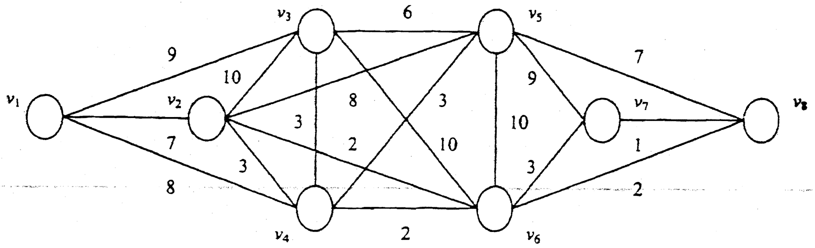
Optimization on graphs - the shortest path

For finding the shortest path between the v1 and other nodes we can use a Dijkstra algorithm: Let us show it on a graph from the previous example:



Optimization on graphs - the shortest path

For finding the shortest path between the v1 and other nodes we can use a Dijkstra algorithm: Let us show it on a graph from the previous example:



An algorithm divides nodes into two groups according to whether we already know their distance from v1 or not. We begin with the path of the length 0 from v1 to v1. In every step, we check all edges connecting a node from the „known distance“ group with the node from the other group and choose that one minimizing the sum of the distance and the weight of the edge; this minimal sum is the distance to a new node. This node now moves to the group with the known distance from v1. The procedure is repeated until all the distances are known

Optimization on graphs - the shortest path

We can follow the steps of the algorithm in a table of neighbours (the weights of edges are in brackets):

neighbours	*v1*	*v2*	*v3*	*v4*	*v5*	*v6*	*v7*	*v8*
	v2 (7)	v1 (7)	v1 (9)	v1 (8)	v2 (8)	v2 (2)	v5 (10)	v5 (7
	v3 (9)	v3 (10)	v2 (10)	v2 (3)	v3 (6)	v3 (10)	v6 (3)	v6 (2
	v4 (8)	v4 (3)	v4 (3)	v3 (3)	v4 (3)	v4 (2)	v8 (1)	v7 (1
		v5 (8)	v5 (6)	v5 (3)	v6 (10)	v5 (10)		
		v6 (2)	v6 (10)	v6 (2)	v7 (9)	v7 (3)		
					v8 (7)	v8 (2)		
dist. from v1	0							

In the first step we know only the distance to v1, so we go through its neighbours and choose the edge with the least weight, which is v1-v2 weighting 7. So the node v2 can be moved to the group with a known distance from v1.

Optimization on graphs - the shortest path

We can follow the steps of the algorithm in a table of neighbours (the weights of edges are in brackets):

neighbours	*v*1	*v*2	*v*3	*v*4	*v*5	*v*6	*v*7	*v*8
	~~v2 (7)~~	~~v1 (7)~~	~~v1 (9)~~	~~v1 (8)~~	~~v2 (8)~~	~~v2 (2)~~	v5 (10)	v5 (7
	v3 (9)	v3 (10)	~~v2 (10)~~	~~v2 (3)~~	v3 (6)	v3 (10)	v6 (3)	v6 (2
	v4 (8)	v4 (3)	v4 (3)	v3 (3)	v4 (3)	v4 (2)	v8 (1)	v7 (
		v5 (8)	v5 (6)	v5 (3)	v6 (10)	v5 (10)		
		v6 (2)	v6 (10)	v6 (2)	v7 (9)	v7 (3)		
					v8 (7)	v8 (2)		
dist. from v1	0	7						

In the second step we check the neighbours of v1 and also of v2, lengths of paths going through v2 are 7+10, 7+3, 7+8, 7+2, so the new node with a known distance from v1 is v4, because the length of v1-v4 is 8.

Optimization on graphs - the shortest path

We can follow the steps of the algorithm in a table of neighbours (the weights of edges are in brackets):

neighbours	*v1*	*v2*	*v3*	*v4*	*v5*	*v6*	*v7*	*v8*
	~~v2 (7)~~	~~v1 (7)~~	~~v1 (9)~~	~~v1 (8)~~	~~v2 (8)~~	~~v2 (2)~~	v5 (10)	v5 (7
	v3 (9)	v3 (10)	~~v2 (10)~~	~~v2 (3)~~	v3 (6)	v3 (10)	v6 (3)	v6 (2
	~~v4 (8)~~	~~v4 (3)~~	~~v4 (3)~~	v3 (3)	~~v4 (3)~~	~~v4 (2)~~	v8 (1)	v7 (
		v5 (8)	v5 (6)	v5 (3)	v6 (10)	v5 (10)		
		v6 (2)	v6 (10)	v6 (2)	v7 (9)	v7 (3)		
					v8 (7)	v8 (2)		
dist. from v1	0	7		8				

In the next step we check all neighbours of v1, v2 and v4 with unknown distance from v1. The shortest path through these nodes to their neighbours is v1-v3 with the length equal to 9. So this is the distance from v1 to v3.

Optimization on graphs - the shortest path

We can follow the steps of the algorithm in a table of neighbours (the weights of edges are in brackets):

neighbours	*v*1	*v*2	*v*3	*v*4	*v*5	*v*6	*v*7	*v*8
	~~v2 (7)~~	~~v1 (7)~~	~~v1 (9)~~	~~v1 (8)~~	~~v2 (8)~~	~~v2 (2)~~	v5 (10)	v5 (7
	~~v3 (9)~~	~~v3 (10)~~	~~v2 (10)~~	~~v2 (3)~~	~~v3 (6)~~	~~v3 (10)~~	v6 (3)	v6 (2
	~~v4 (8)~~	~~v4 (3)~~	~~v4 (3)~~	~~v3 (3)~~	~~v4 (3)~~	~~v4 (2)~~	v8 (1)	v7 (
		v5 (8)	v5 (6)	v5 (3)	v6 (10)	v5 (10)		
		v6 (2)	v6 (10)	v6 (2)	v7 (9)	v7 (3)		
					v8 (7)	v8 (2)		
dist. from v1	0	7	9	8				

The path to v6 through v2 is of the same length (9).

Optimization on graphs - the shortest path

We can follow the steps of the algorithm in a table of neighbours (the weights of edges are in brackets):

neighbours	*v*1	*v*2	*v*3	*v*4	*v*5	*v*6	*v*7	*v*8
	~~v2 (7)~~	~~v1 (7)~~	~~v1 (9)~~	~~v1 (8)~~	~~v2 (8)~~	~~v2 (2)~~	v5 (10)	v5 (7
	~~v3 (9)~~	~~v3 (10)~~	~~v2 (10)~~	~~v2 (3)~~	~~v3 (6)~~	~~v3 (10)~~	~~v6 (3)~~	~~v6 (2~~
	~~v4 (8)~~	~~v4 (3)~~	~~v4 (3)~~	~~v3 (3)~~	~~v4 (3)~~	~~v4 (2)~~	v8 (1)	v7 (1
		v5 (8)	v5 (6)	v5 (3)	~~v6 (10)~~	v5 (10)		
		~~v6 (2)~~	~~v6 (10)~~	~~v6 (2)~~	v7 (9)	v7 (3)		
					v8 (7)	v8 (2)		
dist. from v1	0	7	9	8		9		

We try to prolong certain path to a node with unknown distance from v1. The minimum length we can get by adding one edge is 11 for the path to v5 via v4.

Optimization on graphs - the shortest path

We can follow the steps of the algorithm in a table of neighbours (the weights of edges are in brackets):

neighbours	*v*1	*v*2	*v*3	*v*4	*v*5	*v*6	*v*7	*v*8
	~~v2 (7)~~	~~v1 (7)~~	~~v1 (9)~~	~~v1 (8)~~	~~v2 (8)~~	~~v2 (2)~~	~~v5 (10)~~	~~v5 (~~
	~~v3 (9)~~	~~v3 (10)~~	~~v2 (10)~~	~~v2 (3)~~	~~v3 (6)~~	~~v3 (10)~~	~~v6 (3)~~	~~v6 (~~
	~~v4 (8)~~	~~v4 (3)~~	~~v4 (3)~~	~~v3 (3)~~	~~v4 (3)~~	~~v4 (2)~~	v8 (1)	v7 (
		~~v5 (8)~~	~~v5 (6)~~	~~v5 (3)~~	~~v6 (10)~~	~~v5 (10)~~		
		~~v6 (2)~~	~~v6 (10)~~	~~v6 (2)~~	v7 (9)	v7 (3)		
					v8 (7)	v8 (2)		
dist. from v1	0	7	9	8	11	9		

The path of the same length 11 can be achieved by connecting v8 through v6.

Optimization on graphs - the shortest path

We can follow the steps of the algorithm in a table of neighbours (the weights of edges are in brackets):

neighbours	$v1$	$v2$	$v3$	$v4$	$v5$	$v6$	$v7$	$v8$
	~~v2 (7)~~	~~v1 (7)~~	~~v1 (9)~~	~~v1 (8)~~	~~v2 (8)~~	~~v2 (2)~~	~~v5 (10)~~	~~v5 (7~~
	~~v3 (9)~~	~~v3 (10)~~	~~v2 (10)~~	~~v2 (3)~~	~~v3 (6)~~	~~v3 (10)~~	~~v6 (3)~~	~~v6 (2~~
	~~v4 (8)~~	~~v4 (3)~~	~~v4 (3)~~	~~v3 (3)~~	~~v4 (3)~~	~~v4 (2)~~	~~v8 (1)~~	v7 (1
		~~v5 (8)~~	~~v5 (6)~~	~~v5 (3)~~	~~v6 (10)~~	~~v5 (10)~~		
		~~v6 (2)~~	~~v6 (10)~~	~~v6 (2)~~	v7 (9)	v7 (3)		
					~~v8 (7)~~	~~v8 (2)~~		
dist. from v1	0	7	9	8	11	9		11

Finally, we attach the node v7; the shortest way how to do it is via v6. The total distance is 12.

Optimization on graphs - the median of a graph

The median of a graph is the node minimizing the sum of distances from other nodes. We use a graph from the example on TSP, but now we want choose the city, which is the most suitable location for a central depot.

	Kralupy	Veltrusy	Slaný	Velvary	Zlonice	Vraný	Bříza	SUM
Kralupy	0	4	16	8	18	25	17	88
Veltrusy	4	0	20	12	22	28	13	99
Slaný	16	20	0	12	7	14	17	86
Velvary	8	12	12	0	10	17	10	69
Zlonice	18	22	7	10	0	7	10	74
Vraný	25	28	14	17	7	0	15	106
Bříza	17	13	17	10	10	15	0	82

Optimization on graphs - the median of a graph

The median of a graph is the node minimizing the sum of distances from other nodes. We use a graph from the example on TSP, but now we want choose the city, which is the most suitable location for a central depot.

	Kralupy	Veltrusy	Slaný	Velvary	Zlonice	Vraný	Bříza	SUM
Kralupy	0	4	16	8	18	25	17	88
Veltrusy	4	0	20	12	22	28	13	99
Slaný	16	20	0	12	7	14	17	86
Velvary	8	12	12	0	10	17	10	69
Zlonice	18	22	7	10	0	7	10	74
Vraný	25	28	14	17	7	0	15	106
Bříza	17	13	17	10	10	15	0	82

In order to minimize the total distance to the depot, we should locate it in Velvary.

Comment: When locating two depots we speak about the double median, etc.

Optimization on graphs - the centre of a graph

The centre of a graph is the node minimizing maximal distance from other nodes. Let us show it on the same example as before, but now we want to choose location for a firemen station.

	Kralupy	Veltrusy	Slaný	Velvary	Zlonice	Vraný	Bříza	MAX
Kralupy	0	4	16	8	18	25	17	25
Veltrusy	4	0	20	12	22	28	13	28
Slaný	16	20	0	12	7	14	17	20
Velvary	8	12	12	0	10	17	10	17
Zlonice	18	22	7	10	0	7	10	22
Vraný	25	28	14	17	7	0	15	28
Bříza	17	13	17	10	10	15	0	17

Optimization on graphs - the centre of a graph

The centre of a graph is the node minimizing maximal distance from other nodes. Let us show it on the same example as before, but now we want to choose location for a firemen station.

	Kralupy	Veltrusy	Slaný	Velvary	Zlonice	Vraný	Bříza	MAX
Kralupy	0	4	16	8	18	25	17	25
Veltrusy	4	0	20	12	22	28	13	28
Slaný	16	20	0	12	7	14	17	20
Velvary	8	12	12	0	10	17	10	17
Zlonice	18	22	7	10	0	7	10	22
Vraný	25	28	14	17	7	0	15	28
Bříza	17	13	17	10	10	15	0	17

The best location for a firemen station is Velvary or Bříza, because every city is at most 17 km far from them.

Remark: When locating two stations, we speak about a double centre, etc.

Optimization on networks

A common problem solved on networks is the maximal flow problem (on waterworks, conduction, road or data network,etc.). The problem must be stated in terms of:

- the network description (a graph, usually directed)
- where does the flow start (a special node called the source)
- which is the final node (a target)
- How much of the medium can go through the edges (capacities = non-negative weights of edges)

A graph described above is called the network.

Optimization on networks

A common problem solved on networks is the maximal flow problem (on waterworks, conduction, road or data network,etc.). The problem must be stated in terms of:

- the network description (a graph, usually directed)
- where does the flow start (a special node called the source)
- which is the final node (a target)
- How much of the medium can go through the edges (capacities = non-negative weights of edges)

A graph described above is called the network.
A flow on a network is a function assigning every edge a non-negative number less or equal to its capacity. This function has to fulfil the so-called Kirchhoff's law: The sum of the flows on entering edges is equal to the sum of the flows on leaving edges for every node except the source and the target.
The flow size is the sum of flows entering the target. Zero flow means that nothing goes through the network (every edge is assigned a zero value).

Optimization on networks

The problem of maximal flow can be solved by an improving algorithm: we start with zero flow and look for an improving path (every edge of this path has nonzero reserve - that means the difference between the capacity and the actual flow on the edge). The flow on the path can be increased by the minimal reserve of its edges. We follow by looking for another improving path. However, this approach need not lead to an optimal solution, sometimes we have to decrease flow on some edges in order to obtain higher total flow.

Optimization on networks

The problem of maximal flow can be solved by an improving algorithm: we start with zero flow and look for an improving path (every edge of this path has nonzero reserve - that means the difference between the capacity and the actual flow on the edge). The flow on the path can be increased by the minimal reserve of its edges. We follow by looking for another improving path. However, this approach need not lead to an optimal solution, sometimes we have to decrease flow on some edges in order to obtain higher total flow. Better results are gained by applying following Ford-Fulkerson algorithm:

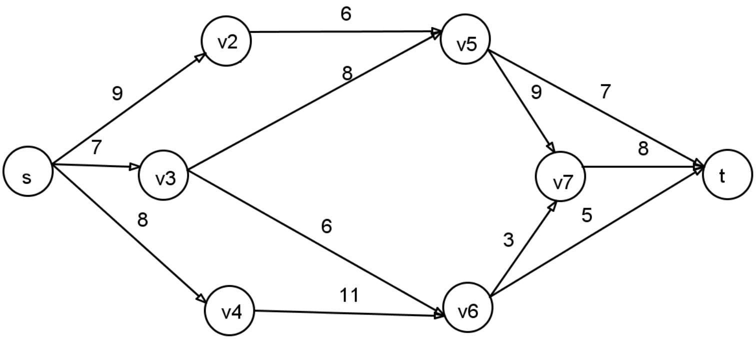
- For each edge, we monitor its reserve as well as its reverse reserve (how much the flow can be decreased). In the beginning, reserves on all edges are equal to their capacities (and reverse reserves of directed edges are zero).
- We find an improving path and increase its flow by its reserve. At the same time, we decrease reserves of all edges on the path and increase their reverse reserves.
- We proceed until there is no path with nonzero reserve.

Maximal flow in network: Ford-Fulkerson algorithm

Let’s apply Ford-Fulkerson algorithm for determining maximal flow in a given network:

Maximal flow in network: Ford-Fulkerson algorithm

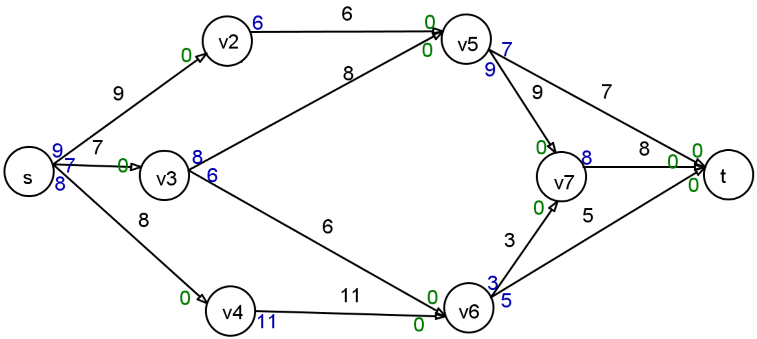
Let's apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



We start with zero flow, we denote reserves by blue numbers on edges and reverse reserves by green numbers.

Maximal flow in network: Ford-Fulkerson algorithm

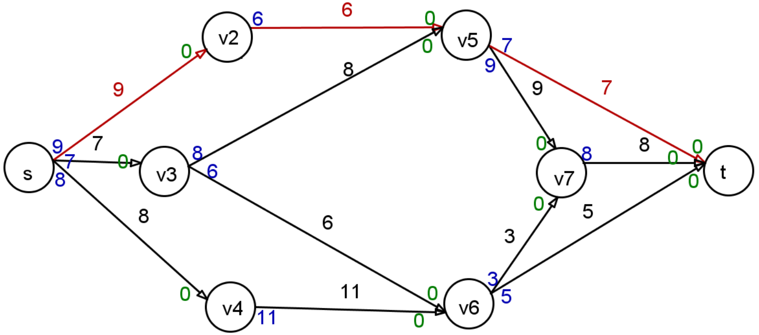
Let’s apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



We find an improving path.

Maximal flow in network: Ford-Fulkerson algorithm

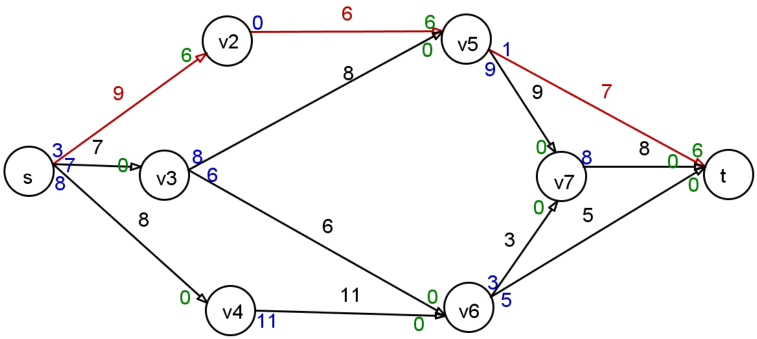
Let's apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



Minimal reserve on this path is 6, so we add this number to the current flow on the path. Reserves of all involved edges drop by 6, simultaneously all their reverse reserves increase by the same number.

Maximal flow in network: Ford-Fulkerson algorithm

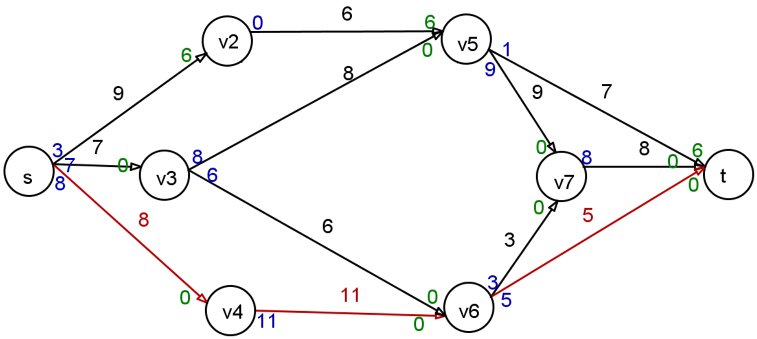
Let’s apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



We find another improving path.

Maximal flow in network: Ford-Fulkerson algorithm

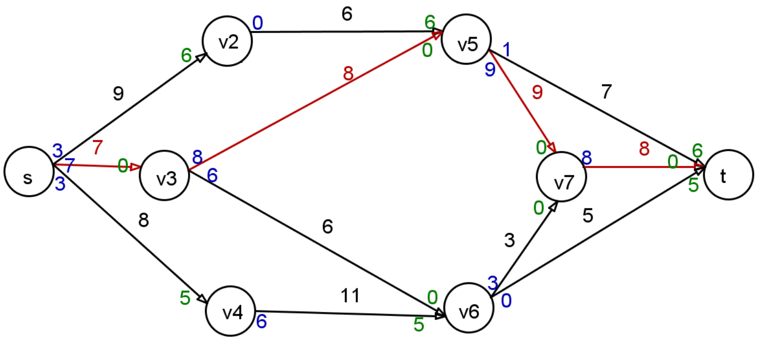
Let’s apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



There is an edge with a reserve equal to 5 on this path, so we cannot improve the flow more than that.

Maximal flow in network: Ford-Fulkerson algorithm

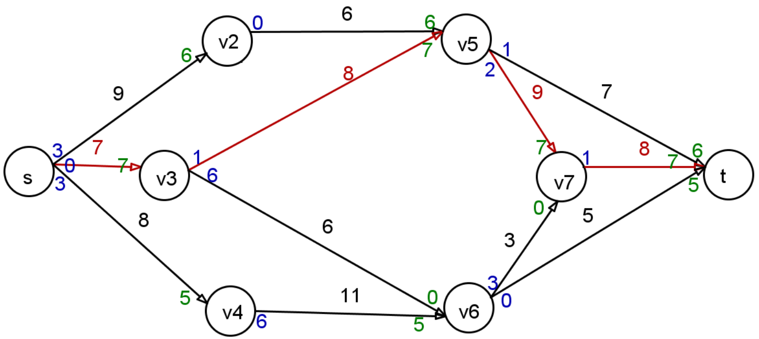
Let's apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



For increasing total flow we find another path with nonzero reserve.

Maximal flow in network: Ford-Fulkerson algorithm

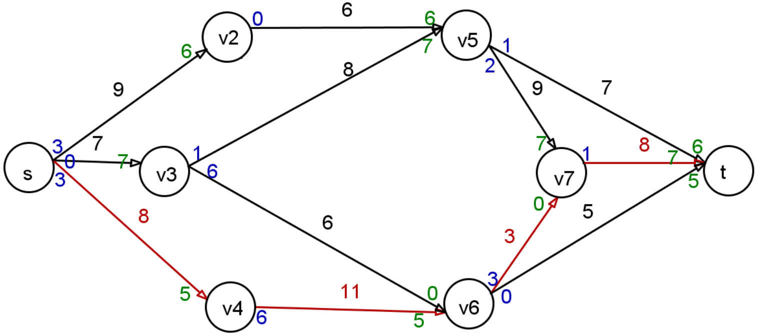
Let's apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



We increased flow on the path by 7.

Maximal flow in network: Ford-Fulkerson algorithm

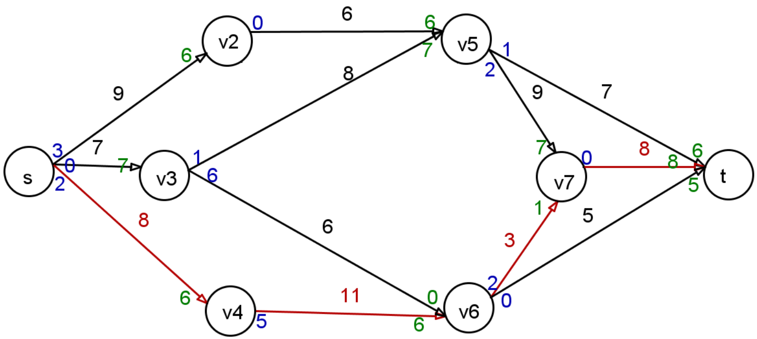
Let’s apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



There is also an improving path with reserve 1.

Maximal flow in network: Ford-Fulkerson algorithm

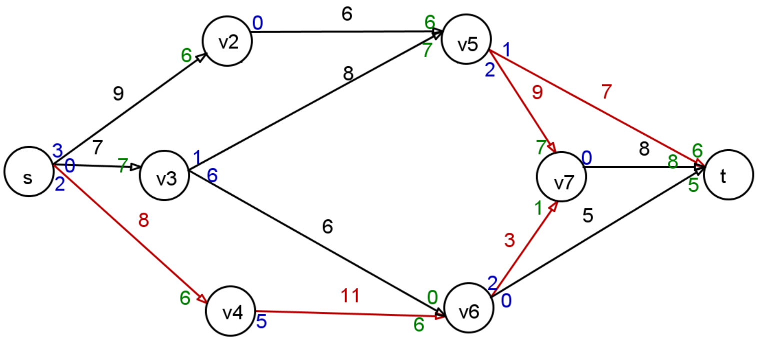
Let’s apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



We increased the flow.

Maximal flow in network: Ford-Fulkerson algorithm

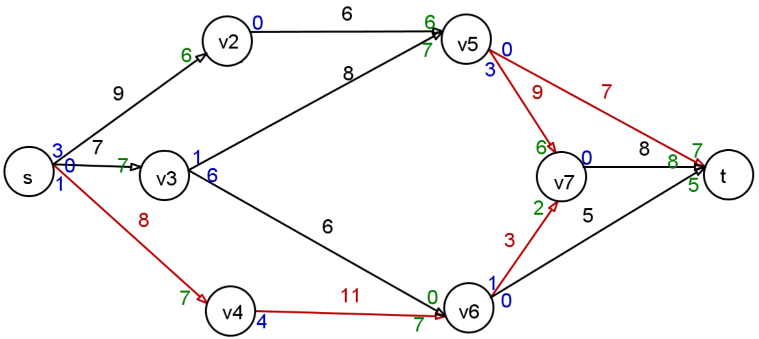
Let’s apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



We can improve the flow on the path that goes between v5 and v7 in reverse direction.

Maximal flow in network: Ford-Fulkerson algorithm

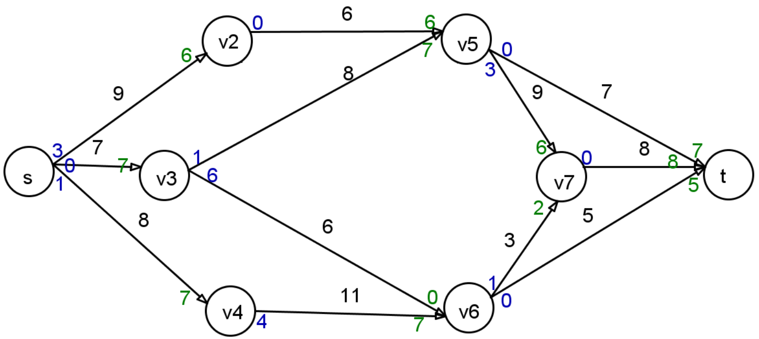
Let's apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



We increase the flow by a minimal reserve of this path (1, on the last edge). The size of the flow on the edge connecting v5 and v7 is decreased by 1.

Maximal flow in network: Ford-Fulkerson algorithm

Let’s apply Ford-Fulkerson algorithm for determining maximal flow in a given network:



Total size of the resulting flow is 7+8 +5=20 (see green numbers on edges entering the target). This must be the optimal solution, there is no free capacity left.

Project management

A typical application area for application of graph algorithms is project management. Generally, a project can be understood as a set of activities and connections between them. The activities are characterized by their expected duration, costs, requirements on conditions of their realization, their predecessors, etc. Usually, we try to find answers to these questions:

- What is the shortest possible time for a realization of the project?
- Which are the key activities for finishing the project on time, i.e. critical activities?
- What time reserves do we have for the other activities?
- How to prepare the time schedule for the project execution?

Besides the time analysis of a project, we are interested in the cost analysis (costs differ according to project duration). We can also explore time distribution of the resources needed for the individual activities.

Project management - construction of a project network

The project can be visualized by a network graph, where the edges represent the activities, their weights are equal to durations of the activities and nodes correspond to the moments of starting or finishing the activities. First, we have to define the activities, estimate their durations and specify the order of activities by a list of activity predecessors.

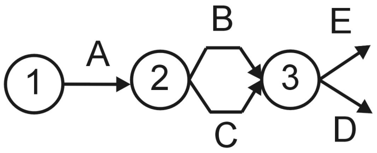
Project management - construction of a project network

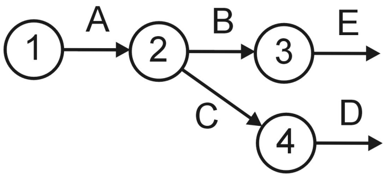
The project can be visualized by a network graph, where the edges represent the activities, their weights are equal to durations of the activities and nodes correspond to the moments of starting or finishing the activities. First, we have to define the activities, estimate their durations and specify the order of activities by a list of activity predecessors. Let us show the network of an example from J.Jablonský: Operační výzkum: Construct the graph of a project of launching a new business centre for Q-mark company which is defined by elementary activities and their characteristics, see the table:

activity	activity description	duration [weeks]	predecessors
A	project kick off	6	-
B	processing the project	4	A
C	hiring management	3	A
D	hiring staff	3	B,C
E	reconstruction and the equipment	8	B
F	staff training	2	D
G	selection of goods	2	B,C
H	contracts with suppliers	5	G
I	ordering goods	3	E,F,H
J	marketing campaign	2	H

Project management - construction of a project network

When constructing a graph we may face difficulties with the definition of nodes. For example, we have to finish B and C before D, but at the same time, B is the predecessor of E, while C is not. How to capture the right order of activities? Examples of an incorrect representation are in the following figure:





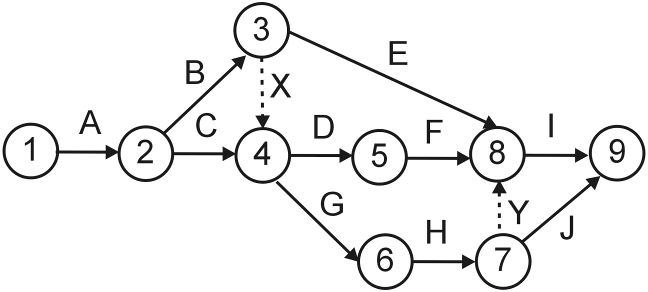
Similar situation occurs for activities I,J that have a common predecessor H, but there are two more predecessors E and F for I.

Project management - construction of a project network

The above-mentioned issues can be solved by introducing dummy activities with corresponding dummy edges that supplement missing connections. We denote them by dashed lines. The duration of a dummy task is always zero. In our example we need two dummy activities: X representing the relation between B and D and a dummy activity Y for H and I.

Project management - construction of a project network

The above-mentioned issues can be solved by introducing dummy activities with corresponding dummy edges that supplement missing connections. We denote them by dashed lines. The duration of a dummy task is always zero. In our example we need two dummy activities: X representing the relation between B and D and a dummy activity Y for H and I. One of the possible ways of constructing the graph is in the figure:



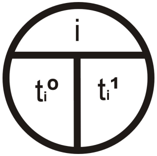
There is a golden rule for constructing a project network: always try to number the nodes in such a way that every edge goes from the node with a lesser index to the node with a higher index. Then the graph doesn't contain any undesirable cycle.

Critical Path Method (CPM)

The method CPM was developed in the fifties of the last century. It is based on computing four characteristics for every activity:

- The earliest start of the activities beginning in the node u_i; we denote it by t_i^0
- The earliest finish of activity represented by the edge h_{ij}; we compute it by adding the duration of the task: $t_i^0 + y_{ij}$
- The latest finish of activities on edges going to u_j; we label it by t_j^1
- The latest start of activity represented by the edge h_{ij} is computed by subtracting its duration: $t_j^1 - y_{ij}$

Let's mark all the nodes in a project network by the earliest start and latest finish of activities going from/to them:



Critical Path Method (CPM)

The algorithm comprises of four phases:

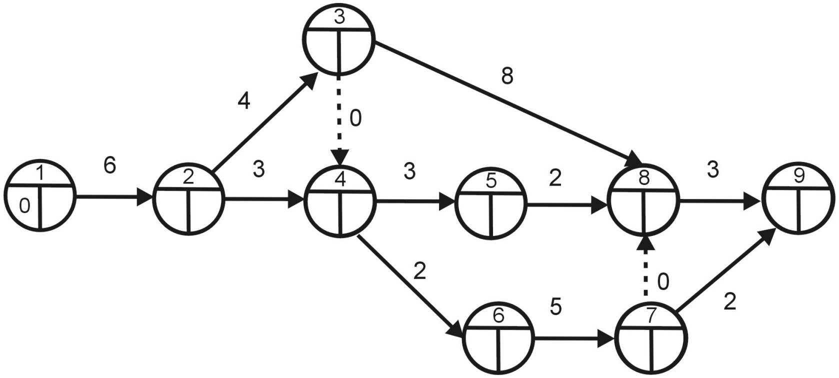
1. a computation of earliest starts: For activities beginning at u_j we find maxima of earliest finishes of entering activities: $t_j^0 = max_i(t_i^0 + y_{ij})$ For the last node we obtain the earliest finish of the whole project.
2. a computation of latest finishes: For activities ending at u_i we find minima of latest starts of following activities. $t_i^1 = min_j(t_j^1 - y_{ij})$
3. a computation of total time reserves: Activity represented by the edge h_{ij} cannot start earlier than t_i^0 and shall not finish later than t_j^1. The time window for its realization is $t_j^1 - t_i^0$ and because its duration is y_{ij}, we get formula for its time reserve: $R_{ij} = t_j^1 - t_i^0 - y_{ij}$
4. a construction of the time schedule

Critical Path Method (CPM) - 1st phase

In the first phase of CPM we go through the graph from the left to the right and compute earliest starts of all activities.

Critical Path Method (CPM) - 1st phase

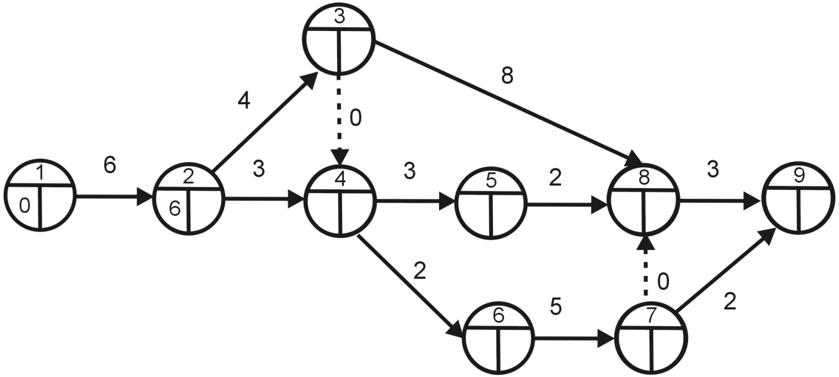
In the first phase of CPM we go through the graph from the left to the right and compute earliest starts of all activities.



First lets have a look on the graph with the durations of activities.

Critical Path Method (CPM) - 1st phase

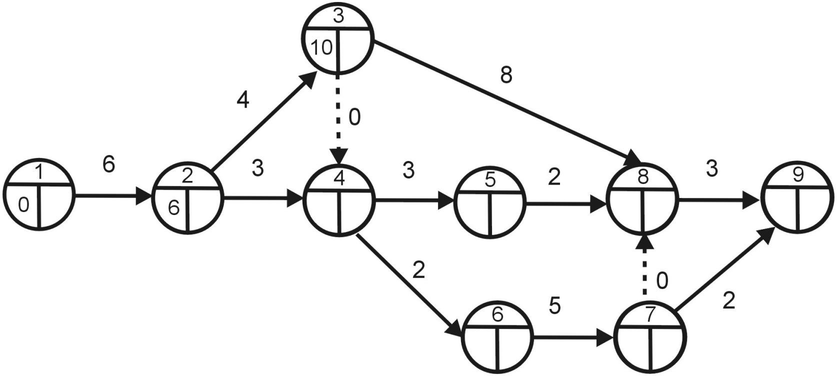
In the first phase of CPM we go through the graph from the left to the right and compute earliest starts of all activities.



The earliest start of tasks B and C is $t_2^0 = 0 + 6 = 6$.

Critical Path Method (CPM) - 1st phase

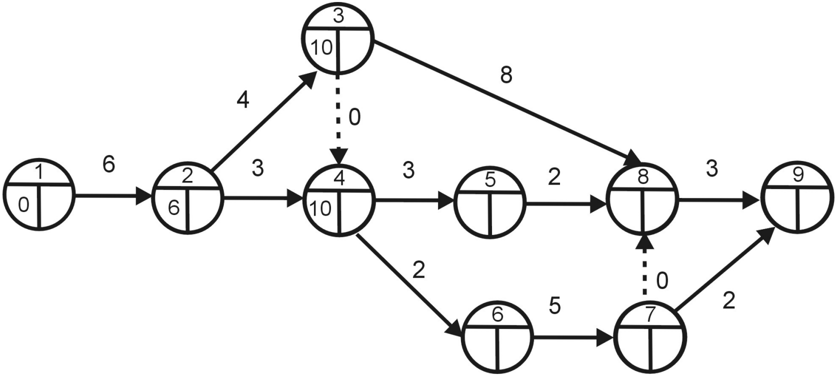
In the first phase of CPM we go through the graph from the left to the right and compute earliest starts of all activities.



The earliest start of E is $t_3^0 = 6 + 4 = 10$.

Critical Path Method (CPM) - 1st phase

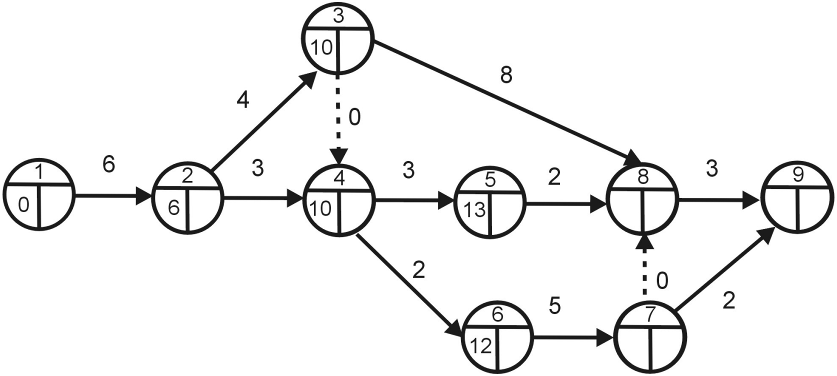
In the first phase of CPM we go through the graph from the left to the right and compute earliest starts of all activities.



The earliest start of activities D and G is $t_4^0 = max(10 + 0, 6 + 3) = 10$.

Critical Path Method (CPM) - 1st phase

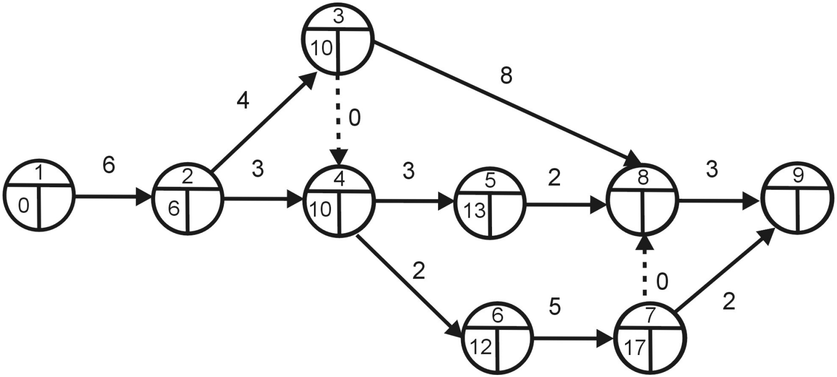
In the first phase of CPM we go through the graph from the left to the right and compute earliest starts of all activities.



The earliest start of F is $t_5^0 = 10 + 3 = 13$ and for H it is $t_6^0 = 10 + 2 = 12$.

Critical Path Method (CPM) - 1st phase

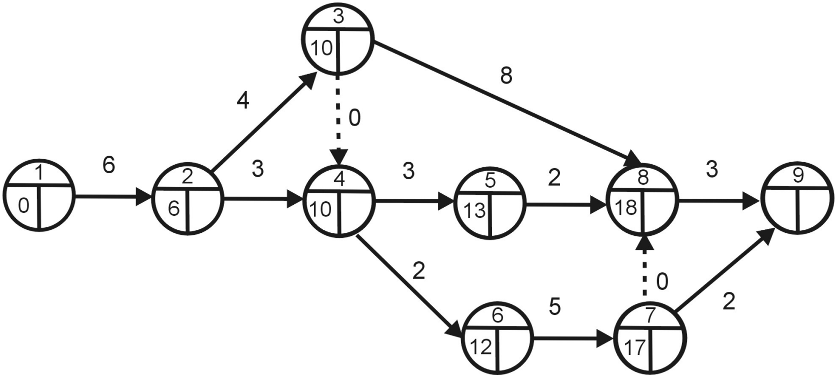
In the first phase of CPM we go through the graph from the left to the right and compute earliest starts of all activities.



The earliest start of task J is $t_7^0 = 12 + 5 = 17$.

Critical Path Method (CPM) - 1st phase

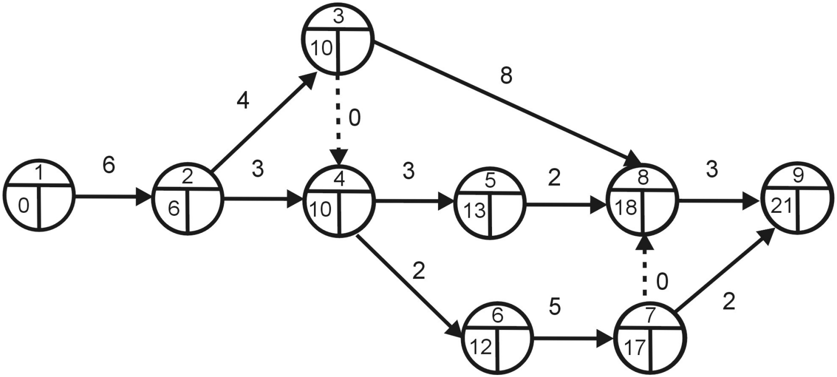
In the first phase of CPM we go through the graph from the left to the right and compute earliest starts of all activities.



The earliest start of task I is $t_8^0 = max(10 + 8, 13 + 2, 17 + 2) = 18$.

Critical Path Method (CPM) - 1st phase

In the first phase of CPM we go through the graph from the left to the right and compute earliest starts of all activities.



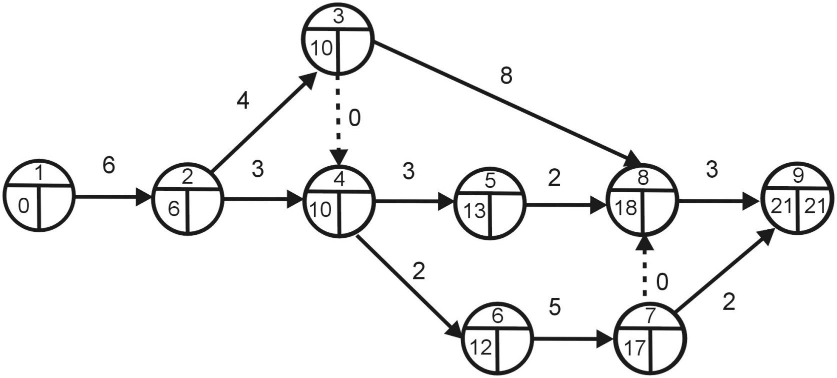
We have gone through the whole network. The project cannot finish earlier than in $t_9^0 = max(18 + 3, 17 + 2) = 21$ weeks.

Critical Path Method (CPM) - 2nd phase

In the second phase we proceed from the right to the left and compute the latest finish for every node. We assume that the project should be completed in the shortest time, that is in 21 weeks.

Critical Path Method (CPM) - 2nd phase

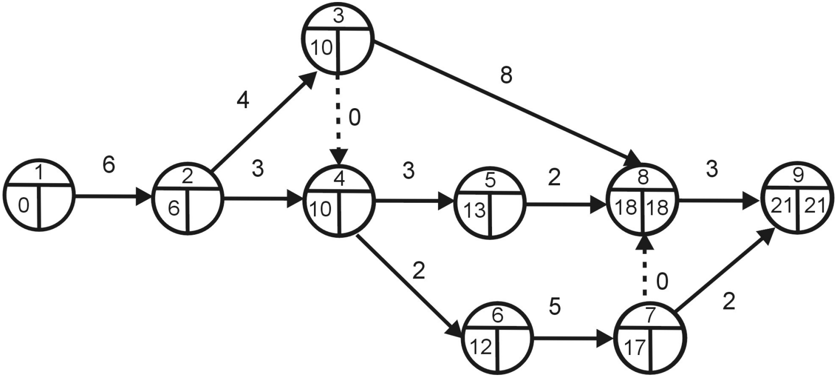
In the second phase we proceed from the right to the left and compute the latest finish for every node. We assume that the project should be completed in the shortest time, that is in 21 weeks.



If we insist on finishing the whole project in 21 weeks, the activities I andJ shouldn't finish later than $t_9^1 = 21$.

Critical Path Method (CPM) - 2nd phase

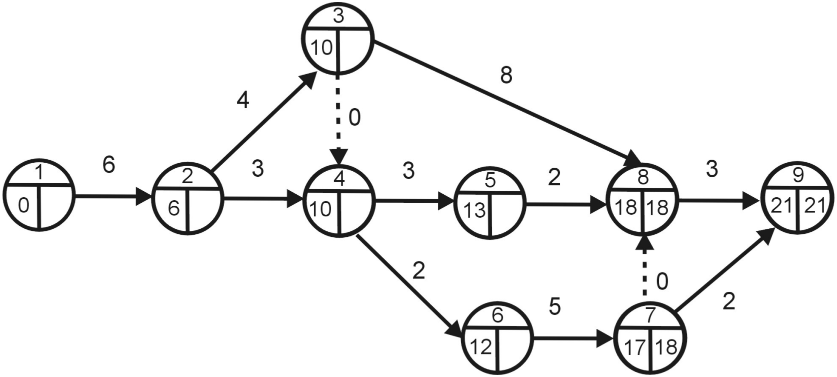
In the second phase we proceed from the right to the left and compute the latest finish for every node. We assume that the project should be completed in the shortest time, that is in 21 weeks.



The latest finish of activities F and E is $t_8^1 = 21 - 3 = 18$.

Critical Path Method (CPM) - 2nd phase

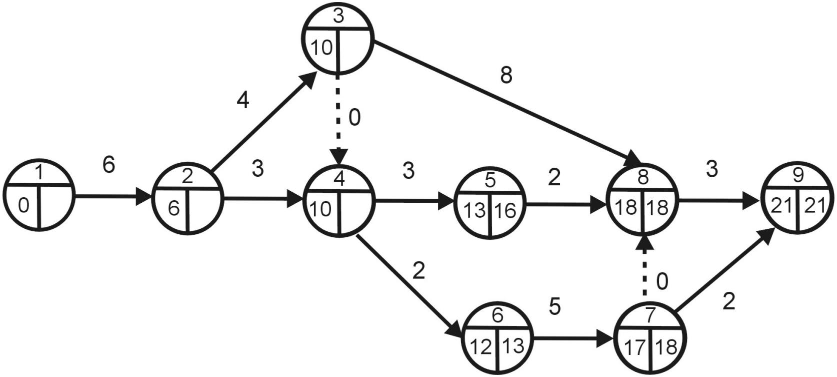
In the second phase we proceed from the right to the left and compute the latest finish for every node. We assume that the project should be completed in the shortest time, that is in 21 weeks.



The latest finish of the task H is $t_7^1 = min(21 - 2, 18 - 0) = 18$.

Critical Path Method (CPM) - 2nd phase

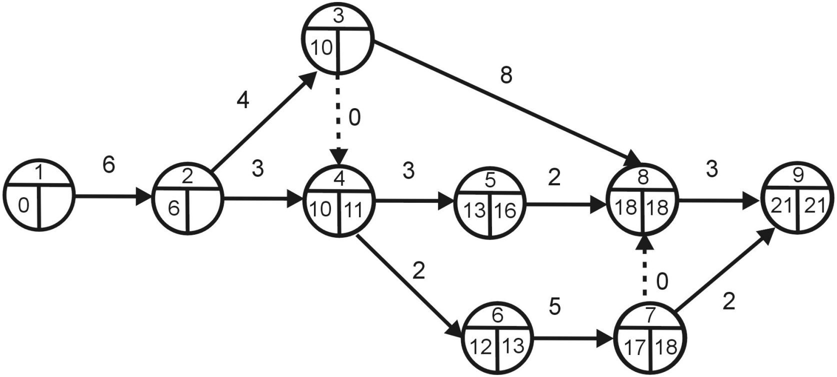
In the second phase we proceed from the right to the left and compute the latest finish for every node. We assume that the project should be completed in the shortest time, that is in 21 weeks.



The latest finish of the task D is $t_5^1 = 18 - 2 = 16$ and for G it is $t_6^1 = 18 - 5 = 13$.

Critical Path Method (CPM) - 2nd phase

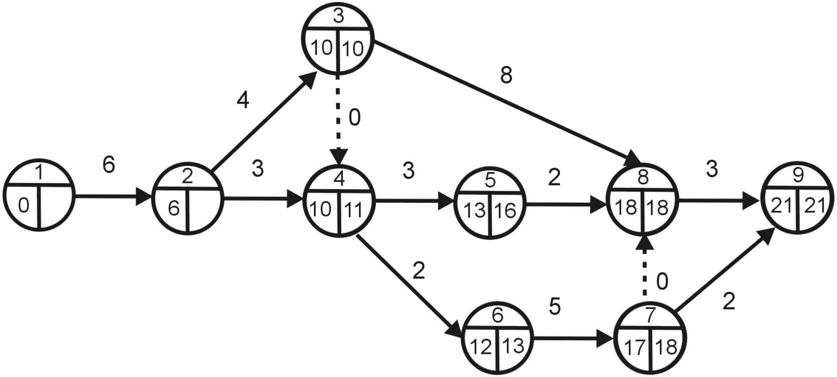
In the second phase we proceed from the right to the left and compute the latest finish for every node. We assume that the project should be completed in the shortest time, that is in 21 weeks.



The latest finish of the activity C is $t_4^1 = min(13 - 2, 16 - 3) = 11$.

Critical Path Method (CPM) - 2nd phase

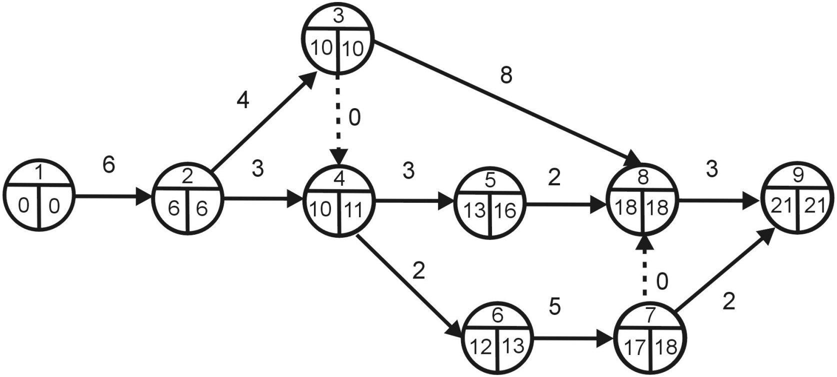
In the second phase we proceed from the right to the left and compute the latest finish for every node. We assume that the project should be completed in the shortest time, that is in 21 weeks.



The latest finish of task B is $t_3^1 = min(18-8, 11-0) = 10$.

Critical Path Method (CPM) - 2nd phase

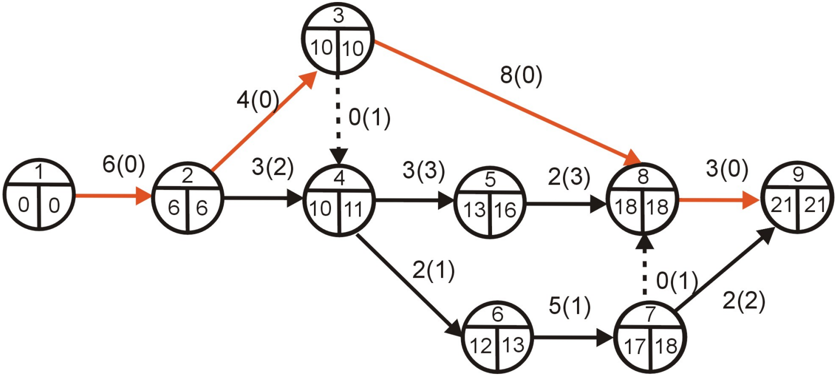
In the second phase we proceed from the right to the left and compute the latest finish for every node. We assume that the project should be completed in the shortest time, that is in 21 weeks.



The latest finish of the first activity A is $t_2^1 = min(10 - 4, 11 - 3) = 6$. It is not surprising that $t_1^1 = 6 - 6 = 0$, as we insisted on the shortest possible realization time.

Critical Path Method (CPM) - 3rd phase

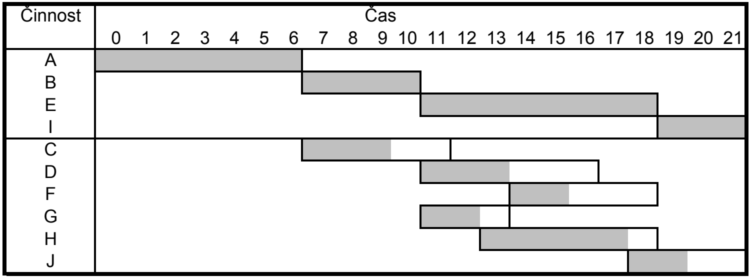
We compute time reserves of the activities according to the formula $R_{ij} = t_j^1 - t_i^0 - y_{ij}$. For example, the time window for a task J is between 17th and 21th week and its realization lasts two weeks, so its reserve is $R_{79} = 21 - 17 - 2 = 2$ weeks, etc.



We supplied the edges with time reserves (the numbers in brackets). The critical path consists of activities with zero reserve, A,B,E,I; they are highlighted by a red colour.

Critical Path Method (CPM) - 4th phase

The last but not least phase is creating the project schedule. It is necessary to decide which activities can be realized simultaneously with regard to the possibility of sharing resources needed for individual tasks. In the figure we can see the timing diagram (it is called a Gantt chart) for our example, earliest starts and latest finishes of the tasks are represented by the frames and the duration is highlighted by shading. Critical activities with zero time reserves are placed in the upper part of the table. Those activities have to be realized immediately one after another unless the project is delayed.



Method PERT

The method PERT (Program Evaluation and Review Technique) is a probabilistic extension of the CPM method. In reality, it is often impossible to specify exact duration of activities, so in the PERT the values y_{ij} are replaced by random variables with supports $\langle a_{ij}, b_{ij} \rangle$. The bounds represent optimistic and pessimistic estimates of the duration. By m_{ij} we denote the most likely, the modal estimate.

Method PERT

The method PERT (Program Evaluation and Review Technique) is a probabilistic extension of the CPM method. In reality, it is often impossible to specify exact duration of activities, so in the PERT the values y_{ij} are replaced by random variables with supports $\langle a_{ij}, b_{ij} \rangle$. The bounds represent optimistic and pessimistic estimates of the duration. By m_{ij} we denote the most likely, the modal estimate. As we generally don't know the probability distribution of the random variables, we approximate it usually by a β-distribution. There are following formulas for its mean and variance:

$$\mu_{ij} = \frac{a_{ij} + 4m_{ij} + b_{ij}}{6}$$

$$\sigma_{ij} = \frac{b_{ij} - a_{ij}}{6}$$

Method PERT

The method PERT (Program Evaluation and Review Technique) is a probabilistic extension of the CPM method. In reality, it is often impossible to specify exact duration of activities, so in the PERT the values y_{ij} are replaced by random variables with supports $\langle a_{ij}, b_{ij} \rangle$. The bounds represent optimistic and pessimistic estimates of the duration. By m_{ij} we denote the most likely, the modal estimate. As we generally don't know the probability distribution of the random variables, we approximate it usually by a β-distribution. There are following formulas for its mean and variance:

$$\mu_{ij} = \frac{a_{ij} + 4m_{ij} + b_{ij}}{6}$$

$$\sigma_{ij} = \frac{b_{ij} - a_{ij}}{6}$$

The algorithm of the PERT method doesn't differ from the CPM, with the exception that durations y_{ij} are substituted by their means μ_{ij}. If certain conditions are met, we can use the central limit theorem and estimate the total time of project realization by a Gaussian random variable. This allows us to answer questions such as: What is the probability of finishing project in a certain time, what total time of finishing corresponds to a probability p, etc.

Method PERT - example

Example : A company plans to reconstruct its manufacturing plant including replacement of the production equipment, construction work, overhaul of the wiring, etc. Individual activities of the project together with their expected durations in weeks (optimistic/modal/pessimistic) are in the table. Use method PERT for determining the critical path.

Activity description	predecessors	a_{ij}	m_{ij}	b_{ij}
A Dismantling old equipment	-	5	8	10
B Repairing the roof of the production hall	-	4	6	7
C Floor Repair	A	1	2	5
D Interior design	B,C	2	4	6
E Overhaul of wiring	D	7	10	14
F Installation New Production Equipment	E	10	12	13
G Installation of air conditioning	E	4	5	8
H Testing	F	3	4	6
I Completing	G	1	3	5

Method PERT - example

Solution: We compute means and variances:

(i,j)	a_{ij}	m_{ij}	b_{ij}	μ_{ij}	σ_{ij}^2
A(1,2)	5	8	10	7,83	0,69
B(1,3)	4	6	7	5,83	0,25
C(2,3)	1	2	5	2,33	0,44
D(3,4)	2	4	6	4,00	0,44
E(4,5)	7	10	14	10,17	1,36
F(5,6)	10	12	13	11,83	0,25
G(5,7)	4	5	8	5,33	0,44
H(6,8)	3	4	6	4,17	0,25
I(7,8)	1	3	5	3,00	0,44

Method PERT - example

Solution: We compute means and variances:

(i,j)	a_{ij}	m_{ij}	b_{ij}	μ_{ij}	σ^2_{ij}
A(1,2)	5	8	10	7,83	0,69
B(1,3)	4	6	7	5,83	0,25
C(2,3)	1	2	5	2,33	0,44
D(3,4)	2	4	6	4,00	0,44
E(4,5)	7	10	14	10,17	1,36
F(5,6)	10	12	13	11,83	0,25
G(5,7)	4	5	8	5,33	0,44
H(6,8)	3	4	6	4,17	0,25
I(7,8)	1	3	5	3,00	0,44

The critical path of simple projects can be determined by checking all the paths connecting the source and target node in the project network (there are four of them in our example). Instead of exact times, we consider their expected values. For every path, we compute the total time of its realization as the sum of the means of durations of activities lying on the path. We sum also individual variances in order to achieve the total variance on the path. The path with the maximal mean of total time is the critical path, in our example it is A, C, D, E, F, H.

Method PERT - example

critical path	a_{ij}	m_{ij}	b_{ij}	μ_{ij}	σ^2_{ij}
A(1,2)	5	8	10	7,83	0,69
C(2,3)	1	2	5	2,33	0,44
D(3,4)	2	4	6	4,00	0,44
E(4,5)	7	10	14	10,17	1,36
F(5,6)	10	12	13	11,83	0,25
H(6,8)	3	4	6	4,17	0,25
$\sum$				$40,33$	$3,43$

Method PERT - example

critical path	a_{ij}	m_{ij}	b_{ij}	μ_{ij}	σ^2_{ij}
A(1,2)	5	8	10	7,83	0,69
C(2,3)	1	2	5	2,33	0,44
D(3,4)	2	4	6	4,00	0,44
E(4,5)	7	10	14	10,17	1,36
F(5,6)	10	12	13	11,83	0,25
H(6,8)	3	4	6	4,17	0,25
$\sum$				$40,33$	$3,43$

Total project duration T is a random variable with the mean $\mu(T) = 40,33$ weeks and the variance $\sigma^2(T) = 3,43$ weeks. For projects with a large number of activities, probabilistic computations are performed under the assumption that their durations are independent random variables. According to the central limit theorem, the probability distribution of T, which is the sum of independent variables t_{ij}, converges to the normal distribution $N(\mu(T), \sigma^2(T))$.

Probability of finishing in the planned time T_{pl}

This probability is computed from the cumulative distribution function $\phi(u)$ of standard normal distribution $N(0;1)$. First we have to standardize random variable T according to the formula $U = \frac{T-\mu(T)}{\sigma(T))} \sim N(0;1)$. So

$$P(T \leq T_{pl}) = \phi\left(\frac{T_{pl}-\mu(T)}{\sigma(T)}\right).$$

Probability of finishing in the planned time T_{pl}

This probability is computed from the cumulative distribution function $\phi(u)$ of standard normal distribution $N(0;1)$. First we have to standardize random variable T according to the formula $U = \frac{T-\mu(T)}{\sigma(T))} \sim N(0;1)$. So

$$P(T \leq T_{pl}) = \phi\left(\frac{T_{pl}-\mu(T)}{\sigma(T)}\right).$$

- If the planned realization time is shorter than the mean duration of the project ($T_{pl} \leq \mu(T)$), then the argument of function $\phi(u)$ is negative. We can determine its value from the tables of standard normal cdf using formula $\phi(u) = 1 - \phi(-u)$. The probability of completing the project in time T_{pl} is less than 50%.
- If the planned realization time is equal to the mean duration of the project ($T_{pl} = \mu(T)$), the probability of completing the project in time is exactly 50%.
- If the planned realization time is longer than mean duration of the project ($T_{pl} \geq \mu(T)$), the probability of completing the project in time is greater than 50%.

Determining the time of project termination in time T_r for a given risk r

This problem can be solved using tables of standard normal cdf $\phi(u)$. If the risk r is given in per cent, we have to find a quantile $u_{1-r/100}$. Corresponding time T_r is derived from the formula $u_{1-r/100} = \frac{T_r - \mu(T)}{\sigma(T)}$: after some algebra, we have $T_r = \mu(T) + \sigma(T) \cdot u_{1-r/100}$

Determining the time of project termination in time T_r for a given risk r

This problem can be solved using tables of standard normal cdf $\phi(u)$. If the risk r is given in per cent, we have to find a quantile $u_{1-r/100}$. Corresponding time T_r is derived from the formula $u_{1-r/100} = \frac{T_r - \mu(T)}{\sigma(T)}$: after some algebra, we have $T_r = \mu(T) + \sigma(T) \cdot u_{1-r/100}$

Example : Let's consider the previous example with expected project duration equal to 40,33 weeks and variance of 3,43.

Determining the time of project termination in time T_r for a given risk r

This problem can be solved using tables of standard normal cdf $\phi(u)$. If the risk r is given in per cent, we have to find a quantile $u_{1-r/100}$. Corresponding time T_r is derived from the formula $u_{1-r/100} = \frac{T_r - \mu(T)}{\sigma(T)}$: after some algebra, we have $T_r = \mu(T) + \sigma(T) \cdot u_{1-r/100}$

Example : Let's consider the previous example with expected project duration equal to 40,33 weeks and variance of 3,43.

1. Find the probability of completing the project in 42 weeks.

Determining the time of project termination in time T_r for a given risk r

This problem can be solved using tables of standard normal cdf $\phi(u)$. If the risk r is given in per cent, we have to find a quantile $u_{1-r/100}$. Corresponding time T_r is derived from the formula $u_{1-r/100} = \frac{T_r - \mu(T)}{\sigma(T)}$: after some algebra, we have $T_r = \mu(T) + \sigma(T) \cdot u_{1-r/100}$

Example : Let's consider the previous example with expected project duration equal to 40,33 weeks and variance of 3,43.

1. Find the probability of completing the project in 42 weeks.

   Solution: $P(T \leq 42) = \phi\left(\frac{42-40{,}33}{\sqrt{3{,}43}}\right) = \phi(0,90)$. Using statistical software or tables we find the probability $\phi(0,90) = 0,8159$. The probability of completing the project no later than in 42 weeks is 81,59%.

Determining the time of project termination in time T_r for a given risk r

This problem can be solved using tables of standard normal cdf $\phi(u)$. If the risk r is given in per cent, we have to find a quantile $u_{1-r/100}$. Corresponding time T_r is derived from the formula $u_{1-r/100} = \frac{T_r - \mu(T)}{\sigma(T)}$: after some algebra, we have $T_r = \mu(T) + \sigma(T) \cdot u_{1-r/100}$

Example : Let's consider the previous example with expected project duration equal to 40,33 weeks and variance of 3,43.

1. Find the probability of completing the project in 42 weeks.

   Solution: $P(T \leq 42) = \phi\left(\frac{42-40{,}33}{\sqrt{3{,}43}}\right) = \phi(0,90)$. Using statistical software or tables we find the probability $\phi(0,90) = 0,8159$. The probability of completing the project no later than in 42 weeks is 81,59%.
2. Find time which won't be exceeded with the risk of 20%.

Determining the time of project termination in time T_r for a given risk r

This problem can be solved using tables of standard normal cdf $\phi(u)$. If the risk r is given in per cent, we have to find a quantile $u_{1-r/100}$. Corresponding time T_r is derived from the formula $u_{1-r/100} = \frac{T_r - \mu(T)}{\sigma(T)}$: after some algebra, we have $T_r = \mu(T) + \sigma(T) \cdot u_{1-r/100}$

Example : Let's consider the previous example with expected project duration equal to 40,33 weeks and variance of 3,43.

1. Find the probability of completing the project in 42 weeks.
   Solution: $P(T \leq 42) = \phi\left(\frac{42-40{,}33}{\sqrt{3{,}43}}\right) = \phi(0,90)$. Using statistical software or tables we find the probability $\phi(0,90) = 0,8159$. The probability of completing the project no later than in 42 weeks is 81,59%.
2. Find time which won't be exceeded with the risk of 20%.
   Solution: The 20% risk corresponds to 80% probability of meeting the deadline. We look up a quantile for this probability $u_{0,8} = 0,84$ and substitute it to the formula $T_r = 40,33 + \sqrt{3,43} \cdot 0,84 = 41,88$. So we can assume that project terminates no later than in 41,88 weeks with 20% risk of breaking this deadline.

Time - cost project analysis

Methods CPM and PERT take into account only the time dimension of the project, whereas optimal time schedule is not necessarily the cheapest one. The basic criterion of project effectiveness is usually represented by realization costs, which are closely related to the project duration:

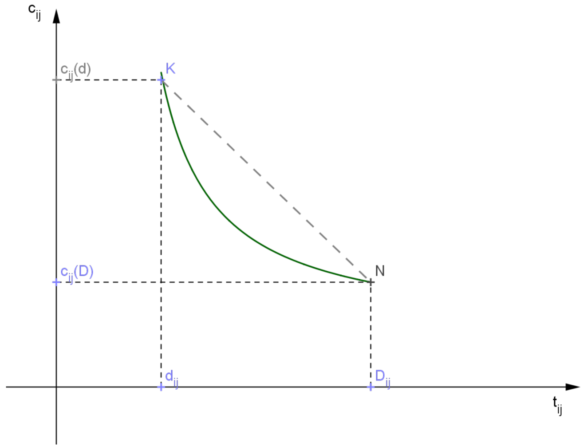
- Indirect costs are connected to the project as a whole (operational costs, penalties, etc). They are increasing function of project duration (we usually suppose linearity).
- Direct costs connected to individual activities (material, labour costs, etc).
- By summing indirect and direct costs of all activities we get total costs.

We will denote by c_{ij} direct costs of activity (i, j) realized in time t_{ij}. We will assume that they are a linear function of t_{ij} (it is a non-increasing function; we can shorten the realization time only with additional costs). Following bounds are considered:

D_{ij} ... normal time for the activity (i, j), corresponding to minimal costs $c_{ij}(D)$

d_{ij} ... crash time for the activity (i, j), corresponding to maximal costs $c_{ij}(d)$.

Time - cost project analysis: direct costs



The line *KN* is the approximation of direct costs according to realization time of activity (i, j). Its analytical equation is: $c_{ij} = b_{ij} - a_{ij} t_{ij}$, where $a_{ij} = \frac{c_{ij}(d) - c_{ij}(D)}{D_{ij} - d_{ij}}$, $b_{ij} = a_{ij} d_{ij} + c_{ij}(d)$

Minimization of direct costs for a given project duration

Total direct costs of the project can be expressed by the equation:
$c_P = \sum_{(i,j) \in P} (b_{ij} - a_{ij} t_{ij})$.
The coefficient a_{ij} is the slope of the line connecting points of normal and crash time.

Minimization of direct costs for a given project duration

Total direct costs of the project can be expressed by the equation:
$c_P = \sum_{(i,j)\in P}(b_{ij} - a_{ij}t_{ij})$.
The coefficient a_{ij} is the slope of the line connecting points of normal and crash time.
Direct costs c_P of realization in time T fulfill the condition:
$\sum_{(i,j)\in P} c_{ij}(D) \leq c_P \leq \sum_{(i,j)\in P} c_{ij}(d)$
These costs can be lowered without delaying whole project by prolonging the duration of non-critical activities till the normal time of activity so that their time slack would diminish.

Minimization of direct costs for a given project duration - example

Example : Minimize costs of the project described by the table (time slacks are labeled by VR_{ij} and critical activities are highlighted by red colour).

(i,j)	t_{ij}	d_{ij}	D_{ij}	$c_{ij}(d)$	$c_{ij}(D)$	a_{ij}	VR_{ij}	c_{ij}
(1,2)	3	3	4	23	20	3	0	23
(1,3)	4	3	5	17	15	1		16
(2,5)	3	2	5	16	10	2	5	14
(3,4)	3	2	4	26	22	2	0	24
(3,5)	7	5	7	38	30	4		30
(4,5)	1	1	1	10	10	-	3	10
$\sum$				130	107			117

Minimization of direct costs for a given project duration - example

Example : Minimize costs of the project described by the table (time slacks are labeled by VR_{ij} and critical activities are highlighted by red colour).

(i,j)	t_{ij}	d_{ij}	D_{ij}	$c_{ij}(d)$	$c_{ij}(D)$	a_{ij}	VR_{ij}	c_{ij}
(1,2)	3	3	4	23	20	3	0	23
(1,3)	4	3	5	17	15	1		16
(2,5)	3	2	5	16	10	2	5	14
(3,4)	3	2	4	26	22	2	0	24
(3,5)	7	5	7	38	30	4		30
(4,5)	1	1	1	10	10	-	3	10
$\sum$				130	107			117

Realization of the project is associated with direct costs equal to 117 cost units (CU). These costs can be reduced by prolonging the duration of non-critical activity (2,5) by 2 time units. This would decrease total costs by 4 CU, so we have a total of 117 – 4 = 113 CU. We continue prolonging the duration of non-critical activities by $\Delta t_{ij} = min(VR_{ij}; D_{ij} - t_{ij})$ and we prefer activities with a higher slope a_{ij}.

Optimal duration of the project

From the effectiveness point of view, such duration is optimal that minimizes total costs (the sum of direct and indirect costs). We already know that direct costs are decreasing and indirect costs increasing function of time.

Optimal duration of the project

From the effectiveness point of view, such duration is optimal that minimizes total costs (the sum of direct and indirect costs). We already know that direct costs are decreasing and indirect costs increasing function of time. The optimal time can be found by shortening duration of critical activities (we prefer activities with the lowest cost slope) until crash time is achieved. We have to be aware of the possibility that another critical path can be formed.

Optimal duration of the project

From the effectiveness point of view, such duration is optimal that minimizes total costs (the sum of direct and indirect costs). We already know that direct costs are decreasing and indirect costs increasing function of time. The optimal time can be found by shortening duration of critical activities (we prefer activities with the lowest cost slope) until crash time is achieved. We have to be aware of the possibility that another critical path can be formed. We can illustrate the procedure on the above example. The initial row represents situation when all activities are realized in normal time. The other rows show the effect of shortening critical activities.

T	(i,j)	t_{ij}	Δc_{ij}	IC	DC	TC=DC+IC
12	-	-	-	60	107	167
11	(1,3)	4	1	55	108	163
10	(1,3)	3	1	50	109	159
9	(3,5)	6	4	45	113	158
8	(3,5)	5	4	40	117	157
	(2,5)	4	2			

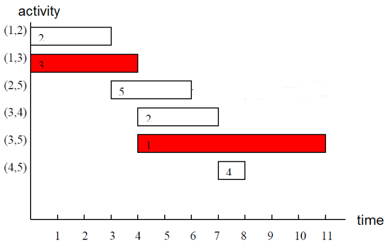
In the last row we can see that new activity has become critical.

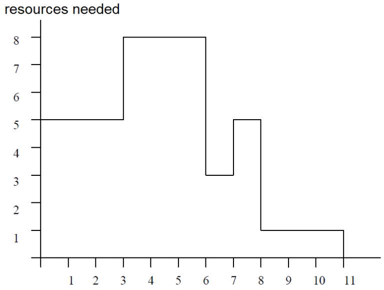
Time and resources project analysis

Every project depends on resources such as labour, material, machines, etc. Managers make effort to distribute resources evenly throughout the project duration. In some cases, the generation of capacity peaks results in a shortage of resources (its need exceeds its available quantity). The summation line graphically expresses aggregate resource requirements at any time of the project, assuming that each activity begins at its earliest possible start. The summation line changes its course at the moments when an activity begins or ends.

Time and resources project analysis - example

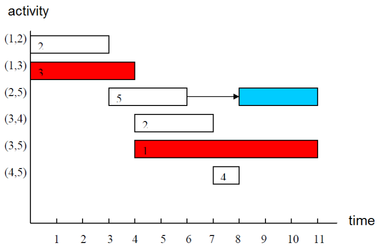
Gantt chart for the problem from the previous example (numbers in the frames express demand for resources) and its summation line.

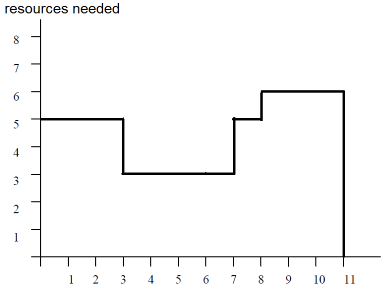




Time and resources project analysis - example

We can reduce the demand for resources between weeks 3 and 6 by exploiting time slack of activity (2,5) and starting this activity in week 8 instead of week 3 (latest start instead of earliest start).





Remark : If we prolong the time of non-critical activities, it influences the need for resources in the course of time.

Multiple-criteria decision making (MCDM)

In real decision situations, it is often important to consider more optimization criteria. However, these are seldom consistent, so it is not possible to find the solution that will be the best with respect to all criteria. The problems of multi-criteria decision-making are divided according to the way decisions are determined.

- Problems with a finite set of decision alternatives, Multiple-criteria evaluation problems and
- Multi-objective (linear) programming problems with infinitely many alternatives defined by a set of constraints.

Multiple-criteria evaluation problems

We have a set of alternatives $X = \{X_1, \dots, X_n\}$ and criteria of evaluation $Y_1, \dots, Y_k$. Each alternative is assigned a vector of criteria values that can be arranged as rows of a criteria matrix.

$$\begin{array}{c|cccc}
 & Y_1 & Y_2 & \dots & Y_k \\
\hline
X_1 & y_{11} & y_{12} & \dots & y_{1k} \\
X_2 & y_{21} & y_{22} & \dots & y_{2k} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
X_n & y_{n1} & y_{n2} & \dots & y_{nk}
\end{array}$$

Multiple-criteria evaluation problems

We have a set of alternatives $X = \{X_1, \dots, X_n\}$ and criteria of evaluation $Y_1, \dots, Y_k$. Each alternative is assigned a vector of criteria values that can be arranged as rows of a criteria matrix.

$$\begin{array}{c|cccc} & Y_1 & Y_2 & \dots & Y_k \\ \hline X_1 & y_{11} & y_{12} & \dots & y_{1k} \\ X_2 & y_{21} & y_{22} & \dots & y_{2k} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & y_{n1} & y_{n2} & \dots & y_{nk} \end{array}$$

Some decision problems deal with either maximization and minimization criteria (e.g., when evaluating the economics of EU countries: the GDP per capita is maximization criteria and unemployment is minimization criteria). Since some methods require that all criteria are of the same type, it is often necessary to transform them. Application area of MCDM is very broad: tenders, ranking of companies, products or services, etc.

Multiple-criteria evaluation problems - example

Methods of multiple-criteria evaluation will be explained in the example of a tablet selection (from Tomáš Šubrt et al .: Ekonomicko - matematické metody): The user has defined five relevant aspects according to which the individual tablets will be evaluated: price [CZK], RAM memory [MB], battery life [h], weight [g], and the fifth criterion (combination of OS, processor, and display size). Five specific T1 - T5 tablets are considered, let us give an overview of their characteristics:

	price	RAM	battery life	weight	OS, processor, display
T1	12000	1000	9,5	680	Android 3.0,Tegra 1 GHz, 10,1”
T2	12000	1000	10	600	Apple 1054,1 GHz Dual, 9,7”
T3	5000	512	7	380	Android 2.2, 1 GHz, 7”
T4	20000	4000	3	1160	Windows 7,iCore5 1 GHz, 12,1”
T5	5000	256	4	400	Android 2.1, 800 MHz, 7”

The first four criteria are quantitative (price and weight are minimization criteria, others are maximization); for the fifth criterion, we have at least ordinal information, i.e. the ranking of the tablets according to an expert: 1,3,4,2,5.

Multiple-criteria evaluation problems

The basic objectives of MCDM analysis include:

- selection of one, the so-called compromise alternative (e.g. elections, etc.)
- ordering alternatives (e.g. quality of products - charts for consumers)
- classification of alternatives (e.g. accepted/unaccepted or bank rating, etc.)

The relation between alternatives can be:

- the alternative X_i dominates the alternative X_j if $(y_{i1}, \ldots, y_{ik}) \geq (y_{j1}, \ldots, y_{jk})$, but they are not the same.
- the alternative X_j dominates the alternative X_i
- the alternatives X_i, X_j are mutually non-dominated.

Let us say that the X_i alternative is non-dominated if there is no other alternative that dominates it. Apparently, the compromise alternative must always be non-dominated. Next, we define the terms basal (ideal) alternative, which means a generally non-existent alternative acquiring the worst (or the best) values according to all criteria.

MCDM - dominance of alternatives, example

Find non-dominated alternatives and construct a basal and an ideal alternative in a tablet selection problem.

	price	RAM	battery life	weight	OS, processor, display
Tablet 1	12000	1000	9,5	680	1
Tablet 2	12000	1000	10	600	3
Tablet 3	5000	512	7	380	4
Tablet 4	20000	4000	3	1160	2
Tablet 5	5000	256	4	400	5

The best values for individual criteria are highlighted in the table.

MCDM - dominance of alternatives, example

Find non-dominated alternatives and construct a basal and an ideal alternative in a tablet selection problem.

	price	RAM	battery life	weight	OS, processor, display
Tablet 1	12000	1000	9,5	680	1
Tablet 2	12000	1000	10	600	3
Tablet 3	5000	512	7	380	4
Tablet 4	20000	4000	3	1160	2
Tablet 5	5000	256	4	400	5

The best values for individual criteria are highlighted in the table. The ideal alternative would therefore be unrealistic (5000 CZK, 4000 MB, 10 hours, 380g and 1st rank according to the expert). Similarly basal alternative doesn't exist (20000 CZK, 256 MB, 3 hours, 1160g and 5th order by the expert). Nearly all variants are non-dominated, with the exception of Tablet 5 (worse than Tablet 3 in every aspect).

MCDM - information on the preference of criteria

For most MCDM methods, it is necessary for the decision maker to express his preferences relating the individual criteria. These preferences can be determined using

- aspiration levels of criteria, i.e. by setting minimum values to be reached for the individual maximization criteria (or maximum values for the minimization criteria). Criteria preference is thus expressed indirectly if we set a higher aspiration level for more important criteria.
- ranking of criteria (ordinal information on criteria)
- weights of criteria: $\mathbf{v} = (v_1, \ldots, v_k)\ \sum v_i = 1,\ v_i > 0,\ i = 1, \ldots, k.$ (cardinal information on criteria)
- substitution rates of the criteria on which the compensation MCDM methods are based

MCDM - estimating weights of criteria

Obtaining weights from the decision maker directly in numerical form is problematic, so it is appropriate to facilitate his situation using a simple tool.

- The ranking method requires only criteria ranking from the least important to the most important. Thus, the assigned values p_i will be $1, \ldots, k$, and the weight estimates can be obtained by normalizing them: $v_i = \frac{p_i}{\sum_{i=1}^{k} p_i}$.
- In the scoring method, the decision maker assigns p_i points of a pre-selected scale to each criterion. The conversion of points into weights is the same as above.
- Fuller's triangle is based on pair comparison of criteria. The individual criterion is assigned p_i points according to the number of criteria that are less or equally important than the i-th criterion (there are $k(k-1)/2$ pairs and they can be arranged in a triangular scheme, hence the name of the method).
- A slightly more sophisticated approach is the Saaty's method, in which the pairs of criteria are assigned the number $s_{ij} \approx \frac{v_i}{v_j}$ that estimates how many times more important is the i-th criterion than the j-th one. The matrix $S = (s_{ij})_{i,j=1,\ldots,k}$ is called the Saaty's matrix.

Saaty's method for estimating weights of criteria

The Saaty's method allows us to formulate preferences verbally and then express them numerically using a scale:

- criteria Y_i a Y_j are of the same importance, $s_{ij} = s_{ji} = 1$,
- criterion Y_i is slightly more important than Y_j, $s_{ij} = 3,\ s_{ji} = 1/3$,
- criterion Y_i is more important than Y_j, $s_{ij} = 5,\ s_{ji} = 1/5$,
- criterion Y_i is much more important than Y_j, $s_{ij} = 7,\ s_{ji} = 1/7$,
- criterion Y_i is absolutely more important than Y_j, $s_{ij} = 9,\ s_{ji} = 1/9$.

If the scale is not sufficient, intermediate steps 2,4,6,8 can also be used. If the Saaty's matrix is consistent, we can determine the weights as the solution of the system $\frac{v_i}{v_j} = s_{ij}, i, j = 1, \ldots, k,\ \sum v_i = 1$. If the matrix is not consistent, the system has no solution and weights can be estimated for example by normalizing geometrical means of rows of the matrix S:

$$p_i = \sqrt[k]{\prod_{j=1}^{k} s_{ij}}.$$

MCDM - estimating weights of criteria, example

We can show different ways of determining the criteria weights for the tablet selection problem with decision-maker preferences „1. price, 2. RAM, 3. expert’s opinion, 4. battery life, 5. weight (size)“. The simplest ranking method would assign criteria weights $\mathbf{v} = (\frac{5}{15}, \frac{4}{15}, \frac{2}{15}, \frac{1}{15}, \frac{3}{15})$.

MCDM - estimating weights of criteria, example

We can show different ways of determining the criteria weights for the tablet selection problem with decision-maker preferences „1. price, 2. RAM, 3. expert's opinion, 4. battery life, 5. weight (size)". The simplest ranking method would assign criteria weights $\mathbf{v} = (\frac{5}{15}, \frac{4}{15}, \frac{2}{15}, \frac{1}{15}, \frac{3}{15})$.
Another way is to express the preference of a criterion in a given row versus the criteria in the individual columns by marking a value of 1 in the Fuller's triangle:

	price	RAM	battery	size	expert	scores	weights
price		1	1	1	1		
RAM			1	1	1		
battery				1	0		
size					0		
expert							

We complete the table by filling the lower triangle symmetrically with opposite values ($f_{ij} = 1 - f_{ji},\ i \neq j$), because we need them for the computation of scores.

MCDM - estimating weights of criteria, example

We can show different ways of determining the criteria weights for the tablet selection problem with decision-maker preferences „1. price, 2. RAM, 3. expert's opinion, 4. battery life, 5. weight (size)“. The simplest ranking method would assign criteria weights $\mathbf{v} = (\frac{5}{15}, \frac{4}{15}, \frac{2}{15}, \frac{1}{15}, \frac{3}{15})$.
Another way is to express the preference of a criterion in a given row versus the criteria in the individual columns by marking a value of 1 in the Fuller's triangle:

	price	RAM	battery	size	expert	scores	weights
price		1	1	1	1		
RAM	0		1	1	1		
battery	0	0		1	0		
size	0	0	0		0		
expert	0	0	1	1			

We calculate scores as the sum of the number of preferences in the given rows.

MCDM - estimating weights of criteria, example

We can show different ways of determining the criteria weights for the tablet selection problem with decision-maker preferences „1. price, 2. RAM, 3. expert's opinion, 4. battery life, 5. weight (size)". The simplest ranking method would assign criteria weights $\mathbf{v} = (\frac{5}{15}, \frac{4}{15}, \frac{2}{15}, \frac{1}{15}, \frac{3}{15})$.
Another way is to express the preference of a criterion in a given row versus the criteria in the individual columns by marking a value of 1 in the Fuller's triangle:

	price	RAM	battery	size	expert	scores	weights
price		1	1	1	1	4	
RAM	0		1	1	1	3	
battery	0	0		1	0	1	
size	0	0	0		0	0	
expert	0	0	1	1		2	

We now calculate the weights by dividing the scores by their total sum, i.e. by 10.

MCDM - estimating weights of criteria, example

We can show different ways of determining the criteria weights for the tablet selection problem with decision-maker preferences „1. price, 2. RAM, 3. expert's opinion, 4. battery life, 5. weight (size)“. The simplest ranking method would assign criteria weights $\mathbf{v} = (\frac{5}{15}, \frac{4}{15}, \frac{2}{15}, \frac{1}{15}, \frac{3}{15})$.
Another way is to express the preference of a criterion in a given row versus the criteria in the individual columns by marking a value of 1 in the Fuller's triangle:

	price	RAM	battery	size	expert	scores	weights
price		1	1	1	1	4	0,4
RAM	0		1	1	1	3	0,3
battery	0	0		1	0	1	0,1
size	0	0	0		0	0	0
expert	0	0	1	1		2	0,2

Unfortunately, the weight of the least important criterion would be zero if the points are divided consistently. We can modify the process by adding ones to diagonal elements (as if each criterion was preferred to itself).

MCDM - estimating weights of criteria, example

We can show different ways of determining the criteria weights for the tablet selection problem with decision-maker preferences „1. price, 2. RAM, 3. expert's opinion, 4. battery life, 5. weight (size)“. The simplest ranking method would assign criteria weights $\mathbf{v} = (\frac{5}{15}, \frac{4}{15}, \frac{2}{15}, \frac{1}{15}, \frac{3}{15})$.
Another way is to express the preference of a criterion in a given row versus the criteria in the individual columns by marking a value of 1 in the Fuller's triangle:

	price	RAM	battery	size	expert	scores	weights
price	1	1	1	1	1	5	$\frac{5}{15}$
RAM	0	1	1	1	1	4	$\frac{4}{15}$
battery	0	0	1	1	0	2	$\frac{2}{15}$
size	0	0	0	1	0	1	$\frac{1}{15}$
expert	0	0	1	1	1	3	$\frac{3}{15}$

We get the same weights as by the ranking method.

MCDM - estimating weights of criteria, example

Let us show one of the possible user evaluations of the Saaty's matrix for the same decision problem.

	price	RAM	battery	size	expert	b_i	v_i
price	1	4	2	9	2		
RAM	1/4	1	1/2	3	1/2		
battery	1/2	2	1	7	1		
size	1/9	1/3	1/7	1	1/7		
expert	1/2	2	1	7	1		

Matrix is consistent enough (its consistency index is equal to 0,005 which is less than the limit 0,1). We compute geometrical means of rows b_i and determine weights v_i.

MCDM - estimating weights of criteria, example

Let us show one of the possible user evaluations of the Saaty's matrix for the same decision problem.

	price	RAM	battery	size	expert	b_i	v_i
price	1	4	2	9	2	2,7	0,41
RAM	1/4	1	1/2	3	1/2	0,72	0,11
battery	1/2	2	1	7	1	1,48	0,22
size	1/9	1/3	1/7	1	1/7	0,24	0,04
expert	1/2	2	1	7	1	1,48	0,22

Saaty's method gives more differentiated weights than the other methods.

MCDM - classification of methods

There are many approaches to solving MCDM problems, we will only mention the simpler ones. The methods can be classified according to the type of preference information for criteria and alternatives, see the following brief overview.

Preference information on alternatives					
aspiration levels	ordinal information	cardinal information			
		utility function	distance of alternative from bazal or ideal	preference relation	marginal rate of substitution

Methods					
PRIAM	Lexicographic ORESTE Permutation	WSA	TOPSIS PROMETHEE ELECTRE	AHP	Compensation

MCDM - methods not using weights of criteria

If there are no preferences between the criteria or the criteria are of equal importance, we can use a simple order of values for each criterion to select the compromise alternative (if there are the same values, the average order is assigned). We choose the alternative that has the lowest total ranking across all criteria.
In the tablets example, we have already rankings in the last column (expert opinion). In the other columns, we replace the individual values with the ranking within the column and compute row totals:

	price	RAM	battery life	size	OS, processor display	sum
Tablet 1	3	2,5	2	4	1	12,5
Tablet 2	3	2,5	1	3	3	12,5
Tablet 3	1	4	3	1	4	13
Tablet 4	5	1	5	5	2	18
Tablet 5	1	5	4	2	5	17

Tablet 1 and tablet 2 would be chosen by the above described approach as most suitable.

MCDM - methods not using weights of criteria

Another method not requiring apriori weights of criteria is the PRIAM method. Alternatives are marked as acceptable or unacceptable according to given setting of aspiration levels for individual criteria. There may be situations where no alternative is acceptable, then some constraint needs to be relaxed. Conversely, the number of acceptable alternatives can be reduced by tightening some of the aspiration levels. The PRIAM method is an interactive procedure of gradual adaptation of aspiration limits to achieve the required number of acceptable alternatives (in the extreme case it would be only one compromise alternative).

MCDM - methods not using weights of criteria

Another method not requiring apriori weights of criteria is the PRIAM method. Alternatives are marked as acceptable or unacceptable according to given setting of aspiration levels for individual criteria. There may be situations where no alternative is acceptable, then some constraint needs to be relaxed. Conversely, the number of acceptable alternatives can be reduced by tightening some of the aspiration levels. The PRIAM method is an interactive procedure of gradual adaptation of aspiration limits to achieve the required number of acceptable alternatives (in the extreme case it would be only one compromise alternative).

	price	RAM	battery life	size	OS, processor, display	acceptable
Tablet 1	12000	1000	9,5	680	1	YES
Tablet 2	12000	1000	10	600	3	YES
Tablet 3	5000	512	7	380	4	YES
Tablet 4	20000	4000	3	1160	2	YES
Tablet 5	5000	256	4	400	5	YES

If wet set initial vector of aspiration levels on the values of basal alternative $z^0 = (20000, 256, 3, 1160, 5)$ then all tablets are acceptable.

MCDM - methods not using weights of criteria

Another method not requiring apriori weights of criteria is the PRIAM method. Alternatives are marked as acceptable or unacceptable according to given setting of aspiration levels for individual criteria. There may be situations where no alternative is acceptable, then some constraint needs to be relaxed. Conversely, the number of acceptable alternatives can be reduced by tightening some of the aspiration levels. The PRIAM method is an interactive procedure of gradual adaptation of aspiration limits to achieve the required number of acceptable alternatives (in the extreme case it would be only one compromise alternative).

	price	RAM	battery life	size	OS, processor, display	acceptable
Tablet 1	12000	1000	9,5	680	1	YES
Tablet 2	12000	1000	10	600	3	YES
Tablet 3	5000	512	7	380	4	YES
Tablet 4	20000	4000	3	1160	2	NO
Tablet 5	5000	256	4	400	5	YES

If we tighten up requirement on price not to exceed 12000 CZK, we get $z^1 = (12000, 256, 3, 1160, 5)$, so the tablet 4 is not acceptable.

MCDM - methods not using weights of criteria

Another method not requiring apriori weights of criteria is the PRIAM method. Alternatives are marked as acceptable or unacceptable according to given setting of aspiration levels for individual criteria. There may be situations where no alternative is acceptable, then some constraint needs to be relaxed. Conversely, the number of acceptable alternatives can be reduced by tightening some of the aspiration levels. The PRIAM method is an interactive procedure of gradual adaptation of aspiration limits to achieve the required number of acceptable alternatives (in the extreme case it would be only one compromise alternative).

	price	RAM	battery life	size	OS, processor, display	acceptable
Tablet 1	12000	1000	9,5	680	1	YES
Tablet 2	12000	1000	10	600	3	YES
Tablet 3	5000	512	7	380	4	NO
Tablet 4	20000	4000	3	1160	2	NO
Tablet 5	5000	256	4	400	5	NO

By tightening up requirement on battery and expert opinion, $z^2 = (12000, 256, 7, 1160, 3)$, we eliminate also tablets 3 and 5.

MCDM - methods not using weights of criteria

Another method not requiring apriori weights of criteria is the PRIAM method. Alternatives are marked as acceptable or unacceptable according to given setting of aspiration levels for individual criteria. There may be situations where no alternative is acceptable, then some constraint needs to be relaxed. Conversely, the number of acceptable alternatives can be reduced by tightening some of the aspiration levels. The PRIAM method is an interactive procedure of gradual adaptation of aspiration limits to achieve the required number of acceptable alternatives (in the extreme case it would be only one compromise alternative).

	price	RAM	battery life	size	OS, processor, display	acceptable
Tablet 1	12000	1000	9,5	680	1	NO
Tablet 2	12000	1000	10	600	3	YES
Tablet 3	5000	512	7	380	4	NO
Tablet 4	20000	4000	3	1160	2	NO
Tablet 5	5000	256	4	400	5	NO

When we require also smaller size of a tablet $z^3 = (12000, 256, 7, 600, 3)$, then only the tablet 2 remains acceptable.

MCDM - methods using ordinal preference of criteria

The most common and probably the simplest method in a class of procedures requiring only ordinal information on the criteria is the lexicographical method. We follow the most important criterion, and if the best option under this criterion is the only one, it is chosen as a compromise. If the best value is reached by multiple alternatives, we choose the one that has better ranking according to the second most important criterion, etc.

MCDM - methods using ordinal preference of criteria

The most common and probably the simplest method in a class of procedures requiring only ordinal information on the criteria is the lexicographical method. We follow the most important criterion, and if the best option under this criterion is the only one, it is chosen as a compromise. If the best value is reached by multiple alternatives, we choose the one that has better ranking according to the second most important criterion, etc.
By using the lexicographic method for selection of a tablet, if the first is the price criterion and the second criterion is battery life, we choose tablet 3 (Tablets 3 and 5 are the cheapest and battery of tablet 3 lasts 7 hours whereas tablet 5 lasts only 4 hours):

	price	RAM	battery life	size	OS, processor, display	ranking
Tablet 1	12000	1000	9,5	680	1	4
Tablet 2	12000	1000	10	600	3	3
Tablet 3	5000	512	7	380	4	1
Tablet 4	20000	4000	3	1160	2	5
Tablet 5	5000	256	4	400	5	2

MCDM - methods using weights of criteria

There are three basic categories of approaches to evaluation of alternatives using weighting criteria, namely:

- maximization of utility
- minimization of distance from an ideal alternative
- the preference relation

MCDM - methods using weights of criteria

There are three basic categories of approaches to evaluation of alternatives using weighting criteria, namely:

- maximization of utility
- minimization of distance from an ideal alternative
- the preference relation

We will introduce at least one representative from each group of methods. The first option is to quantify the utility of alternatives on a scale from 0 to 1. To express the overall benefit of the alternative, it is necessary to express the partial functions of utility u_j according to the individual criteria $j = 1, \ldots, k$.
Thus, the values y_{ij} are replaced with the values $u_{ij} = u_j(y_{ij}),\ j = 1, \ldots, k.$
Utility functions are normalized so that the ideal alternative has a partial utility according to all criteria equal to 1 and basal alternative all zeros. We distinguish three types of benefits:

1. linear (utility changes proportionally to the value of criteria)
2. progressive (rate of change in utility increases with the growing value of criteria)
3. regressive (rate of change in utility decreases with the growing value of criteria)

MCDM - methods using utility function

The method of weighted sum (WSA) is based on a linear utility function with the scale of 0 to 1. We denote by D_j minimal and by H_j maximal value in the column Y_j (i.e. for the j-th criterion), then all values y_{ij} in the column Y_j can be standardized using the formula

$y'_{ij} = \frac{y_{ij} - D_j}{H_j - D_j}$

for maximization type of criteria. For minimization criteria we use the formula

$y'_{ij} = \frac{H_j - y_{ij}}{H_j - D_j}$.

After this transformation, the worst option in every column is assigned zero value and the best option corresponds to 1. Total utility of the alternative X_i is computed as the weighted sum of standardized values $u(X_i) = \sum_{j=1}^{k} v_j y'_{ij}$.

Finally the alternatives are ranked according to their utilities.

MCDM - method WSA, example

Lets apply WSA on tablet selection problem, we use weights given by Saaty's method:

	price	RAM	battery life	size	OS, processor, display	utility
Tablet 1	12000	1000	9,5	680	1	
Tablet 2	12000	1000	10	600	3	
Tablet 3	5000	512	7	380	4	
Tablet 4	20000	4000	3	1160	2	
Tablet 5	5000	256	4	400	5	
weight	0,41	0,12	0,22	0,03	0,22	
D_j	5000	256	3	380	1	
H_j	20000	4000	10	1160	5	
type	min	max	max	min	min	

We compute standardized values (partial utilities) y'_{ij}.

MCDM - method WSA, example

Lets apply WSA on tablet selection problem, we use weights given by Saaty's method:

	price	RAM	battery life	size	OS, processor, display	utility
Tablet 1	0,53	0,2	0,93	0,62	1	
Tablet 2	0,53	0,2	1	0,72	0,5	
Tablet 3	1	0,07	0,57	1	0,25	
Tablet 4	0	1	0	0	0,75	
Tablet 5	1	0	0,14	0,97	0	
weight	0,41	0,12	0,22	0,03	0,22	
D_j	5000	256	3	380	1	
H_j	20000	4000	10	1160	5	
type	min	max	max	min	min	

Then we aggregate partial utilities for each alternative.

MCDM - method WSA, example

Lets apply WSA on tablet selection problem, we use weights given by Saaty's method:

	price	RAM	battery life	size	OS, processor, display	utility
Tablet 1	0,53	0,2	0,93	0,62	1	0,69
Tablet 2	0,53	0,2	1	0,72	0,5	0,59
Tablet 3	1	0,07	0,57	1	0,25	0,63
Tablet 4	0	1	0	0	0,75	0,29
Tablet 5	1	0	0,14	0,97	0	0,47
weight	0,41	0,12	0,22	0,03	0,22	
D_j	5000	256	3	380	1	
H_j	20000	4000	10	1160	5	
type	min	max	max	min	min	

Maximal utility can be achieved by purchasing the tablet 1 (the second best is the tablet 3).

MCDM - methods based on distance from basal and ideal alternatives

The method TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) tries to find the alternative that is as close as possible to the ideal alternative and at the same time as far as possible from the basal alternative. Let's describe the procedure for the case of maximization criteria. The method consists of the following steps:

1. Normalization: values y_{ij} are transformed to r_{ij} using the formula $r_{ij} = \frac{y_{ij}}{\sqrt{\sum_{i=1}^{n} y_{ij}^2}}$.
2. Weighted criteria matrix $W = (w_{ij})$ is computed as $w_{ij} = v_j r_{ij}$.
3. Column maxima of W form ideal alternative $H = (H_1, \dots, H_k)$ and column minima form basal alternative $D = (D_1, \dots, D_k)$, i.e. $H_j = max_i(w_{ij})$, $D_j = min_i(w_{ij})$. (Values of H and D are different from that used in WSA!)
4. For each alternative we compute its distance from ideal and basal alternative: $d_i^+ = \sqrt{\sum_{j=1}^{k}(w_{ij} - H_j)^2}$, $d_i^- = \sqrt{\sum_{j=1}^{k}(w_{ij} - D_j)^2}$
5. Finally, we assign each alternative the index $c_i = \frac{d_i^-}{d_i^+ + d_i^-}$ and rank alternatives according to descending values of the index.

MCDM - method TOPSIS, example

Lets apply the TOPSIS method on a tablet selection problem for Saaty's weights.

	price	RAM	battery life	size	expert	d_i^+	d_i^-	c_i
T1	0,442	0,234	0,584	0,432	0,135			
T2	0,442	0,234	0,615	0,382	0,405			
T3	0,184	0,120	0,431	0,242	0,539			
T4	0,736	0,934	0,185	0,738	0,270			
T5	0,184	0,06	0,246	0,254	0,674			
v_j	0,41	0,12	0,22	0,03	0,22			
type	min	max	max	min	min			
D_j								
H_j								

Normalized values r_{ij}.

MCDM - method TOPSIS, example

Lets apply the TOPSIS method on a tablet selection problem for Saaty's weights.

	price	RAM	battery life	size	expert	d_i^+	d_i^-	c_i
T1	0,181	0,028	0,129	0,013	0,03			
T2	0,181	0,028	0,135	0,011	0,089			
T3	0,075	0,014	0,095	0,007	0,119			
T4	0,302	0,112	0,041	0,022	0,059			
T5	0,075	0,007	0,054	0,008	0,148			
v_j	0,41	0,12	0,22	0,03	0,22			
type	min	max	max	min	min			
D_j	0,302	0,007	0,041	0,022	0,148			
H_j	0,075	0,112	0,135	0,007	0,030			

Weighted matrix W_{ij}, ideal alternative H and basal alternative D are in bottom rows.

MCDM - method TOPSIS, example

Lets apply the TOPSIS method on a tablet selection problem for Saaty's weights.

	price	RAM	battery life	size	expert	d_i^+	d_i^-	c_i
T1	0,181	0,028	0,129	0,013	0,03	0,135	0,192	0,587
T2	0,181	0,028	0,135	0,011	0,089	0,148	0,166	0,53
T3	0,075	0,014	0,095	0,007	0,119	0,138	0,235	0,63
T4	0,302	0,112	0,041	0,022	0,059	0,248	0,138	0,357
T5	0,075	0,007	0,054	0,008	0,148	0,178	0,227	0,56
v_j	0,41	0,12	0,22	0,03	0,22			
type	min	max	max	min	min			
D_j	0,302	0,007	0,041	0,022	0,148			
H_j	0,075	0,112	0,135	0,007	0,030			

Final step is computation of distances from ideal and basal alternative d_i^+ and d_i^- and the relative index c_i. The smallest value of c_i is obtained for the tablet 3.

MCDM - methods of class ELECTRE

An important group of methods is based on the preference relation between the alternatives. The overall preference is derived from partial preferences by appropriate aggregation procedures. Unfortunately, the result does not have to be transitive, so there is a need for another procedure that determines the overall order of the alternatives. The method ELECTRE I doesn't produce complete ranking but a partition into groups of the so-called effective or ineffective alternatives.

MCDM - methods of class ELECTRE

An important group of methods is based on the preference relation between the alternatives. The overall preference is derived from partial preferences by appropriate aggregation procedures. Unfortunately, the result does not have to be transitive, so there is a need for another procedure that determines the overall order of the alternatives. The method ELECTRE I doesn't produce complete ranking but a partition into groups of the so-called effective or ineffective alternatives.

Assume that all criteria are maximizing. For each pair of alternatives $X_i,\ X_j$, we determine the set

$C_{ij} = \{h \in \{1, \ldots, k\}; y_{ih} \geq y_{jh}\}$

consisting of indexes of those criteria, that doesn't rank X_i worse than X_j.

Additionally, we determine the set

$D_{ij} = \{h \in \{1, \ldots, k\}; y_{ih} < y_{jh}\}$

consisting of indexes of those criteria, that rank X_i worse than X_j

MCDM - methods of class ELECTRE

Weights $v_1, \dots, v_k$ are used for computation of the preference degree for each pair of alternatives according to the formula

$$c_{ij} = \sum_{h \in C_{ij}} v_h$$

Apparently, $c_{ij} \in \langle 0, 1 \rangle$.

MCDM - methods of class ELECTRE

Weights $v_1, \dots, v_k$ are used for computation of the preference degree for each pair of alternatives according to the formula

$$c_{ij} = \sum_{h \in C_{ij}} v_h$$

Apparently, $c_{ij} \in \langle 0, 1 \rangle$.
In the next step, we assign each pair of alternatives dispreference degree by the formula

$$d_{ij} = \frac{max_{h \in D_{ij}} |y_{ih} - y_{jh}|}{max_{h=1,\dots,k} |y_{ih} - y_{jh}|};$$

in the case $D_{ij} = \emptyset$ we define $d_{ij} = 0$. Again $d_{ij} \in \langle 0, 1 \rangle$.

MCDM - methods of class ELECTRE

Weights $v_1, \dots, v_k$ are used for computation of the preference degree for each pair of alternatives according to the formula

$$c_{ij} = \sum_{h \in C_{ij}} v_h$$

Apparently, $c_{ij} \in \langle 0, 1 \rangle$.

In the next step, we assign each pair of alternatives dispreference degree by the formula

$$d_{ij} = \frac{max_{h \in D_{ij}} |y_{ih} - y_{jh}|}{max_{h=1,\dots,k} |y_{ih} - y_{jh}|};$$

in the case $D_{ij} = \emptyset$ we define $d_{ij} = 0$. Again $d_{ij} \in \langle 0, 1 \rangle$.

Suitable constants called preference threshold c^* and dispreference threshold d^* help to decide on the relation between X_i and X_j:

X_i is preferred to X_j, if $c_{ij} \geq c^* \wedge d_{ij} \leq d^*$.

MCDM - methods of class ELECTRE

Weights $v_1, \dots, v_k$ are used for computation of the preference degree for each pair of alternatives according to the formula

$$c_{ij} = \sum_{h \in C_{ij}} v_h$$

Apparently, $c_{ij} \in \langle 0, 1 \rangle$.
In the next step, we assign each pair of alternatives dispreference degree by the formula

$$d_{ij} = \frac{max_{h \in D_{ij}} |y_{ih} - y_{jh}|}{max_{h=1,\dots,k} |y_{ih} - y_{jh}|};$$

in the case $D_{ij} = \emptyset$ we define $d_{ij} = 0$. Again $d_{ij} \in \langle 0, 1 \rangle$.
Suitable constants called preference threshold c^* and dispreference threshold d^* help to decide on the relation between X_i and X_j:

X_i is preferred to X_j, if $c_{ij} \geq c^* \wedge d_{ij} \leq d^*$.

Alternatives that are preferred over at least one another, but at the same time there is no alternative preferred to them are considered to be effective. The result depends on the choice of thresholds, so adjustments in their values can be used to reduce the number of effective alternatives to find "the best"compromise option.

MCDM - method ELECTRE I - example

Determination of the sets C_{ij} for the tablet problem:

C_{ij}	T1	T2	T3	T4	T5
T1		$\{1,2,5\}$	$\{2,3,5\}$	$\{1,3,4,5\}$	$\{2,3,5\}$
T2	$\{1,2,3,4\}$		$\{2,3,5\}$	$\{1,3,4\}$	$\{2,3,5\}$
T3	$\{1,4\}$	$\{1,4\}$		$\{1,4\}$	$\{1,2,3,4,5\}$
T4	$\{2\}$	$\{2,5\}$	$\{2,3,5\}$		$\{2,5\}$
T5	$\{1,4\}$	$\{1,4\}$	$\{1\}$	$\{1,3,4\}$	

MCDM - method ELECTRE I - example

Determination of the sets C_{ij} for the tablet problem:

C_{ij}	T1	T2	T3	T4	T5
T1		$\{1,2,5\}$	$\{2,3,5\}$	$\{1,3,4,5\}$	$\{2,3,5\}$
T2	$\{1,2,3,4\}$		$\{2,3,5\}$	$\{1,3,4\}$	$\{2,3,5\}$
T3	$\{1,4\}$	$\{1,4\}$		$\{1,4\}$	$\{1,2,3,4,5\}$
T4	$\{2\}$	$\{2,5\}$	$\{2,3,5\}$		$\{2,5\}$
T5	$\{1,4\}$	$\{1,4\}$	$\{1\}$	$\{1,3,4\}$	

Again we use Saaty's weights $(0,41;\ 0,12;\ 0,22;\ 0,03;\ 0,22)$
for the determination of preference degrees:

c_{ij}	T1	T2	T3	T4	T5
T1		0,75	0,56	0,88	0,56
T2	0,78		0,56	0,66	0,56
T3	0,44	0,44		0,44	1
T4	0,12	0,34	0,56		0,34
T5	0,44	0,44	0,41	0,66	

MCDM - method ELECTRE I - example

The sets D_{ij} for individual pairs of tablets are

D_{ij}	T1	T2	T3	T4	T5
T1		$\{3,4\}$	$\{1,4\}$	$\{2\}$	$\{1,4\}$
T2	$\{5\}$		$\{1,4\}$	$\{2,5\}$	$\{1,4\}$
T3	$\{2,3,5\}$	$\{2,3,5\}$		$\{2,3,5\}$	$\{\}$
T4	$\{1,3,4,5\}$	$\{1,3,4\}$	$\{1,4\}$		$\{1,3,4\}$
T5	$\{2,3,5\}$	$\{2,3,5\}$	$\{2,3,4,5\}$	$\{2,5\}$	

MCDM - method ELECTRE I - example

The sets D_{ij} for individual pairs of tablets are

D_{ij}	T1	T2	T3	T4	T5
T1		$\{3,4\}$	$\{1,4\}$	$\{2\}$	$\{1,4\}$
T2	$\{5\}$		$\{1,4\}$	$\{2,5\}$	$\{1,4\}$
T3	$\{2,3,5\}$	$\{2,3,5\}$		$\{2,3,5\}$	$\{\}$
T4	$\{1,3,4,5\}$	$\{1,3,4\}$	$\{1,4\}$		$\{1,3,4\}$
T5	$\{2,3,5\}$	$\{2,3,5\}$	$\{2,3,4,5\}$	$\{2,5\}$	

We use standardized values y'_{ij} to complete the table of dispreference degrees:

d_{ij}	T1	T2	T3	T4	T5
T1		0,1/0,5	0,47/0,75	0,8/0,93	0,47/1
T2	0,5/0,5		0,47/0,47	0,8/1	0,47/0,86
T3	0,75/0,75	0,28/0,47		0,93/1	0/0,43
T4	0,93/0,93	1/1	1/1		1/1
T5	1/1	0,86/0,86	0,43/0,43	1/1	

MCDM - method ELECTRE I - example

We set the preference threshold $c^* = 0,5$ and highlight $c_{ij} \geq c^*$.

c_{ij}	T1	T2	T3	T4	T5
T1		0,75	0,56	0,88	0,56
T2	0,78		0,56	0,66	0,56
T3	0,44	0,44		0,44	1
T4	0,12	0,34	0,56		0,34
T5	0,44	0,44	0,41	0,66	

MCDM - method ELECTRE I - example

We set the preference threshold $c^* = 0,5$ and highlight $c_{ij} \geq c^*$.

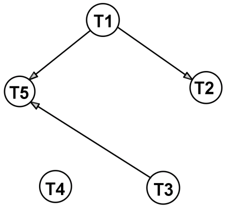
c_{ij}	T1	T2	T3	T4	T5
T1		0,75	0,56	0,88	0,56
T2	0,78		0,56	0,66	0,56
T3	0,44	0,44		0,44	1
T4	0,12	0,34	0,56		0,34
T5	0,44	0,44	0,41	0,66	

We set the preference threshold $d^* = 0,5$ and highlight $d_{ij} \leq d^*$.

d_{ij}	T1	T2	T3	T4	T5
T1		0,2	0,62	0,86	0,47
T2	1		1	0,8	0,55
T3	1	0,6		0,93	0
T4	1	1	1		1
T5	1	1	1	1	

MCDM - method ELECTRE I - example

Preference relation can be represented by a matrix or an oriented graph (it contains only those edges that correspond to pairs satisfying both conditions $c_{ij} \geq c^*$, $d_{ij} \leq d^*$):



As can be seen, alternatives 1 and 3 are effective (corresponding nodes are starting points of some edges, but no edges end in them).

MCDM - methods of class PROMETHEE

These methods are one of the most popular among MCDM procedures. They are based on a pair comparison of alternatives with respect to all criteria. We define for each pair X_i, X_j the preference intensity with respect to Y_h:

$$P_h(X_i, X_j)$$

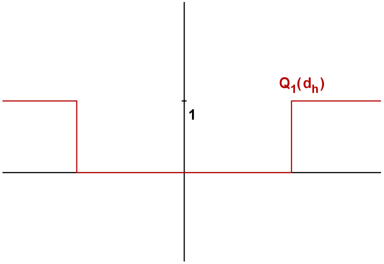
as a function with the range $\langle 0, 1 \rangle$, such that $P_h(X_i, X_j) = 0$ if X_i is not preferred to X_j and $P_h(X_i, X_j) = 1$ in the case it is absolutely preferred.

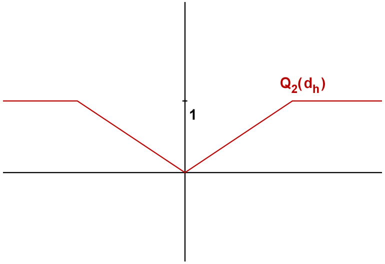
MCDM - methods of class PROMETHEE

These methods are one of the most popular among MCDM procedures. They are based on a pair comparison of alternatives with respect to all criteria. We define for each pair X_i, X_j the preference intensity with respect to Y_h:

$P_h(X_i, X_j)$

as a function with the range $\langle 0, 1\rangle$, such that $P_h(X_i, X_j) = 0$ if X_i is not preferred to X_j and $P_h(X_i, X_j) = 1$ in the case it is absolutely preferred. Transformation Q converting the difference d_h to the preference intensity $P_h(X_i, X_j)$ is called a generalized criterion and may have various shapes, for example, see the figure:





MCDM - methods of class PROMETHEE

For given weights we can compute the global preference index of X_i, X_j by the formula

$$P(X_i, X_j) = \sum_{h=1}^{k} v_h P_h(X_i, X_j).$$

To obtain the final order of alternatives we have to assign to them their positive and negative flow as

$$F^+(X_i) = \sum_{j=1}^{n} P(X_i, X_j)/(n-1),$$

$$F^-(X_i) = \sum_{j=1}^{n} P(X_j, X_i)/(n-1),$$

MCDM - methods of class PROMETHEE

For given weights we can compute the global preference index of X_i, X_j by the formula

$P(X_i, X_j) = \sum_{h=1}^{k} v_h P_h(X_i, X_j).$

To obtain the final order of alternatives we have to assign to them their positive and negative flow as

$F^+(X_i) = \sum_{j=1}^{n} P(X_i, X_j)/(n-1),$

$F^-(X_i) = \sum_{j=1}^{n} P(X_j, X_i)/(n-1),$

There are different ways how to conclude, for example the method PROMETHEE II ranks alternatives with respect to the net flow $F(X_i) = F^+(X_i) - F^-(X_i)$.

MCDM - method PROMETHEE II, example

In a tablet problem, we derive differences d_h from standardized values y'_{ij}:

y'_{ij}	price	RAM	battery life	size	OS, processor, display
Tablet 1	0,53	0,2	0,93	0,62	1
Tablet 2	0,53	0,2	1	0,72	0,5
Tablet 3	1	0,07	0,57	1	0,25
Tablet 4	0	1	0	0	0,75
Tablet 5	1	0	0,14	0,97	0

MCDM - method PROMETHEE II, example

In a tablet problem, we derive differences d_h from standardized values y'_{ij}:

y'_{ij}	price	RAM	battery life	size	OS, processor, display
Tablet 1	0,53	0,2	0,93	0,62	1
Tablet 2	0,53	0,2	1	0,72	0,5
Tablet 3	1	0,07	0,57	1	0,25
Tablet 4	0	1	0	0	0,75
Tablet 5	1	0	0,14	0,97	0

After computation of differences $d_h(X_i, X_j)$ for all $i, j = 1, \dots, n$ and all $h = 1, \dots, k$ we use general criterion Q_1 with the threshold $d^* = 0,5$ to transform them to preference intensities $P_h(X_i, X_j)$.
Computations for the first criterion are shown in the following tables; first, it is necessary to determine d_1, then transform it to P_1.

MCDM - method PROMETHEE II, example

$d_1(X_i, X_j)$	T1	T2	T3	T4	T5
T1	0	0	-0,47	0,53	-0,47
T2	0	0	-0,47	0,53	-0,47
T3	0,47	0,47	0	1	0
T4	-0,53	-0,53	-1	0	1
T5	0,47	0,47	0	-1	0

MCDM - method PROMETHEE II, example

$d_1(X_i, X_j)$	T1	T2	T3	T4	T5
T1	0	0	-0,47	0,53	-0,47
T2	0	0	-0,47	0,53	-0,47
T3	0,47	0,47	0	1	0
T4	-0,53	-0,53	-1	0	1
T5	0,47	0,47	0	-1	0

Table of preference intensity:

$P_1(X_i, X_j)$	T1	T2	T3	T4	T5
T1	0	0	0	1	0
T2	0	0	0	1	0
T3	0	0	0	1	0
T4	0	0	0	0	1
T5	0	0	0	0	0

MCDM - method PROMETHEE II, example

The procedure is repeated for other criteria and finally the Saaty's weights $(0,41;\ 0,12;\ 0,22;\ 0,03;\ 0,22)$ are used to aggregate preference intensities to get the global preference index:

$P(X_i, X_j)$	T1	T2	T3	T4	T5	$F^+(X_i)$
T1	0	0	0,22	0,66	0,22	
T2	0	0	0	0,44	0,22	
T3	0	0	0	0,66	0,22	
T4	0,12	0,12	0,12	0	0,75	
T5	0	0	0	0	0	
$F^-(X_i)$						

Row averages show positive flows and column averages negative flows.

MCDM - method PROMETHEE II, example

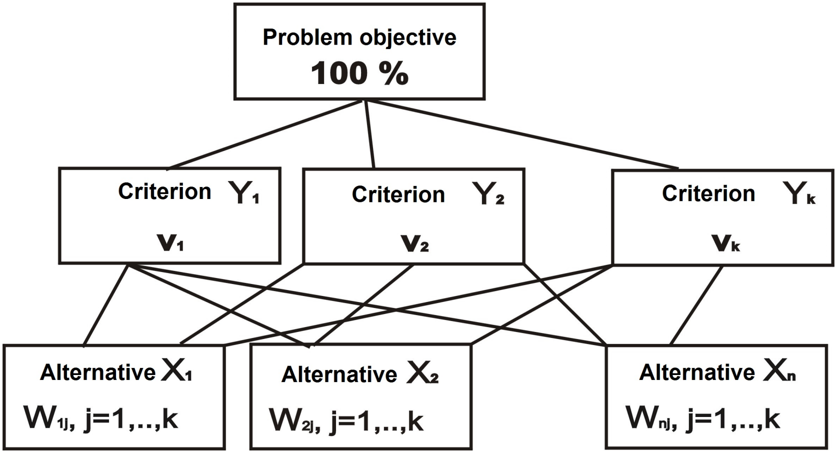
The procedure is repeated for other criteria and finally the Saaty's weights $(0,41;\ 0,12;\ 0,22;\ 0,03;\ 0,22)$ are used to aggregate preference intensities to get the global preference index:

$P(X_i, X_j)$	T1	T2	T3	T4	T5	$F^+(X_i)$
T1	0	0	0,22	0,66	0,22	0,275
T2	0	0	0	0,44	0,22	0,165
T3	0	0	0	0,66	0,22	0,22
T4	0,12	0,12	0,12	0	0,75	0,2775
T5	0	0	0	0	0	0
$F^-(X_i)$	0,03	0,03	0,085	0,44	0,3525	

The difference of positive and negative flow is the net flow:
$F(T1) = 0,245,\ F(T2) = 0,135,\ F(T3) = 0,135,$
$F(T4) = -0,1625,\ F(T5) = -0,3525$
The highest net flow is obtained for the tablet 1.

MCDM - method AHP

The method AHP (Analytic Hierarchy Process) models decision problem by means of hierarchical structure. The simplest problems are described using three-level hierarchy (see figure).



MCDM - method AHP

The importance of the individual elements of the hierarchy can be expressed by dividing the initial unit (100%) according to the preference of the decision-maker to the next level. First, on the second level, criteria are assigned weights $v_j,\ j = 1, \ldots, k$. Each weight v_j is further subdivided into numbers $w_{ij},\ i = 1, \ldots, n$ expressing how individual alternatives fulfil a given criterion. So we have $\sum_{j=1}^{k} v_j = 1,\ \sum_{i=1}^{n} w_{ij} = v_j,\ j = 1, \ldots k$ Finally total ranking of alternatives is derived from their benefit given as $u(X_i) = \sum_{j=1}^{k} w_{ij}$.

MCDM - method AHP

The importance of the individual elements of the hierarchy can be expressed by dividing the initial unit (100%) according to the preference of the decision-maker to the next level. First, on the second level, criteria are assigned weights $v_j,\ j = 1, \ldots, k$. Each weight v_j is further subdivided into numbers $w_{ij},\ i = 1, \ldots, n$ expressing how individual alternatives fulfil a given criterion. So we have $\sum_{j=1}^{k} v_j = 1,\ \sum_{i=1}^{n} w_{ij} = v_j,\ j = 1, \ldots k$ Finally total ranking of alternatives is derived from their benefit given as $u(X_i) = \sum_{j=1}^{k} w_{ij}$.
The numerical realization is based on pairwise comparison of elements as in the Saaty s method: For the highest node, a comparison matrix $(k \times k)$ is constructed and the criterion weights are derived from it. Subsequently, for each criterion, preference of alternatives is determined by pairwise comparisons in the matrix $(n \times n)$. The disadvantage of the method is obviously a large number of comparisons. On the other hand the pros of the method are its versatility and the possibility of using a verbal scale to express preferences.

Multi-criteria programming

The goal of multi-criteria programming is the optimization of more objective functions on the feasible region defined by the system of constraints. Unlike in MCDM problems, the set of decision alternatives is infinite and the criteria are defined in terms of functions. If all objective functions and limiting conditions are linear, we talk about multi-criteria linear programming (MLP). Therefore, the MLP problem can be formulated as a problem to "optimize"

$$z_1 = \mathbf{c^1} \cdot \mathbf{x},\ z_2 = \mathbf{c^2} \cdot \mathbf{x}, \ldots z_k = \mathbf{c^k} \cdot \mathbf{x},$$

subject to

$$\mathbf{x} \in \mathbf{X} = \{\mathbf{x} \in \mathbb{R}^{\mathbf{n}} | \mathbf{Ax} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\},$$

where $\mathbf{c^i}$ is the price vector of the i-th objective function.

Multi-criteria programming

The goal of multi-criteria programming is the optimization of more objective functions on the feasible region defined by the system of constraints. Unlike in MCDM problems, the set of decision alternatives is infinite and the criteria are defined in terms of functions. If all objective functions and limiting conditions are linear, we talk about multi-criteria linear programming (MLP). Therefore, the MLP problem can be formulated as a problem to "optimize"

$$z_1 = \mathbf{c^1} \cdot \mathbf{x},\ z_2 = \mathbf{c^2} \cdot \mathbf{x}, \dots z_k = \mathbf{c^k} \cdot \mathbf{x},$$

subject to

$$\mathbf{x} \in \mathbf{X} = \{\mathbf{x} \in \mathbb{R}^{\mathbf{n}} | \mathbf{Ax} \le \mathbf{b},\ \mathbf{x} \ge \mathbf{0}\},$$

where $\mathbf{c^i}$ is the price vector of the *i*-th objective function.
Using the equivalency of a minimization problem with an objective z_i to a maximization problem with $-z_i$, we can convert a multi-criteria problem into a form with all criteria having maximizing character. The problem can then be written using a matrix notation, if we denote $\mathbf{z} = (z_1,\ z_2, \dots, z_k)$ vector of objective functions and $\mathbf{C}$ matrix with rows given by price vectors $\mathbf{c^1},\ \mathbf{c^2}, \dots \mathbf{c^k}$:

$$\mathbf{z} = \mathbf{C} \cdot \mathbf{x} \to MAX,\ \mathbf{x} \in \mathbf{X}.$$

MLP model - basic definitions

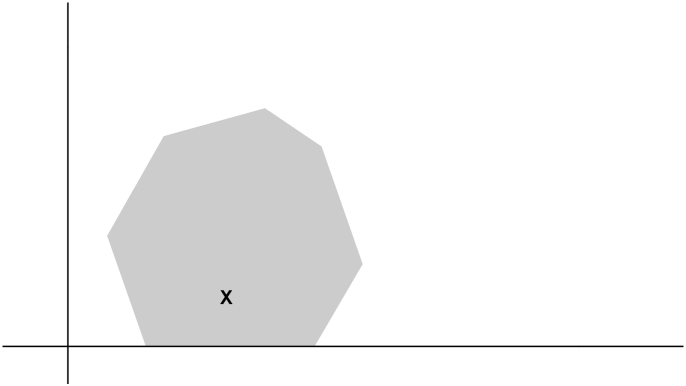
Typically, the goal of MLP is to find an acceptable compromise solution in a set of all feasible solutions. It is good to realize that when looking for a compromise solution, we can restrict the search to non-dominated solutions only. The solution $\mathbf{x} \in \mathbf{X}$ is non-dominated if there is no feasible solution with value vector „greater“ than the vector $\mathbf{C} \cdot \mathbf{x}$ („Vector $\mathbf{u}$ is greater than $\mathbf{v}$“ means that all their components satisfy $u_i \geq v_i$ and at least one component satisfies $u_i > v_i$).

MLP model - basic definitions

Typically, the goal of MLP is to find an acceptable compromise solution in a set of all feasible solutions. It is good to realize that when looking for a compromise solution, we can restrict the search to non-dominated solutions only. The solution $\mathbf{x} \in \mathbf{X}$ is non-dominated if there is no feasible solution with value vector „greater“ than the vector $\mathbf{C} \cdot \mathbf{x}$ („Vector $\mathbf{u}$ is greater than $\mathbf{v}$“ means that all their components satisfy $u_i \geq v_i$ and at least one component satisfies $u_i > v_i$).
Most of the principles for finding a compromise solution are based on solving partial linear programming problems $z_i = \mathbf{c^i} \cdot \mathbf{x} \rightarrow max, \ \mathbf{x} \in \mathbf{X}$ by a standard simplex method. The vector $\mathbf{x_H}$, for which all objective functions gain their optimal values, is called the ideal solution. By analogy, we can introduce the basal solution $\mathbf{x_D}$. Optimal and basal solutions usually don't lie in the feasible set.

MLP model - graphical representation

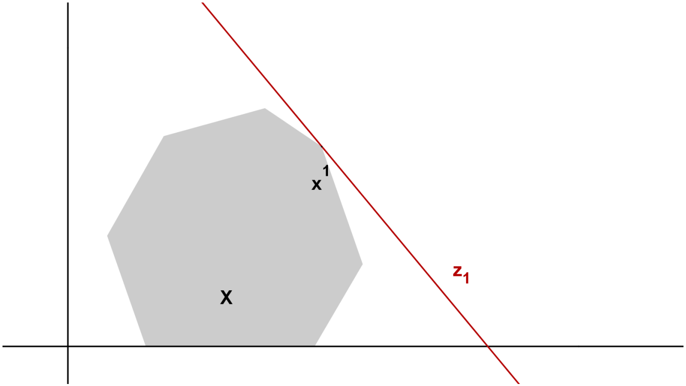
The multi-criteria linear model with two variables can be displayed in the decision space or in the criteria space. First, we show a problem in the decision space where the coordinate axes represent the values of the variables.



The feasible set X.

MLP model - graphical representation

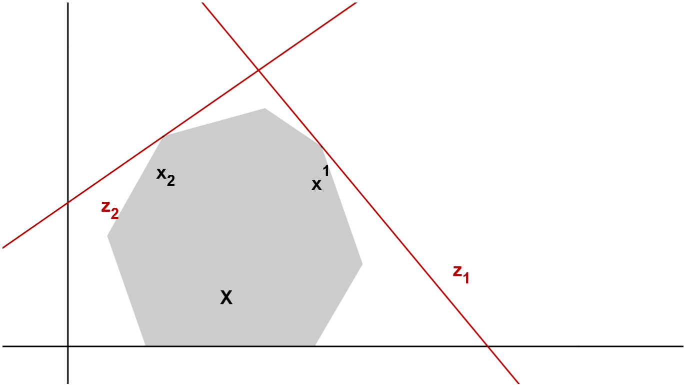
The multi-criteria linear model with two variables can be displayed in the decision space or in the criteria space. First, we show a problem in the decision space where the coordinate axes represent the values of the variables.



Partial LP problem for the objective $\mathbf{z_1} = \mathbf{c^1} \cdot \mathbf{x}$ with its optimum point $\mathbf{x^1}$.

MLP model - graphical representation

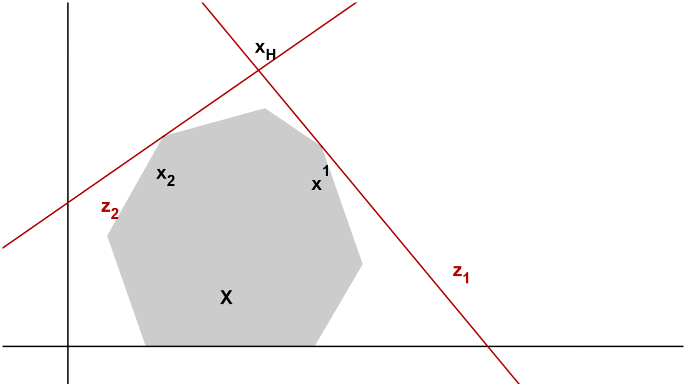
The multi-criteria linear model with two variables can be displayed in the decision space or in the criteria space. First, we show a problem in the decision space where the coordinate axes represent the values of the variables.



Partial LP problem for the objective $\mathbf{z_2} = \mathbf{c^2} \cdot \mathbf{x}$ with its optimum point $\mathbf{x^2}$.

MLP model - graphical representation

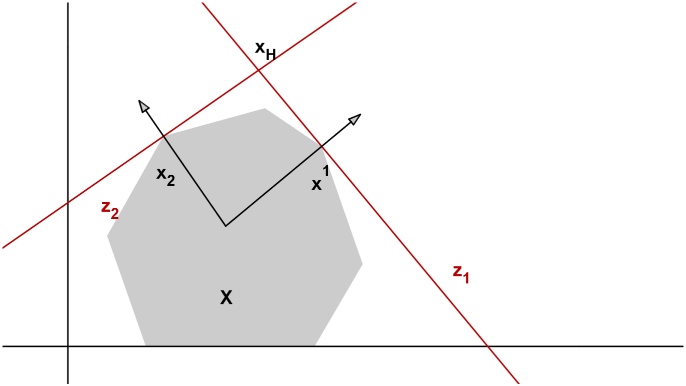
The multi-criteria linear model with two variables can be displayed in the decision space or in the criteria space. First, we show a problem in the decision space where the coordinate axes represent the values of the variables.



Ideal solution x_H lies in the intersection of level curves of individual objective functions going through partial optima points.

MLP model - graphical representation

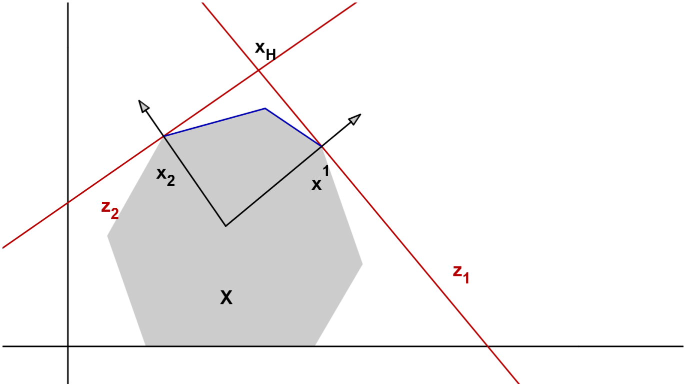
The multi-criteria linear model with two variables can be displayed in the decision space or in the criteria space. First, we show a problem in the decision space where the coordinate axes represent the values of the variables.



Normal vectors to these lines (i.e. the price coefficient vectors c^i) determine the direction of growth of the objective functions. Their non-negative linear combinations define the so-called criteria hull.

MLP model - graphical representation

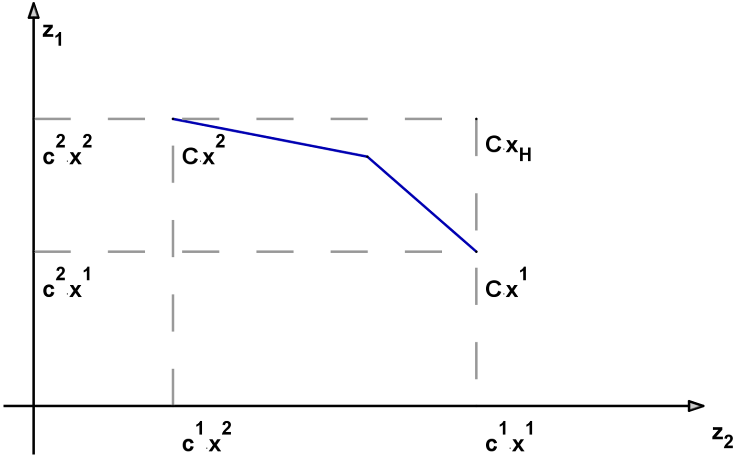
The multi-criteria linear model with two variables can be displayed in the decision space or in the criteria space. First, we show a problem in the decision space where the coordinate axes represent the values of the variables.



All non-dominated solutions lie in the intersection of the criteria hull and feasible set boundary.

MLP model - graphical representation

When displaying the solution in the criteria space, the individual axes correspond to the values partial objective functions.



Non-dominated values are marked by a blue color.

Multi-criteria programming - classification of methods

When solving MLP problems, we may require results in the form of a complete description of a set of non-dominated solutions or its representative subset or the selection of several compromise solutions. If the user wants to choose the only compromise solution, his decision will depend heavily on his preferences of the individual criteria. These can be entered in different phases of calculation:

- before the computation,
- interactively in the course of the computation,
- after the computation.

The preferences can be expressed by means of aspiration levels, criteria order or the substitution rates of criteria values.

Multi-criteria programming - classification of methods

When solving MLP problems, we may require results in the form of a complete description of a set of non-dominated solutions or its representative subset or the selection of several compromise solutions. If the user wants to choose the only compromise solution, his decision will depend heavily on his preferences of the individual criteria. These can be entered in different phases of calculation:

- before the computation,
- interactively in the course of the computation,
- after the computation.

The preferences can be expressed by means of aspiration levels, criteria order or the substitution rates of criteria values. We distinguish between:

- methods with apriori preference information
- methods with aposteriori preference information
- methods with interactive adjusting preference information
- combined methods

Multi-criteria programming - classification of methods

Methods with apriori preference information can be divided into several groups:

- lexicographic method
- method of minimal component
- aggregation of criteria
- switching the role of criteria and constraints
- "minimization of deviation"from ideal values (for a suitable metric)

Multi-criteria programming - classification of methods

Methods with apriori preference information can be divided into several groups:

- lexicographic method
- method of minimal component
- aggregation of criteria
- switching the role of criteria and constraints
- "minimization of deviation"from ideal values (for a suitable metric)

Methods with information aposteriori is based on the description of the set of non-dominated solutions in which the user chooses a compromise solution. This class includes:

- the parametric method (aggregation of criteria for a parametric vector of weights)
- the constraint method (searches for non-dominated solutions where criterion values reach parametric target values)
- multi-criteria simplex algorithm (determines non-dominated basic solutions gradually)

Multi-criteria programming - classification of methods

Methods with apriori preference information can be divided into several groups:

- lexicographic method
- method of minimal component
- aggregation of criteria
- switching the role of criteria and constraints
- "minimization of deviation"from ideal values (for a suitable metric)

Methods with information aposteriori is based on the description of the set of non-dominated solutions in which the user chooses a compromise solution. This class includes:

- the parametric method (aggregation of criteria for a parametric vector of weights)
- the constraint method (searches for non-dominated solutions where criterion values reach parametric target values)
- multi-criteria simplex algorithm (determines non-dominated basic solutions gradually)

Interactive methods are based on throughout communication between the analyst and the decision-maker.

Multi-criteria programming - example

Ski and snowboard rental management is considering to broaden its product range by four types of sets. The calculation includes the daily profits from lending the individual sets [in CZK] and the risk of loss of non-lending [in points].

	ski set adult	ski set child	cross-country skiing set	snowboard set
profit	300	200	170	250
risk	10	15	25	5

The company allocated 1 million CZK for the purchase of sets, where at least 200 thousand CZK should be used for snowboard sets. Suggest managers of the company the portfolio that maximizes the profit and minimizes the loss.

Multi-criteria LP example - building the model

Let's denote the amounts of respective sets purchased by $x_1, \ldots, x_4$. The objectives are

$z_1 = 300x_1 + 200x_2 + 170x_3 + 250x_4 \rightarrow max$

$z_2 = 10x_1 + 15x_2 + 25x_3 + 5x_4 \rightarrow min$

The budget and requirement on snowboard sets give following constraints:

$x_1 + x_2 + x_3 + x_4 \leq 100$

$x_4 \geq 20$

We include also non-negativity constraints :

$x_1, x_2, x_3, x_4 \geq 0$

We should also treat the problem as an integer problem, but for the sake of simplicity let us omit this constraint.

First, we solve the simplified model considering only one criteria function, we get the partial optimal solution.

Multi-criteria LP example - partial optimal solution

We can see that the company can achieve minimal risk if they purchase only required snowboard sets, i.e. 20 pcs costing 10,000 CZK each.

Multi-criteria LP example - partial optimal solution

We can see that the company can achieve minimal risk if they purchase only required snowboard sets, i.e. 20 pcs costing 10,000 CZK each.

Similarly, maximal return is gained if in addition to obligatory snowboard sets (20 pcs), the company spends the rest of the budget on the most profitable adult ski sets. The money would suffice for 80 pcs of sets.

Multi-criteria LP example - partial optimal solution

We can see that the company can achieve minimal risk if they purchase only required snowboard sets, i.e. 20 pcs costing 10,000 CZK each.

Similarly, maximal return is gained if in addition to obligatory snowboard sets (20 pcs), the company spends the rest of the budget on the most profitable adult ski sets. The money would suffice for 80 pcs of sets.

Partial optimal solutions can be arranged in the criteria table:

	Criteria function	
	return	risk
80 ski adult + 20 snowboard sets	29000	880
20 snowboard sets	5000	100

The values on the diagonal form the (nonexistent) ideal solution with return of 29,000 CZK and 100 risk points.

Multi-criteria LP - lexicographic method

Let's describe some methods with apriori preference information.
When using the lexicographic method, the decision-maker determines the order of importance of the criteria functions (assuming that the functions are already marked so that z_1 is the most significant, and z_k least significant, their optimal values are denoted by $z_1^{opt}, \ldots, z_k^{opt}$). Finding a compromise solution involves solving a sequence of optimization problems
$max\ z_1 = \mathbf{c^1} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X}.$ If it has more than one optimal solution, we solve another problem:
$max\ z_2 = \mathbf{c^2} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X},\ \mathbf{c^1 x} \geq z_1^{opt}.$ Again, if the solution is non-unique, we proceed further until we find the compromise solution as the optimal point of the problem
$max\ z_k = \mathbf{c^k} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X},\ \mathbf{c^1 x} \geq z_1^{opt}, \ldots,\ \mathbf{c^{k-1} x} \geq z_{k-1}^{opt}.$

Multi-criteria LP - lexicographic method

Let's describe some methods with apriori preference information.
When using the lexicographic method, the decision-maker determines the order of importance of the criteria functions (assuming that the functions are already marked so that z_1 is the most significant, and z_k least significant, their optimal values are denoted by $z_1^{opt}, \dots, z_k^{opt}$). Finding a compromise solution involves solving a sequence of optimization problems
$max\ z_1 = \mathbf{c^1} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X}.$ If it has more than one optimal solution, we solve another problem:
$max\ z_2 = \mathbf{c^2} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X},\ \mathbf{c^1 x} \geq z_1^{opt}.$ Again, if the solution is non-unique, we proceed further until we find the compromise solution as the optimal point of the problem
$max\ z_k = \mathbf{c^k} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X},\ \mathbf{c^1 x} \geq z_1^{opt}, \dots,\ \mathbf{c^{k-1} x} \geq z_{k-1}^{opt}.$
To mitigate the absolute preference of more important criteria, it is possible to allow a deviation from optimal value $\delta_i, i = 1, \dots, k$ in each step. For example, in the second step, we would solve the problem
$max\ z_2 = \mathbf{c^2} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X},\ \mathbf{c^1 x} \geq z_1^{opt} - \delta_1,$ etc.
The described method is similar to the way of thinking of managers.

Lexicographic method, example

We already know that when we prefer the risk, we should buy only 20 snowboard sets. Under the preference of profit, 20 snowboard and 80 adult sets should be purchased.
What is the solution in the second step if we prefer risk, but accept its deviation from the ideal value of 100 to 120? We maximize the profit
$z_1 = 300x_1 + 200x_2 + 170x_3 + 250x_4$ subject to
$z_2 = 10x_1 + 15x_2 + 25x_3 + 5x_4 \leq 120$, (risk constraint) and other constraints of the model
$x_1 + x_2 + x_3 + x_4 \leq 100 \; x_4 \geq 20$, $x_1, \ldots, x_4 \geq 0.$

Lexicographic method, example

We already know that when we prefer the risk, we should buy only 20 snowboard sets. Under the preference of profit, 20 snowboard and 80 adult sets should be purchased.
What is the solution in the second step if we prefer risk, but accept its deviation from the ideal value of 100 to 120? We maximize the profit
$z_1 = 300x_1 + 200x_2 + 170x_3 + 250x_4$ subject to
$z_2 = 10x_1 + 15x_2 + 25x_3 + 5x_4 \leq 120$, (risk constraint) and other constraints of the model
$x_1 + x_2 + x_3 + x_4 \leq 100$ $x_4 \geq 20$, $x_1, \ldots, x_4 \geq 0$.
We can easily arrive at optimal solution $x_1 = 2, x_2 = 0, x_3 = 0, x_4 = 20$, so we should buy 2 adult ski sets in addition to 20 snowboard sets. The return would be 5600 CZK and the risk would reach its limit 120 risk points.

Multi-criteria LP - minimal component method

Another approach to determining the compromise solution is a minimal component method, where we maximize the worst (i.e. the smallest) component of the function value vector. So we get LP to maximize $z = \delta$ under conditions $\mathbf{c}^{\mathbf{1}} \cdot \mathbf{x} \geq \delta,\ \mathbf{c}^{\mathbf{2}} \cdot \mathbf{x} \geq \delta, \ldots \mathbf{c}^{\mathbf{k}} \cdot \mathbf{x} \geq \delta,\ \mathbf{x} \in \mathbf{X}$.

Multi-criteria LP - minimal component method

Another approach to determining the compromise solution is a minimal component method, where we maximize the worst (i.e. the smallest) component of the function value vector. So we get LP to maximize $z = \delta$ under conditions $\mathbf{c^1} \cdot \mathbf{x} \geq \delta,\ \mathbf{c^2} \cdot \mathbf{x} \geq \delta, \ldots \mathbf{c^k} \cdot \mathbf{x} \geq \delta,\ \mathbf{x} \in \mathbf{X}$.

If some of the criteria are of a minimization nature, they must first be converted to maximization and all have to be adjusted to dimensionless variables to ensure their comparability.

Multi-criteria LP - aggregation of criteria functions

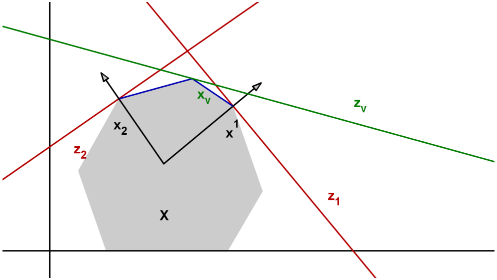
By using a suitable operator, it is possible to merge all the criteria functions into a single one. Generally, different operators are used for aggregation, for example linear combination of individual functions using a standardized weight vector $\mathbf{v} = (\mathbf{v_1}, \ldots, \mathbf{v_k})$. We replace the initial problem $\mathbf{z} = \mathbf{C} \cdot \mathbf{x} \to MAX,\ \mathbf{x} \in \mathbf{X}$ by an one-dimensional problem $z_v = \mathbf{v} \cdot \mathbf{C} \cdot \mathbf{x} \to max,\ \mathbf{x} \in \mathbf{X}$.

Multi-criteria LP - aggregation of criteria functions

By using a suitable operator, it is possible to merge all the criteria functions into a single one. Generally, different operators are used for aggregation, for example linear combination of individual functions using a standardized weight vector $\mathbf{v} = (\mathbf{v_1}, \ldots, \mathbf{v_k})$. We replace the initial problem $\mathbf{z} = \mathbf{C} \cdot \mathbf{x} \rightarrow MAX, \ \mathbf{x} \in \mathbf{X}$ by an one-dimensional problem

$z_v = \mathbf{v} \cdot \mathbf{C} \cdot \mathbf{x} \rightarrow max, \ \mathbf{x} \in \mathbf{X}.$

We can depict the aggregate criterion graphically (green line). It has no practical interpretation, it is only an auxiliary criterion. Attention! When setting the weights it is necessary to take into account different scales of the respective functions.



Aggregation of criteria functions, example

For aggregation of the profit z_1 and the risk z_2 we need to take into account different types of criteria, e.g. to multiply z_2 by the coefficient (-1).
If we set the weights equal to

$\mathbf{v} = \left(\frac{5}{100}, \frac{95}{100}\right)$,

the aggregated criteria function would be given by the formula

$z_v = 5,5x_1 - 4,25x_2 - 15,25x_3 + 7,75x_4.$

Aggregation of criteria functions, example

For aggregation of the profit z_1 and the risk z_2 we need to take into account different types of criteria, e.g. to multiply z_2 by the coefficient (-1).
If we set the weights equal to

$$\mathbf{v} = \left(\frac{5}{100}, \frac{95}{100}\right),$$

the aggregated criteria function would be given by the formula

$$z_v = 5,5x_1 - 4,25x_2 - 15,25x_3 + 7,75x_4.$$

We can easily see that the optimal solution would be
$x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 100$, spending all the money on snowboard sets.
The optimal value of $z_v = 775$ has no meaning, but can be used to calculate the profit 25000 CZK and the risk of 500 points for the solution found.

Multi-criteria LP - conversion of criteria to constraints

The procedure is similar to the lexicographic method when we apply the sequential conversion of criteria. Again, we assume that the functions are ordered from the most to the least important. Finding a compromise solution involves solving a sequence of optimization tasks

$max.\ z_1 = \mathbf{c^1} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X}.$ Optimal value is denoted by z_1^*. In the next step we solve another problem for z_2 allowing for a certain deviation of the first criteria from z_1^*:

$max.\ z_2 = \mathbf{c^2} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X},\ \mathbf{c^1 x} \geq z_1^* - \delta_1.$ Again we find z_2^* and proceed further until we find a compromise solution as the solution of

$max.\ z_k = \mathbf{c^k} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X},\ \mathbf{c^1 x} \geq z_1^* - \delta_1, \ldots,\ \mathbf{c^{k-1} x} \geq z_{k-1}^* - \delta_k.$

Multi-criteria LP - conversion of criteria to constraints

The procedure is similar to the lexicographic method when we apply the sequential conversion of criteria. Again, we assume that the functions are ordered from the most to the least important. Finding a compromise solution involves solving a sequence of optimization tasks

$max.\ z_1 = \mathbf{c^1} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X}.$ Optimal value is denoted by z_1^*. In the next step we solve another problem for z_2 allowing for a certain deviation of the first criteria from z_1^*:

$max.\ z_2 = \mathbf{c^2} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X},\ \mathbf{c^1 x} \geq z_1^* - \delta_1.$ Again we find z_2^* and proceed further until we find a compromise solution as the solution of

$max.\ z_k = \mathbf{c^k} \cdot \mathbf{x},\ \mathbf{x} \in \mathbf{X},\ \mathbf{c^1 x} \geq z_1^* - \delta_1, \ldots,\ \mathbf{c^{k-1} x} \geq z_{k-1}^* - \delta_k.$

We can also convert the criteria to the constraints simultaneously by maximizing only the most important criterion z_1 and adding to the model all the conditions $z_i \geq au_i,\ i = 2, \ldots k$ (aspiration levels of criteria $au_i, i = 2, \ldots k$ set somewhere between basal and ideal value for a given criterion $\langle z_i^{min}, z_i^{max} \rangle$.) The disadvantage of this approach is that it can be difficult to set aspiration levels correctly, so that the set is not empty and neither is the significance of the criteria completely eliminated.

Goal programming

It represents a different approach to solving LP problems. Instead of distinguishing constraints and the criterion of optimality, fixed and free goals assigned with target values are used. The value of fixed targets have to be reached exactly (analogy of limiting conditions). For free targets, we can achieve both higher and lower values than the target (but it must not differ too much). Since there are in general many target values and not all of them can be reached, one of two approaches is usually chosen:

- using preference information - the goal of top priority is optimized first, etc.
- using weights - coefficients expressing the importance of respective goals; then the weighted sum of deviations from all targets is optimized

Goal programming models are more general than standard LP models and they often better represent real situations in applications.

Goal programming - example

Example from J. Jablonský, „Operational Research“: Management of the pension fund decides to purchase two types of assets (shares and bonds). There is a budget that can not be exceeded. Up to 50% of the total budget can be invested in the shares and the maximum of 75 % of money in the funds. Expected revenues are is 15 % from shares and alert 10 % from bonds, the risk of the investment is rated at 5 pts for shares and 2 pts for bonds. Design a portfolio $[x_1, x_2]$ to achieve the average return of 12 % p.a. and risk equal to 3 points of risk .

Goal programming - example

Example from J. Jablonský, „Operational Research“: Management of the pension fund decides to purchase two types of assets (shares and bonds). There is a budget that can not be exceeded. Up to 50% of the total budget can be invested in the shares and the maximum of 75 % of money in the funds. Expected revenues are is 15 % from shares and alert 10 % from bonds, the risk of the investment is rated at 5 pts for shares and 2 pts for bonds. Design a portfolio $[x_1, x_2]$ to achieve the average return of 12 % p.a. and risk equal to 3 points of risk .

In the standard LP approach, we would have to choose one of the criteria as the objective function (let’s say revenue) and maximize it under conditions, that the risk cannot exceed 3 points. It can be verified that the optimal solution can be reached in this approach if $\frac{1}{3}$ of funds would be invested into shares and $\frac{2}{3}$ into bonds. At the average risk equal to 3 points, we get return 11,67 % p.a.

Goal programming - example

Example from J. Jablonský, „Operational Research“: Management of the pension fund decides to purchase two types of assets (shares and bonds). There is a budget that can not be exceeded. Up to 50% of the total budget can be invested in the shares and the maximum of 75 % of money in the funds. Expected revenues are is 15 % from shares and alert 10 % from bonds, the risk of the investment is rated at 5 pts for shares and 2 pts for bonds. Design a portfolio $[x_1, x_2]$ to achieve the average return of 12 % p.a. and risk equal to 3 points of risk.

In the standard LP approach, we would have to choose one of the criteria as the objective function (let's say revenue) and maximize it under conditions, that the risk cannot exceed 3 points. It can be verified that the optimal solution can be reached in this approach if $\frac{1}{3}$ of funds would be invested into shares and $\frac{2}{3}$ into bonds. At the average risk equal to 3 points, we get return 11,67 % p.a.
Similarly, we can minimize the objective function expressed by the weighted risk with an additional condition that the average yield is at least 12 % p.a. We can get the solution that it is optimal to invest 40 % into shares and 60 % into bonds. At the return 12% p.a. the risk rate will be 3.2 points.

Goal programming - model formulation

We use deviation variables for expressing positive or negative slacks in free goals (we denote them by d_i^+ or d_i^- respectively). If the target value is achieved, then $d_i^+ = d_i^- = 0$. If the target is exceeded, then $d_i^+ > 0,\ d_i^- = 0$, and if the goal is not achieved, then $d_i^+ = 0,\ d_i^- > 0$. Fixed targets must be respected, no deviations are allowed.

Goal programming - model formulation

We use deviation variables for expressing positive or negative slacks in free goals (we denote them by d_i^+ or d_i^- respectively). If the target value is achieved, then $d_i^+ = d_i^- = 0$. If the target is exceeded, then $d_i^+ > 0,\ d_i^- = 0$, and if the goal is not achieved, then $d_i^+ = 0,\ d_i^- > 0$. Fixed targets must be respected, no deviations are allowed.

In the goal programming model, the objective function is always expressed as minimizing deviation variables, including either positive, negative or both types of deviations (then we approach the target values from the top or bottom or from both sides). Therefore, a portfolio optimization problem formulation would be:

$d_2^+,\ d_1^- \rightarrow min,$

subject to:

$x_1 + x_2 \leq 1$

$x_1 \leq 0,5;\ x_2 \leq 0,75$

$15x_1 + 10x_2 + d_1^+ - d_1^- = 12$

$5x_1 + 2x_2 + d_2^+ - d_2^- = 3$

$x_1, x_2, d_1^+, d_1^-, d_2^+, d_2^- \geq 0$

Goal programming - weights method

While minimizing multiple deviations, we can express their importance by weights. In order to avoid problems with different units, it is better to work with relative deviations $\frac{d_i^+}{g_i}$, $\frac{d_i^-}{g_i}$, where g_i is i-th goal. If the return is five times more important than the risk, we will set the weights equal to 5 and 1 and the objective function will take the form $z = 5\frac{d_1^-}{12} + \frac{d_2^+}{3}$. The problem can be solved by a simplex method.

Goal programming - weights method

While minimizing multiple deviations, we can express their importance by weights. In order to avoid problems with different units, it is better to work with relative deviations $\frac{d_i^+}{g_i}$, $\frac{d_i^-}{g_i}$, where g_i is i-th goal. If the return is five times more important than the risk, we will set the weights equal to 5 and 1 and the objective function will take the form $z = 5\frac{d_1^-}{12} + \frac{d_2^+}{3}$. The problem can be solved by a simplex method.

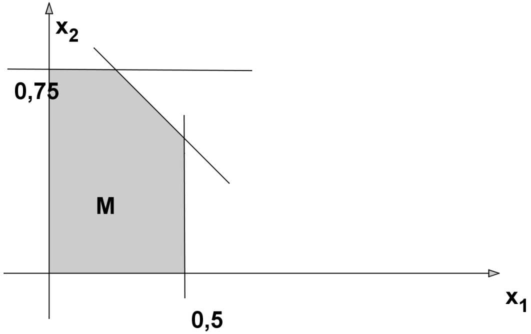
The solution is to invest $\frac{1}{3}$ of funds to shares and $\frac{2}{3}$ to bonds; the target value of the risk is achieved and the return is 11.67%.

Goal programming - preemptive method

In the preemptive method, we start by minimizing the deviation from a more important goal. If we get more solutions, we minimize the second most important deviation on this set, etc. If the priority of return is higher in our portfolio problem, we minimize d_1^- first.

Goal programming - preemptive method

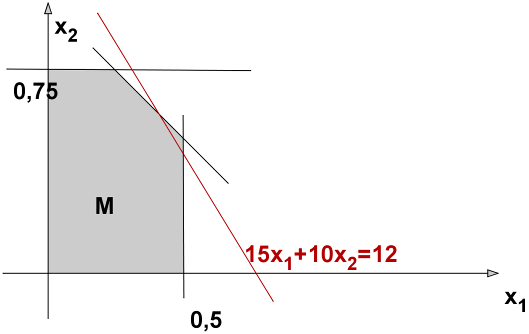
In the preemptive method, we start by minimizing the deviation from a more important goal. If we get more solutions, we minimize the second most important deviation on this set, etc. If the priority of return is higher in our portfolio problem, we minimize d_1^- first.
The situation can be visualized in the plane x_1, x_2.



First, we show a feasible set M where all fixed targets are met.

Goal programming - preemptive method

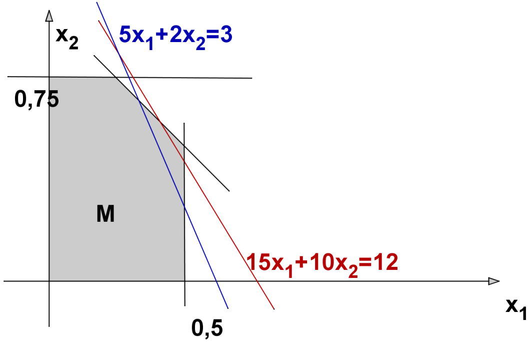
In the preemptive method, we start by minimizing the deviation from a more important goal. If we get more solutions, we minimize the second most important deviation on this set, etc. If the priority of return is higher in our portfolio problem, we minimize d_1^- first.
The situation can be visualized in the plane x_1, x_2.



The line $15x_1 + 10x_2 = 12$ corresponds to the lowest possible value of $d_1^- = 0$. Its intersection with M represents all the optimal solutions in the first step.

Goal programming - preemptive method

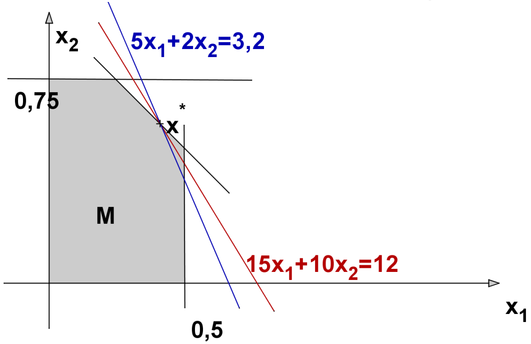
In the preemptive method, we start by minimizing the deviation from a more important goal. If we get more solutions, we minimize the second most important deviation on this set, etc. If the priority of return is higher in our portfolio problem, we minimize d_1^- first.
The situation can be visualized in the plane x_1, x_2.



In the second step, we minimize the deviation d_2^+. Zero value of this deviation is achieved on the line $5x_1 + 2x_2 = 3$.

Goal programming - preemptive method

In the preemptive method, we start by minimizing the deviation from a more important goal. If we get more solutions, we minimize the second most important deviation on this set, etc. If the priority of return is higher in our portfolio problem, we minimize d_1^- first.
The situation can be visualized in the plane x_1, x_2.



Since the lines intersect outside the feasible set, the equality $d_2^+ = 0$ cannot be satisfied. We have to raise the risk, i.e. to move the blue line so that it intersects with the optimal line for d_1^- inside M. We've found the optimum point $x^* = [0,4; 0,6]$.

Linear fractional programming

A class of problems that can be converted to a linear program, is represented by problems of linear fractional programming, i. e. to optimize a function $f(\mathbf{x}) = \frac{\mathbf{d}^\top \cdot \mathbf{x} + d_0}{\mathbf{c}^\top \cdot \mathbf{x} + c_0}$ subject to $\mathbf{x} \geq \mathbf{0}$, $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$. Assuming positivity of the denominator $\mathbf{c}^\top \cdot \mathbf{x} + c_0 > 0$, we can use the substitution $r = \frac{1}{\mathbf{c}^\top \cdot \mathbf{x} + c_0}$. We get the objective function in the form $f = \mathbf{d}^\top \cdot \mathbf{x} \cdot r + d_0 r$, which can be linearized after introducing new variables $r, y_i = x_i \cdot r, i = 1, \ldots, n$: $f(r, y_1, \ldots y_n) = \mathbf{d}^\top \cdot \mathbf{y} + d_0 r$. Similar linearization can be used for the constraints $r \geq 0, \mathbf{y} \geq \mathbf{0}$, $\mathbf{A} \cdot \mathbf{y} \leq \mathbf{b} \cdot r$. We must not forget to add another constraint given by the definition of variable r, i.e. $1 = r(\mathbf{c}^\top \cdot \mathbf{x} + c_0)$ or in linear form: $1 = \mathbf{c}^\top \cdot \mathbf{y} + c_0 r$.

We have a common LP problem, which can be solved e.g. by the simplex method.

Linear fractional programming - example

In corporate economy, many indicators of a ratio type are used. If the expressions in the numerator and the denominator of the indicator are linear functions of the variables, optimizing the indicator is a problem of linear fractional programming, see the example from the book I. Gros: "Kvantitativní metody v manažerském rozhodování":

Linear fractional programming - example

In corporate economy, many indicators of a ratio type are used. If the expressions in the numerator and the denominator of the indicator are linear functions of the variables, optimizing the indicator is a problem of linear fractional programming, see the example from the book I. Gros: "Kvantitativní metody v manažerském rozhodování":

Example: Let us consider the company that has three products A, B and C in the production program with the following characteristics:

Product	Variable costs [CZK/t]	Price [CZK/t]
A	11 000	12 000
B	15 000	18 000
C	14 000	16 000
fixed costs 150 000 CZK		

If we denote the amount of the respective products sold by $x_1,\ x_2,\ x_3$, we can define for example following indicators:

cost of sales $z = \frac{11000x_1+15000x_2+14000x_3+150000}{12000x_1+18000x_2+16000x_3}$,

profitability of sales $z = \frac{1000x_1+3000x_2+2000x_3-150000}{12000x_1+18000x_2+16000x_3}$.

Linear fractional programming - example

Example: Minimize the cost of sales from this example, provided that the total cost does not exceed 200 000 CZK.

Linear fractional programming - example

Example: Minimize the cost of sales from this example, provided that the total cost does not exceed 200 000 CZK.

Mathematical formulation: minimize $z = \frac{11x_1+15x_2+14x_3+150}{12x_1+18x_2+16x_3}$
subject to $11x_1 + 15x_2 + 14x_3 + 150 \leq 200$, $x_1,\ x_2,\ x_3 \geq 0$.

Linear fractional programming - example

Example: Minimize the cost of sales from this example, provided that the total cost does not exceed 200 000 CZK.

Mathematical formulation: minimize $z = \frac{11x_1+15x_2+14x_3+150}{12x_1+18x_2+16x_3}$
subject to $11x_1 + 15x_2 + 14x_3 + 150 \leq 200$, $x_1,\ x_2,\ x_3 \geq 0$.

After the substitution $r = \frac{1}{12x_1+18x_2+16x_3}$ we get the problem of minimizing function
$f(y_1, y_2, y_3, r) = 11y_1 + 15y_2 + 14y_3 + 150r$
subject to $11y_1 + 15y_2 + 14y_3 - 50r \leq 0$, $y_1,\ y_2,\ y_3,\ r \geq 0$ and additional condition $12y_1 + 18y_2 + 16y_3 = 1$

Linear fractional programming - example

Example: Minimize the cost of sales from this example, provided that the total cost does not exceed 200 000 CZK.

Mathematical formulation: minimize $z = \frac{11x_1+15x_2+14x_3+150}{12x_1+18x_2+16x_3}$
subject to $11x_1 + 15x_2 + 14x_3 + 150 \leq 200$, $x_1,\ x_2,\ x_3 \geq 0$.

After the substitution $r = \frac{1}{12x_1+18x_2+16x_3}$ we get the problem of minimizing function
$f(y_1, y_2, y_3, r) = 11y_1 + 15y_2 + 14y_3 + 150r$
subject to $11y_1 + 15y_2 + 14y_3 - 50r \leq 0$, $y_1,\ y_2,\ y_3,\ r \geq 0$ and additional condition $12y_1 + 18y_2 + 16y_3 = 1$

Similar procedure can be used for the problem of maximizing the profitability of sales.

Introduction

Data envelopment analysis (DEA) is used to evaluate the technical efficiency of production units based on the size of inputs and outputs (without the need for pricing).

Introduction

Data envelopment analysis (DEA) is used to evaluate the technical efficiency of production units based on the size of inputs and outputs (without the need for pricing).

History:
1957: Farrell: model for one input and one output
1978: Charnes, Cooper, Rhodes: CCR model: multiple inputs and outputs, constant returns to scale
1984: Banker, Charnes, Cooper: BCC model: variable returns to scale

Introduction

Data envelopment analysis (DEA) is used to evaluate the technical efficiency of production units based on the size of inputs and outputs (without the need for pricing).

History:
1957: Farrell: model for one input and one output
1978: Charnes, Cooper, Rhodes: CCR model: multiple inputs and outputs, constant returns to scale
1984: Banker, Charnes, Cooper: BCC model: variable returns to scale

Let us consider homogeneous production units consuming the same type of resources (material, floor area, workers, etc.), so-called inputs, to produce equivalent effects (sales, profits, number of serviced clients), i.e. outputs. If there is only one input and one output, it is easy to express efficiency using the ratio indicator
efficiency = input/output

Introduction

Data envelopment analysis (DEA) is used to evaluate the technical efficiency of production units based on the size of inputs and outputs (without the need for pricing).

History:
1957: Farrell: model for one input and one output
1978: Charnes, Cooper, Rhodes: CCR model: multiple inputs and outputs, constant returns to scale
1984: Banker, Charnes, Cooper: BCC model: variable returns to scale

Let us consider homogeneous production units consuming the same type of resources (material, floor area, workers, etc.), so-called inputs, to produce equivalent effects (sales, profits, number of serviced clients), i.e. outputs. If there is only one input and one output, it is easy to express efficiency using the ratio indicator
efficiency = input/output
An aggregation indicator can be constructed for multiple inputs and outputs
efficiency = weighted inputs/weighted outputs

One output/ one input model

Let us consider 8 branches of a business firm, which are characterized by one input (number of employees) and one output (the number of contracts concluded with clients). Their efficiency can be expressed using the indicator "number of contracts per employee"; data are recorded in the table:

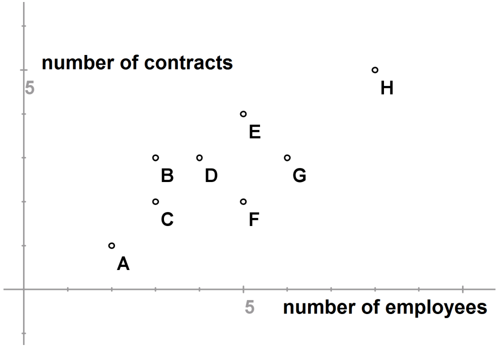
branch	A	B	C	D	E	F	G	H
employees	2	3	3	4	5	5	6	8
contracts	1	3	2	3	4	2	3	5
efficiency	0,5	1	0,667	0,75	0,8	0,4	0,5	0,625

One output/ one input model

We represent the data graphically, see Fig:

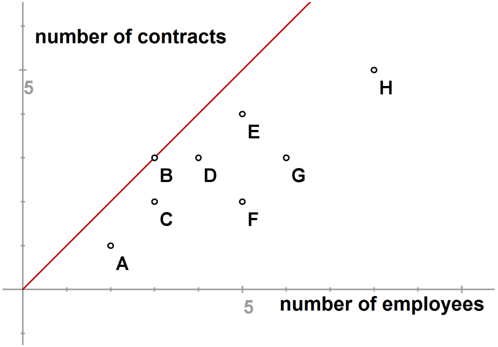
One output/ one input model

We represent the data graphically, see Fig:



One output/ one input model

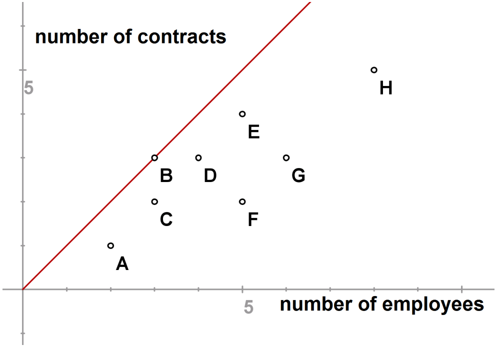
We represent the data graphically, see Fig:



The efficiency of a given branch is indicated by the slope of the line connecting the point to the origin. The highest value of the slope is obtained for branch B - this line will be called the efficient frontier.

One output/ one input model

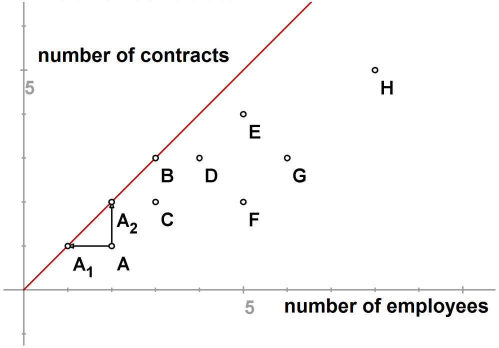
We represent the data graphically, see Fig:



Efficiency of a given branch is indicated by the slope of the line connecting the point to the origin. The line with the largest slope is obtained for branch B - this line will be called the efficient frontier. The branch B is the best and the efficiency of other units can be expressed relatively as the ratio of their slopes to the slope of B. For example Efficiency of A / Efficiency of B =0.5 meaning that unit A is only 50% efficient compared to B. This relative measure acquires values from [0, 1] for all branches and is independent of units used for input and output.

One output/ one input model

How can unit A achieve 100% value of efficiency score, i.e. reach effective frontier? It can reduce inputs while maintaining outputs (input-oriented model, graphically represented by the point A_1) or increase output while maintaining inputs (output-oriented model, graphically represented by the point A_2) or change both. Units A_1, A_2 are called virtual, they represent no real branch. Line segment A_1A_2 represents all points on the efficient frontier, that are reachable from A without increasing the number of employees or reducing the number of contracts.



One output/ one input model

Let's denote the coordinates of the points $A[x, y]$, $A_1[x_1, y_1]$, $A_2[x_2, y_2]$. The points A_1, A_2 are on the efficient frontier, so we can substitute coordinates of these virtual units to the denominator of the ratio expressing relative efficiency of A.
We have
$\frac{x}{y}/\frac{x_1}{y_1} = x/x_1$, because $y = y_1$
or
$\frac{x}{y}/\frac{x_2}{y_2} = y_2/y$, because $x = x_2$.

One output/ one input model

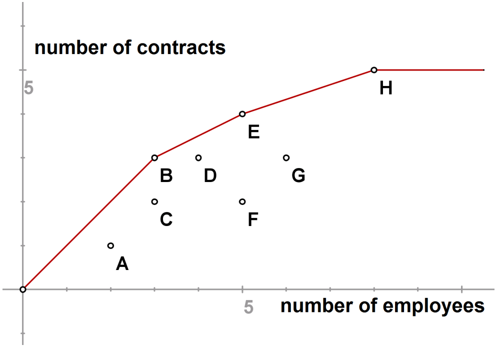
Let's denote the coordinates of the points $A[x, y]$, $A_1[x_1, y_1]$, $A_2[x_2, y_2]$. The points A_1, A_2 are on the efficient frontier, so we can substitute coordinates of these virtual units to the denominator of the ratio expressing relative efficiency of A.
We have
$\frac{x}{y} / \frac{x_1}{y_1} = x/x_1$, because $y = y_1$
or
$\frac{x}{y} / \frac{x_2}{y_2} = y_2/y$, because $x = x_2$.
In the output-oriented model, we can interpret relative efficiency as a necessary increase in output, $\frac{y_2}{y} = \frac{2}{1} = 2$, so the branch A can reach the efficient frontier by doubling the number of contracts.
For the input-oriented model, the reciprocal of relative efficiency represents necessary input reduction, $\frac{x_1}{x} = \frac{1}{2} = 0,5$, so the branch A would lay on the the efficient frontier with half of the employees.

One output/ one input model

Until now, we have assumed constant returns to scale, so the efficient frontier was formed by ray going from the origin to the most efficient unit. For every unit with the coordinates $[x, y]$, the input and output $[\alpha x, \alpha y]$ are feasible for every $\alpha > 0$ under CRS assumption. The efficiency score is independent of the input/output orientation.

One output/ one input model

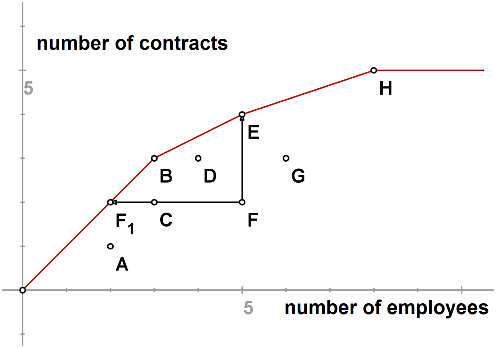
Until now, we have assumed constant returns to scale, so the efficient frontier was formed by ray going from the origin to the most efficient unit. For every unit with the coordinates $[x, y]$, the input and output $[\alpha x, \alpha y]$ are feasible for every $\alpha > 0$ under CRS assumption. The efficiency score is independent of the input/output orientation.
Let's redraw the efficient frontier assuming variable returns to scale.



Units E and H are fully efficient under VRS assumption.

One output/ one input model

The efficiency scores differ according to the orientation of the VRS model. For example unit F achieves the score efficiency F_1 / efficiency F = 2/5 = 0,4, under input orientation, whereas its score is efficiency F/efficiency E = 1/2= 0,5 under output orientation.



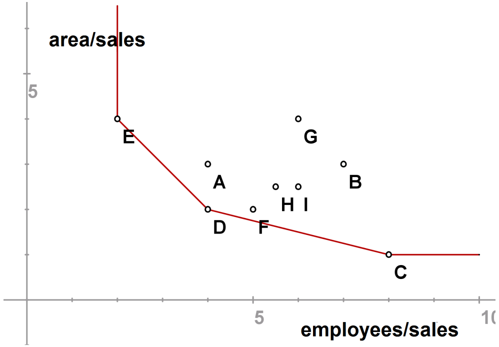
Two inputs and one output

Consider an example of 9 stores with inputs given by the number of employees (in tens) and the floor area (1000 m^2) and annual sales (in 1000000 CZK) on the output side. Let's assume constant returns to scale. For the sake of comparability, we can further consider the input values per 1 million of CZK of the sales. The normalized values are listed in the table.

Store	A	B	C	D	E	F	G	H	I
employees	4	7	8	4	2	5	6	5.5	6
area	3	3	1	2	4	2	4	2.5	2.5
sales	1	1	1	1	1	1	1	1	1

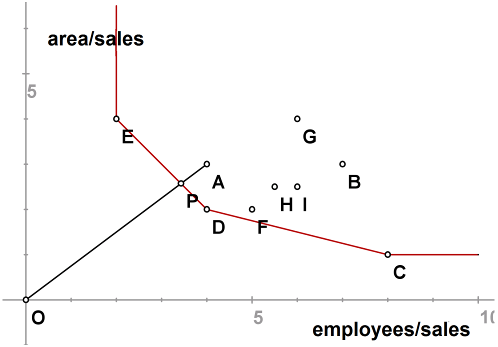
Two inputs and one output

In the graphical representation, those stores that are closer to the origin appear to be more efficient. The efficient frontier envelops the data as follows:



Two inputs and one output

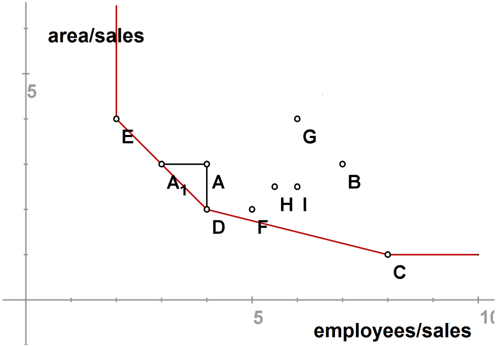
In the graphical representation, those stores that are closer to the origin appear to be more efficient. The efficient frontier envelops the data as follows:



Store A is inefficient, its efficiency can be measured radially as $\frac{|OP|}{|OA|} = 0.8571$. Virtual unit P is convex combination of D and E which are called peer units of A.

Two inputs and one output

In the graphical representation, those stores that are closer to the origin appear to be more efficient. The efficient frontier envelops the data as follows:



Store A is inefficient, its efficiency can be measured radially as $\frac{|OP|}{|OA|} = 0.8571$. Virtual unit P is a convex combination of D and E which are called peer units of A. Efficient frontier can be achieved by proportionally reducing both inputs by 15% or different way; decreasing one input while maintaining the level of the second one is demonstrated by the points A_1, D.

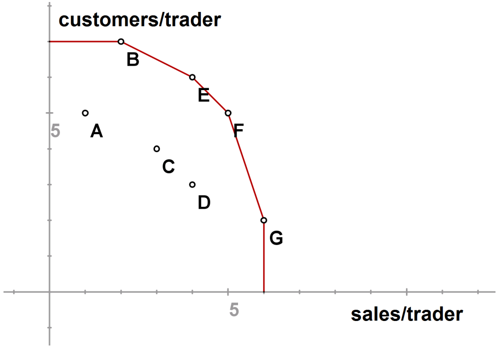
One input and two outputs

Now consider the case of 1 input (number of traders) and two outputs (customers and sales) at 7 sales offices. The values of the outputs per 1 trader for individual branches are listed in the table.

Office	A	B	C	D	E	F	G
traders	1	1	1	1	1	1	1
customers	1	2	3	4	4	5	6
sales	5	7	4	3	6	5	2

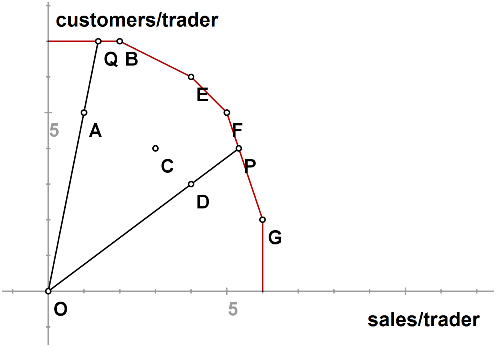
One input and two outputs

We can depict each office by its unitary outputs; the efficient frontier will envelop the data from the opposite side, because the points lying closer to the origin represent less efficient units (units A, C, D).



One input and two outputs

We can depict each office by its unitary outputs; the efficient frontier will envelop the data from the opposite side, because the points lying closer to the origin represent less efficient units (units A, C, D).



We can measure the efficiency radially, e.g. for unit D we have $\frac{|OD|}{|OP|} = 0.75$.

For unit Q we have this measure equal to 1. This fact doesn't mean that Q is fully efficient; one can still increase sales in Q without losing clients to the level of B.)

CCR model

Consider n decision making units (DMUs) $U_1, \ldots, U_n$ transforming m inputs into r outputs. We will use the notation x_{iq} for i-th input of q-th DMU and y_{jq} for j-th output of q-th DMU. The values of U_q can be arranged into vectors $\mathbf{x_q} = (x_{1q}, \ldots, x_{mq})'$, $\mathbf{y_q} = (y_{1q}, \ldots, y_{rq})'$ and these vectors can be arranged into matrices

$$\mathbf{X} = [x_{iq}]_{q=1,\ldots,n}^{i=1,\ldots,m}, \ \mathbf{Y} = [y_{jq}]_{q=1,\ldots,n}^{j=1,\ldots,r}.$$

CCR model

Consider n decision making units (DMUs) $U_1, \ldots, U_n$ transforming m inputs into r outputs. We will use the notation x_{iq} for i-th input of q-th DMU and y_{jq} for j-th output of q-th DMU. The values of U_q can be arranged into vectors $\mathbf{x_q} = (x_{1q}, \ldots, x_{mq})'$, $\mathbf{y_q} = (y_{1q}, \ldots, y_{rq})'$ and these vectors can be arranged into matrices

$$\mathbf{X} = [x_{iq}]_{q=1,\ldots,n}^{i=1,\ldots,m}, \ \mathbf{Y} = [y_{jq}]_{q=1,\ldots,n}^{j=1,\ldots,r}.$$

By introducing nonnegative weights $\mathbf{v} = (v_1, \ldots, v_m)$, $\mathbf{u} = (u_1, \ldots, u_r)$ we can define for every unit U_q its

virtual input = $v_1 x_{1q} + \ldots + v_m x_{mq} = \mathbf{vx_q}$ and
virtual output = $u_1 y_{1q} + \ldots + u_r y_{rq} = \mathbf{uy_q}$.
The efficiency of the decision making unit can be expressed as the ratio of its virtual output and input.

CCR model

Consider n decision making units (DMUs) $U_1, \ldots, U_n$ transforming m inputs into r outputs. We will use the notation x_{iq} for i-th input of q-th DMU and y_{jq} for j-th output of q-th DMU. The values of U_q can be arranged into vectors $\mathbf{x_q} = (x_{1q}, \ldots, x_{mq})'$, $\mathbf{y_q} = (y_{1q}, \ldots, y_{rq})'$ and these vectors can be arranged into matrices

$$\mathbf{X} = [x_{iq}]_{q=1,\ldots,n}^{i=1,\ldots,m}, \ \mathbf{Y} = [y_{jq}]_{q=1,\ldots,n}^{j=1,\ldots,r}.$$

By introducing nonnegative weights $\mathbf{v} = (v_1, \ldots, v_m)$, $\mathbf{u} = (u_1, \ldots, u_r)$ we can define for every unit U_q its

virtual input = $v_1 x_{1q} + \ldots + v_m x_{mq} = \mathbf{vx_q}$ and
virtual output = $u_1 y_{1q} + \ldots + u_r y_{rq} = \mathbf{uy_q}$.
The efficiency of the decision making unit can be expressed as the ratio of its virtual output and input.
In the CCR model we optimize input and output weights so that efficiency of the unit U_q, $z = \frac{u_1 y_{1q} + \ldots + u_r y_{rq}}{v_1 x_{1q} + \ldots + v_m x_{mq}} = \frac{\mathbf{uy_q}}{\mathbf{vx_q}}$ is maximal subject to the constraints that efficiency of other unit is less or equal to 1. If there exist positive solution with optimal objective value $z^* = 1$, the unit U_q is said to be CCR efficient.

Input-oriented CCR model

The model for DMU U_q can be formulated as fractional linear program

$$z = \frac{u_1 y_{1q} + \ldots + u_r y_{rq}}{v_1 x_{1q} + \ldots + v_m x_{mq}} \to max_{u,v}$$

subject to

$$\frac{u_1 y_{1k} + \ldots + u_r y_{rk}}{v_1 x_{1k} + \ldots + v_m x_{mk}} \leq 1,\ k = 1, \ldots n,$$

$$u_i \geq 0,\ v_j \geq 0,\ i = 1, \ldots m,\ j = 1, \ldots r.$$

Input-oriented CCR model

The model for DMU U_q can be formulated as fractional linear program

$$z = \frac{u_1 y_{1q} + \ldots + u_r y_{rq}}{v_1 x_{1q} + \ldots + v_m x_{mq}} \to max_{u,v}$$

subject to

$$\frac{u_1 y_{1k} + \ldots + u_r y_{rk}}{v_1 x_{1k} + \ldots + v_m x_{mk}} \leq 1, \ k = 1, \ldots n,$$
$$u_i \geq 0, \ v_j \geq 0, \ i = 1, \ldots m, \ j = 1, \ldots r.$$

The problem is easy to linearize using Charnes-Cooper transformation:

$$z = u_1 y_{1q} + \ldots + u_r y_{rq} \to max_{u,v}$$

za omezení

$$v_1 x_{1q} + \ldots + v_m x_{mq} = 1$$
$$u_1 y_{1k} + \ldots + u_r y_{rk} \leq v_1 x_{1k} + \ldots + v_m x_{mk}, \ k = 1, \ldots n,$$
$$u_i \geq 0, \ v_j \geq 0, \ i = 1, \ldots m, \ j = 1, \ldots r.$$

Input-oriented CCR model

The model for DMU U_q can be formulated as fractional linear program

$$z = \frac{u_1 y_{1q} + \ldots + u_r y_{rq}}{v_1 x_{1q} + \ldots + v_m x_{mq}} \to max_{u,v}$$

subject to

$$\frac{u_1 y_{1k} + \ldots + u_r y_{rk}}{v_1 x_{1k} + \ldots + v_m x_{mk}} \leq 1, \; k = 1, \ldots n,$$
$$u_i \geq 0, \; v_j \geq 0, \; i = 1, \ldots m, \; j = 1, \ldots r.$$

The problem is easy to linearize using Charnes-Cooper transformation:

$$z = u_1 y_{1q} + \ldots + u_r y_{rq} \to max_{u,v}$$

za omezení

$$v_1 x_{1q} + \ldots + v_m x_{mq} = 1$$
$$u_1 y_{1k} + \ldots + u_r y_{rk} \leq v_1 x_{1k} + \ldots + v_m x_{mk}, \; k = 1, \ldots n,$$
$$u_i \geq 0, \; v_j \geq 0, \; i = 1, \ldots m, \; j = 1, \ldots r.$$

This problem is called Input-oriented CCR model. The set of indexes $k \in \{1, \ldots n\}$ of efficient units, which correspond to binding constraints for U_q, defines peer DMUs for U_q.

CCR model - example

Let's consider the problem of 6 DMUs with two inputs and one output:

Unit	A	B	C	D	E	F
x_1	4	7	8	4	2	10
x_2	3	3	1	2	4	1
y	1	1	1	1	1	1

CCR model - example

Let’s consider the problem of 6 DMUs with two inputs and one output:

Unit	A	B	C	D	E	F
x_1	4	7	8	4	2	10
x_2	3	3	1	2	4	1
y	1	1	1	1	1	1

Linearized problem for unit A is to be:
$z = u \to \max$
subject to

$$\begin{array}{ll} 4v_1 + 3v_2 = 1, & u,\ v_1,\ v_2 \geq 0 \\ u \leq 4v_1 + 3v_2 \text{ (A)} & u \leq 7v_1 + 3v_2 \text{ (B)} \\ u \leq 8v_1 + v_2 \text{ (C)} & u \leq 4v_1 + 2v_2 \text{ (D)} \\ u \leq 2v_1 + 4v_2 \text{ (E)} & u \leq 10v_1 + v_2 \text{ (F)} \end{array}$$

CCR model - example

Let's consider the problem of 6 DMUs with two inputs and one output:

Unit	A	B	C	D	E	F
x_1	4	7	8	4	2	10
x_2	3	3	1	2	4	1
y	1	1	1	1	1	1

Linearized problem for unit A is to be:
$z = u \to$ max
subject to

$$\begin{array}{ll} 4v_1 + 3v_2 = 1, & u,\ v_1,\ v_2 \geq 0 \\ u \leq 4v_1 + 3v_2 \text{ (A)} & u \leq 7v_1 + 3v_2 \text{ (B)} \\ u \leq 8v_1 + v_2 \text{ (C)} & u \leq 4v_1 + 2v_2 \text{ (D)} \\ u \leq 2v_1 + 4v_2 \text{ (E)} & u \leq 10v_1 + v_2 \text{ (F)} \end{array}$$

The problem can be solved by standard methods of linear programming. The solution for DMU A is $z^* = u^* = 6/7,\ v_1^* = v_2^* = 1/7$. Unit A is not efficient, because $z^* = 6/7 \leq 1$. Constraints corresponding to the units D, E are binding, hence D,E are peer units for A.

CCR model - example

Problems for remaining DMUs are similar, they differ only in an equality constraint; it will be $7v_1 + 3v_2 = 1$ for unit B, etc. Table summarizes results for all DMUs:

DMU	x_1	x_2	y	v_1^*	v_2^*	$z^* = u^*$	peer units
A	4	3	1	1/7	1/7	6/7	D,E
B	7	3	1	1/19	4/19	12/19	C,D
C	8	1	1	1/12	1/3	1	C
D	4	2	1	1/6	1/6	1	D
E	2	4	1	3/14	1/7	1	E
F	10	1	1	0	1	1	C

CCR model - example

Problems for remaining DMUs are similar, they differ only in an equality constraint; it will be $7v_1 + 3v_2 = 1$ for unit B, etc. Table summarizes results for all DMUs:

DMU	x_1	x_2	y	v_1^*	v_2^*	$z^* = u^*$	peer units
A	4	3	1	1/7	1/7	6/7	D,E
B	7	3	1	1/19	4/19	12/19	C,D
C	8	1	1	1/12	1/3	1	C
D	4	2	1	1/6	1/6	1	D
E	2	4	1	3/14	1/7	1	E
F	10	1	1	0	1	1	C

We can see that units C,D,E are efficient and their scores are equal to 1. Optimal objective value for F is equal to 1 as well, but its peer unit is C, because it has the same input v_2, but lesser input v_1. It must be emphasized that a unit is CCR efficient, iff $z^* = 1$ and the solution satisfies $u^* > 0, \; v^* > 0$. Requirement on positivity of weights can be embedded directly in the model by changing the right-hand side of the constraint to an $\varepsilon > 0$.

CCR model - dual problem

Let's rewrite the CCR model for U_q using matrix notation:

$z = \mathbf{u}\mathbf{y_q} \rightarrow max_{u,v}$

subject to:

$\mathbf{v}\mathbf{x_q} = 1$

$\mathbf{u}\mathbf{Y} \leq \mathbf{v}\mathbf{X}$,

$\mathbf{u}, \ \mathbf{v} \geq 0$.

CCR model - dual problem

Let's rewrite the CCR model for U_q using matrix notation:

$z = \mathbf{u}\mathbf{y_q} \to max_{u,v}$

subject to:

$\mathbf{v}\mathbf{x_q} = 1$

$\mathbf{u}\mathbf{Y} \leq \mathbf{v}\mathbf{X},$

$\mathbf{u},\ \mathbf{v} \geq 0.$

We can formulate dual problem by introducing dual variables θ and $\lambda = (\lambda_1, \ldots \lambda_n)'$:

$z = \theta \to min_{\theta,\lambda}$

subject to:

$\theta\mathbf{x_q} \geq \mathbf{X}\lambda,$

$\mathbf{Y}\lambda \geq \mathbf{y_q},$

$\lambda \geq 0$

Using duality is suitable for computational and interpretation reasons.

Dual CCR model - interpretation

The model looks for a virtual unit with inputs and outputs $\mathbf{X}\lambda, \mathbf{Y}\lambda$ that is better or at least comparable to the radial projection of the evaluated unit U_q at the effective boundary: $\theta\mathbf{x_q} \geq \mathbf{X}\lambda, \mathbf{Y}\lambda \geq \mathbf{y_q}$

Dual CCR model - interpretation

The model looks for a virtual unit with inputs and outputs $\mathbf{X}\lambda, \mathbf{Y}\lambda$ that is better or at least comparable to the radial projection of the evaluated unit U_q at the effective boundary: $\theta \mathbf{x_q} \geq \mathbf{X}\lambda, \mathbf{Y}\lambda \geq \mathbf{y_q}$

The unit U_q under evaluation lies directly at the efficient frontier if it is identical to the corresponding virtual unit. Therefore, the optimal value of the objective function θ^* (so-called Farrell efficiency) must necessarily be equal to 1. It represents necessary radial reduction of inputs needed for achieving the frontier performance. When condition $\theta^* = 1$ holds, the unit U_q is called technically efficient.

Dual CCR model - interpretation

The model looks for a virtual unit with inputs and outputs $\mathbf{X}\lambda, \mathbf{Y}\lambda$ that is better or at least comparable to the radial projection of the evaluated unit U_q at the effective boundary: $\theta\mathbf{x_q} \geq \mathbf{X}\lambda, \mathbf{Y}\lambda \geq \mathbf{y_q}$

The unit U_q under evaluation lies directly at the efficient frontier if it is identical to the corresponding virtual unit. Therefore, the optimal value of the objective function θ^* (so-called Farrell efficiency) must necessarily be equal to 1. It represents necessary radial reduction of inputs needed for achieving the frontier performance. When condition $\theta^* = 1$ holds, the unit U_q is called technically efficient.

To achieve CCR-efficiency, it is necessary that all slacks in the inequality constraints are zeros:
$\mathbf{s}^- = \theta\mathbf{x_q} - \mathbf{X}\lambda = \mathbf{0}, \mathbf{s}^+ = \mathbf{Y}\lambda - \mathbf{y_q} = \mathbf{0}$.

Dual CCR model - interpretation

The model looks for a virtual unit with inputs and outputs $\mathbf{X}\lambda, \mathbf{Y}\lambda$ that is better or at least comparable to the radial projection of the evaluated unit U_q at the effective boundary: $\theta \mathbf{x_q} \geq \mathbf{X}\lambda, \mathbf{Y}\lambda \geq \mathbf{y_q}$

The unit U_q under evaluation lies directly at the efficient frontier if it is identical to the corresponding virtual unit. Therefore, the optimal value of the objective function θ^* (so-called Farrell efficiency) must necessarily be equal to 1. It represents necessary radial reduction of inputs needed for achieving the frontier performance. When condition $\theta^* = 1$ holds, the unit U_q is called technically efficient.

To achieve CCR-efficiency, it is necessary that all slacks in the inequality constraints are zeros:
$\mathbf{s}^- = \theta \mathbf{x_q} - \mathbf{X}\lambda = \mathbf{0}, \mathbf{s}^+ = \mathbf{Y}\lambda - \mathbf{y_q} = \mathbf{0}$.

When all the above mentioned conditions are satisfied, the unit is efficient according to Pareto-Koopmans definition of efficiency, i.e. it is not possible to improve any input or output without worsening the other.

Dual CCR model - alternative formulation

Consider primal CCR model with positivity constraint for weights $\mathbf{u}, \ \mathbf{v} \geq \varepsilon > 0$ and denote $\mathbf{e} = (1, \ldots, 1)^\top$. Dual problem can be formulated as follows

$$z = \theta - \varepsilon \cdot (\mathbf{e} \cdot \mathbf{s}^+ + \mathbf{e} \cdot \mathbf{s}^-) \rightarrow min_{\theta, \lambda, \mathbf{s}^+, \ \mathbf{s}^-}$$

subject to

$\mathbf{s}^- = \theta \mathbf{x_q} - \mathbf{X}\lambda,$

$\mathbf{s}^+ = \mathbf{Y}\lambda - \mathbf{y_q},$

$\lambda, \ \mathbf{s}^+, \ \mathbf{s}^- \geq 0$

Dual CCR model - alternative formulation

Consider primal CCR model with positivity constraint for weights $\mathbf{u},\ \mathbf{v} \geq \varepsilon > 0$ and denote $\mathbf{e} = (1, \ldots, 1)^\top$. Dual problem can be formulated as follows

$$z = \theta - \varepsilon \cdot (\mathbf{e} \cdot \mathbf{s}^+ + \mathbf{e} \cdot \mathbf{s}^-) \to min_{\theta,\lambda,\mathbf{s}^+,\ \mathbf{s}^-}$$

subject to

$$\mathbf{s}^- = \theta \mathbf{x_q} - \mathbf{X}\lambda,$$
$$\mathbf{s}^+ = \mathbf{Y}\lambda - \mathbf{y_q},$$
$$\lambda,\ \mathbf{s}^+,\ \mathbf{s}^- \geq 0$$

Optimal solution for U_q defines how to improve inputs and outputs to $\mathbf{x}'_\mathbf{q},\ \mathbf{y}'_\mathbf{q}$ by means of CCR - projection:

$$\mathbf{x}'_\mathbf{q} = \theta^* \mathbf{x_q} - \mathbf{s}^{-*}, \mathbf{y_q}' = \mathbf{y_q} + \mathbf{s}^{+*},$$

or equivalently

$$\mathbf{x}'_\mathbf{q} = \mathbf{X}\lambda^*, \mathbf{y_q}' = \mathbf{Y}\lambda^*.$$

The indexes $j \in \{1, \ldots, n\}$, corresponding to positive λ_j^*, determine peer units for U_q.

Output-oriented CCR model

We can also define Output-oriented CCR model for U_q, let's formulate it using modified dual problem:

$z = \Theta + \varepsilon \cdot (\mathbf{e} \cdot \mathbf{s}^+ + \mathbf{e} \cdot \mathbf{s}^-) \to max_{\Theta, \lambda, \mathbf{s}^+,\ \mathbf{s}^-}$

subject to

$\mathbf{s}^- = \mathbf{x_q} - \mathbf{X}\lambda,$

$\mathbf{s}^+ = \mathbf{Y}\lambda - \Theta\mathbf{y_q},$

$\lambda,\ \mathbf{s}^+,\ \mathbf{s}^- \geq 0$

Output-oriented CCR model

We can also define Output-oriented CCR model for U_q, let's formulate it using modified dual problem:

$$z = \Theta + \varepsilon \cdot (\mathbf{e} \cdot \mathbf{s}^{+} + \mathbf{e} \cdot \mathbf{s}^{-}) \rightarrow max_{\Theta, \lambda, \mathbf{s}^{+},\ \mathbf{s}^{-}}$$

subject to

$$\mathbf{s}^{-} = \mathbf{x_q} - \mathbf{X}\lambda,$$
$$\mathbf{s}^{+} = \mathbf{Y}\lambda - \Theta\mathbf{y_q},$$
$$\lambda,\ \mathbf{s}^{+},\ \mathbf{s}^{-} \geq 0$$

If $\Theta^{*} > 1$, the unit U_q is not efficient and value of Θ^{*} expresses necessary proportional increase of outputs. CCR projection can be defined analogically as

$$\mathbf{x}'_{\mathbf{q}} = \mathbf{X}\lambda^{*}, \mathbf{y_q}' = \mathbf{Y}\lambda^{*}.$$

Output-oriented CCR model

We can also define Output-oriented CCR model for U_q, let's formulate it using modified dual problem:

$$z = \Theta + \varepsilon \cdot (\mathbf{e} \cdot \mathbf{s}^{+} + \mathbf{e} \cdot \mathbf{s}^{-}) \rightarrow max_{\Theta, \lambda, \mathbf{s}^{+},\ \mathbf{s}^{-}}$$

subject to

$\mathbf{s}^{-} = \mathbf{x_q} - \mathbf{X}\lambda,$
$\mathbf{s}^{+} = \mathbf{Y}\lambda - \Theta\mathbf{y_q},$
$\lambda,\ \mathbf{s}^{+},\ \mathbf{s}^{-} \geq 0$

If $\Theta^{*} > 1$, the unit U_q is not efficient and value of Θ^{*} expresses necessary proportional increase of outputs. CCR projection can be defined analogically as

$$\mathbf{x}'_{\mathbf{q}} = \mathbf{X}\lambda^{*}, \mathbf{y_q}' = \mathbf{Y}\lambda^{*}.$$

It is true in CCR model that the input-oriented efficiency is the reciprocal of output-oriented efficiency, $\theta^{*} \cdot \Theta^{*} = 1$

Output-oriented BCC model

As a modification of the CCR model, which assumes constant returns to scale and defines a conical envelope of the data, Banker, Charnes and Cooper introduced a model using variable returns to scale, so-called BCC model. This approach defines a convex envelope of the data, virtual units are not arbitrary non-negative combinations $\mathbf{X}\lambda,\ \mathbf{Y}\lambda$, the coefficients must also satisfy another condition, $\mathbf{e}\lambda = 1$. Due to this additional constraint, there are usually more units efficient under BCC specification.

Output-oriented BCC model

As a modification of the CCR model, which assumes constant returns to scale and defines a conical envelope of the data, Banker, Charnes and Cooper introduced a model using variable returns to scale, so-called BCC model. This approach defines a convex envelope of the data, virtual units are not arbitrary non-negative combinations $\mathbf{X}\lambda,\ \mathbf{Y}\lambda$, the coefficients must also satisfy another condition, $\mathbf{e}\lambda = 1$. Due to this additional constraint, there are usually more units efficient under BCC specification.
Let's formulate the input-oriented BCC model:

$$z = \theta - \varepsilon \cdot (\mathbf{e} \cdot \mathbf{s}^{+} + \mathbf{e} \cdot \mathbf{s}^{-}) \rightarrow min_{\theta,\lambda,\mathbf{s}^{+},\ \mathbf{s}^{-}}$$

subject to

$\mathbf{s}^{-} = \theta \mathbf{x_q} - \mathbf{X}\lambda,$

$\mathbf{s}^{+} = \mathbf{Y}\lambda - \mathbf{y_q},$

$\mathbf{e}\lambda = 1$

$\lambda,\ \mathbf{s}^{+},\ \mathbf{s}^{-} \geq 0$

Output-oriented BCC model

As a modification of the CCR model, which assumes constant returns to scale and defines a conical envelope of the data, Banker, Charnes and Cooper introduced a model using variable returns to scale, so-called BCC model. This approach defines a convex envelope of the data, virtual units are not arbitrary non-negative combinations $\mathbf{X}\lambda,\ \mathbf{Y}\lambda$, the coefficients must also satisfy another condition, $\mathbf{e}\lambda = 1$. Due to this additional constraint, there are usually more units efficient under BCC specification.

Let's formulate the input-oriented BCC model:

$$z = \theta - \varepsilon \cdot (\mathbf{e} \cdot \mathbf{s}^{+} + \mathbf{e} \cdot \mathbf{s}^{-}) \rightarrow min_{\theta,\lambda,\mathbf{s}^{+},\ \mathbf{s}^{-}}$$

subject to

$$\mathbf{s}^{-} = \theta \mathbf{x_q} - \mathbf{X}\lambda,$$
$$\mathbf{s}^{+} = \mathbf{Y}\lambda - \mathbf{y_q},$$
$$\mathbf{e}\lambda = 1$$
$$\lambda,\ \mathbf{s}^{+},\ \mathbf{s}^{-} \geq 0$$

BCC efficient are those units that have $\theta^{*} = 1, \mathbf{s}^{+*} = \mathbf{0}, \mathbf{s}^{-*} = \mathbf{0}$.

Alternative DEA models

In addition to the basic DEA models, a number of modifications have been proposed, among others:

- Additive model SBM (Slack-Based Measure) model: it is not necessary to specify input or output orientation, the efficiency is expressed using additional variables $\mathbf{s}^-$, $\mathbf{s}^+$
- DEA models with non-controllable inputs or outputs
- DEA models with undesirable inputs or outputs
- Superefficiency models
- discrete models, e.g. FDH (Free Disposable Hull) model
- Malmquist index for the efficiency change in time

Methods are described in more detail in the books

- S. C. Ray: Data Envelopment Analysis: Theory and Techniques for Economics and Operations Research, Cambridge 2004
- W. W. Cooper, L. M. Seiford, K. Tone: Data Envelopement Analysis, Springer, New York 2007