MSIE 4.0

Co-funded by the
Erasmus+ Programme
of the European Union

Advanced Optimization: Techniques and Industrial Applications

Curriculum Development
of Master's Degree Program in
Industrial Engineering for Thailand Sustainable Smart Industry

MSIE 4.0

Co-funded by the
Erasmus+ Programme
of the European Union

Session 1.3:
Integer/Mixed Integer Programming & Combinatorial Optimization

Curriculum Development
of Master's Degree Program in
Industrial Engineering for Thailand Sustainable Smart Industry

Introduction

Consider the standard LP:

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j = b_i \geq 0 \qquad \forall i = 1,2,\dots,m$$

$$x_j \geq 0 \qquad \forall j = 1,2,\dots,n$$

- If all variables are integers: Pure Integer Program (IP or ILP)
- If some variables are integers: Mixed Integer Program (MIP)
- If all variables are either 0 or 1: Zero-One Integer Program

Co-funded by the Erasmus+ Programme of the European Union

Introduction

Example 1: 0-1 Knapsack Problem

Consider *n* items to be carried. Item j $(j = 1,2,..,n)$ has a value c_j and a weight a_j. The limited weight that can be carried is *K*. The problem is to select the items so as to maximize the total value.

Decision variable: $$x_j = \begin{cases} 1 & \text{if item } j \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

The IP model: Maximize $\sum_{j=1}^{n} c_j x_j$

s.t. $\sum_{j=1}^{n} a_j x_j \leq K$

$x_j = 0 \text{ or } 1 \quad \forall j$

Co-funded by the Erasmus+ Programme of the European Union

Introduction

Example 2: Either-Or Problem

Consider an IP in which *one and only one* of the following two constraints holds

$$\sum_j a_{1j}x_j \leq b_1 \quad \text{or} \quad \sum_j a_{2j}x_j \leq b_2$$

This type of constraint can be reformulated by introducing a 0-1 variable y and sufficiently large number M such that:

$$\sum_j a_{1j}x_j - b_1 \leq My \qquad (1)$$
$$\sum_j a_{2j}x_j - b_2 \leq M(1-y) \qquad (2)$$

Note that:

- If $y = 0$: (2) becomes redundant
- If $y = 1$: (1) becomes redundant

Co-funded by the Erasmus+ Programme of the European Union

Introduction

In general, if we have m constraints $g_i(x) \leq 0$ $(i = 1,2, \dots, m)$ and we want k constraints out of m to hold $(k < m)$ then we can introduce m 0-1 variables y_1, y_2, …, y_m and a very large numbers M such that:

$$g_i(x) \leq My_i \qquad (i = 1,2, \dots, m)$$
$$y_1 + y_2 + .. + y_m = m - k$$
$$y_i = 0 \text{ or } 1 \qquad (i = 1,2, \dots, m)$$

Co-funded by the
Erasmus+ Programme
of the European Union

Introduction

Example 3: Sequencing Problem

Consider the problem in which the optimal processing sequence of n jobs on a machine is determined. Assume that the machine can only process one job at a time.

Denote p_i : processing time of job i

t_i : start time of job i

For a pair of jobs (i, j) there are only two options:

1. Job i starts before job j or: $t_j \geq t_i + p_i$
2. Job j starts before job i or: $t_i \geq t_j + p_j$

Co-funded by the Erasmus+ Programme of the European Union

Introduction

Introduce a very large number M and 0-1 variables y_{ij} in which $y_{ij} = 1$ if job i is processed before job j, $y_{ij} = 0$ otherwise. The above options can be described as follows:

$$My_{ij} + t_i - t_j \geq p_j$$

$$M(1 - y_{ij}) + t_j - t_i \geq p_i$$

$$y_{ij} = 0 \text{ or } 1$$

Co-funded by the
Erasmus+ Programme
of the European Union

Introduction

Example 4: Warehouse Location Problem

Consider the decision to open a number of warehouses at potential sites. Denote:

y_i : 0-1 variable; =1 if a warehouse is opened at site *i*, =0 otherwise
f_i : fixed operation cost of the warehouse at site *i*, if opened
x_{ij} : amount of goods to be shipped from warehouse at site *i*, if opened, to customer *j*,
c_{ij} : operation & transportation cost per unit from warehouse at site *i* to customer *j*
d_j : demand of customer *j*
s_i : supply capacity of warehouse at site *i*, if opened

Co-funded by the Erasmus+ Programme of the European Union

Introduction

The problem can be formulated as a mixed IP:

$$\text{Min} \quad \text{Cost} = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} \leq s_i y_i \qquad \forall i$$

$$\sum_{i=1}^{m} x_{ij} \geq d_j \qquad \forall j$$

$$x_{ij} \geq 0 \qquad \forall i, j$$

Co-funded by the
Erasmus+ Programme
of the European Union

Introduction

Example 5: Traveling Salesman Problem

A traveling salesman has to visit each of n sites $C_1, C_2, \ldots, C_n$. He must start from his office located at site C_1 and return to C_1 after visit each site exactly once. The distance between C_i and C_j is c_{ij}. The problem is to find the route which minimizes the total distance traveled.

Denote $$x_{ij} = \begin{cases} 1 & \text{if the route includes the arc } (C_i, C_j) \\ 0 & \text{otherwise} \end{cases}$$

Co-funded by the Erasmus+ Programme of the European Union

Introduction

The problem can be formulated as a mixed IP as follows:

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

s.t. $\sum_{j=1}^{n} x_{ij} = 1 \ \forall i$

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

$x_{ij} \geq 0 \qquad \forall i,j$

$u_i - u_j + nx_{ij} \leq n-1 \quad i,j = 2,3,..,n; \ i \neq j$(*)

$x_{ij} = 0 \text{ or } 1 \ \forall i,j$

$u_i \geq 0 \text{ and integer } \forall i = 2,3,..,n$

The role of (*): To eliminate solutions that contain a subtour (disconnected route)

Co-funded by the Erasmus+ Programme of the European Union

Introduction

- In fact, if a “solution” contains a subtour, it will be infeasible.

Consider a subtour $C_{i_1}, C_{i_2}, \ldots, C_{i_k}, C_{i_{k+1}} \equiv C_{i_1}$ $(k < n, i_k \neq 1)$. Note that $x_{i_j,i_{j+1}} = 1$ $(j = 1,2, \ldots, k)$, we have:

$$u_{i_1} - u_{i_2} + n \leq n - 1$$
$$u_{i_2} - u_{i_3} + n \leq n - 1$$
$$\ldots$$
$$u_{i_k} - u_{i_1} + n \leq n - 1$$
$$\Rightarrow \quad kn \leq k(n-1) \text{ : impossible!}$$

- For any feasible solution, there exist values of the $u_i's$ that satisfy (*)

In fact, let $u_i =$ the location of C_i in the tour then (*) will be satisfied.

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach

It should be noted that the solution of an IP model could not be simply achieved by solving the LP relaxation problem and then rounding off its solution.

Example 6: Consider the IP model

$$\begin{aligned} \text{Max} \quad & z = 5x_1 + 8x_2 \\ \text{s.t.} \quad & 5x_1 + 3x_2 \leq 30 \\ & 5x_1 + 9x_2 \leq 45 \\ & x_1, x_2 \geq 0 \text{ and integer} \end{aligned}$$

Solution of the LP relaxation: $x_1 = 4.5, x_2 = 2.5$

Solution of the IP: $x_1 = 0, x_2 = 5$

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

Applied for "Pure Integer Programs"

Consider the optimal simplex tableau of the LP relaxation of a pure IP model:

	Z	x_j	RHS
Z	1	y_{oj}	y_{00}
$\mathbf{x_B}$	0 $\vdots$	y_{1j} $\vdots$	y_{10} $\vdots$
	0 $\vdots$	y_{ij} $\vdots$	y_{i0} $\vdots$
	0	y_{mj}	y_{m0}

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

From the above simplex tableau, we have:

$$\sum_{j=1}^{n} y_{ij} x_j = y_{i0} \qquad \forall i \qquad (*)$$

Denote $[k]$ to be the integer part of k, i.e., $k = [k] + f \ (0 \leq f < 1)$

Then for a non-integer value y_{i0}, we have: $\sum_{j=1}^{n} [y_{ij}] x_j \leq y_{i0}$

Since x_j is integer, we can rewrite the above inequality as:

$$\sum_{j=1}^{n} [y_{ij}] x_j \leq [y_{i0}] \qquad (**)$$

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

From (*) and (**), we can derive:

$$\sum_{j=1}^{n}([y_{ij}] - y_{ij})x_j \leq [y_{i0}] - y_{i0}$$

Note that $y_{ij} = 0$ or 1 for all $j \in \mathbf{B} \Rightarrow [y_{i0}] - y_{i0}$ for all $j \in \mathbf{B}$. Hence, the above inequality is equivalent to

$$\sum_{j \in \boldsymbol{D}}([y_{ij}] - y_{ij})x_j \leq [y_{i0}] - y_{i0}$$

This inequality is called a *Gomory cut*.

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

It is noted that:

1. The noninteger optimal solution for the LP relaxation problem do not satisfy the Gomory constraint due to

    - LHS = 0 because of $x_j = 0 \; \forall j \in \boldsymbol{D}$ while
    - RHS <0 because y_{i0} is not an integer.

2. All integer feasible solutions for the LP relaxation problem will satisfy this constraint.

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

Procedure of Cutting Plane Method

Step 1: Solve the LP relaxation problem
If the optimal solution is integer, stop. The optimal solution of the original IP has been found. Otherwise, go to step 2.

Step 2: Select the constraint in the final simplex tableau with the largest value of $y_{i0} - [y_{i0}]$. Add the associated Gomory cut in the set of constraints and go back to step 1.

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

Example 7: Consider the IP program:

$$\begin{aligned} \text{Max} \quad & z = -5x_1 - 6x_2 \\ \text{s.t.} \quad & 10x_1 + 3x_2 \leq 52 \\ & 2x_1 + 3x_2 \leq 18 \\ & x_1, x_2 \geq 0 \text{ and integer} \end{aligned}$$

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

Optimal simplex tableau of the LP relaxation problem:

	Z	x_1	x_2	x_3	x_4	RHS
Z	1	0	0	-1/8	-15/8	-161/4
x_1	0	1	0	1/8	-1/8	17/4
x_2	0	0	1	-1/12	5/12	19/6

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

Both x_1 and x_2 are not integers. Introduce a cutting plane associated with the basic variable having largest fraction, i.e., x_1:

$$\left(\left[\frac{1}{8}\right] - \frac{1}{8}\right) x_3 + \left(\left[-\frac{1}{8}\right] - \left(-\frac{1}{8}\right)\right) x_4 \leq \left[\frac{17}{4}\right] - \frac{17}{4}$$

$$\Leftrightarrow \quad -\frac{1}{8}\, x_3 - \frac{7}{8} x_4 \leq -\frac{1}{4}$$

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

The updated simplex tableau:

	Z	x_1	x_2	x_3	x_4	x_5	RHS
Z	1	0	0	-1/8	-15/8	0	-161/4
x_1	0	1	0	1/8	-1/8	0	17/4
x_2	0	0	1	-1/12	5/12	0	19/6
x_5	0	0	0	-1/8*	-7/8	1	-1/4

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

Applying the dual simplex method we obtain:

	Z	x_1	x_2	x_3	x_4	x_5	RHS
Z	1	0	0	0	-1	-1	-40
x_1	0	1	0	0	-1	1	4
x_2	0	0	1	0	1	-2/3	10/3
x_3	0	0	0	1	7	-8	2

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

Since x_2 is still not an integer, introduce another Gomory cut associated with x_2:

$$([1]-1)x_4+\left(\left[-\frac{2}{3}\right]-\left(-\frac{2}{3}\right)\right)x_5 \leq \left[\frac{10}{3}\right]-\frac{10}{3}$$

$$\Leftrightarrow \quad -\frac{1}{3}\,x_5 \leq -\frac{1}{3}$$

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

The updated simplex tableau:

	Z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
Z	1	0	0	0	-1	-1	0	-40
x_1	0	1	0	0	-1	1	0	4
x_2	0	0	1	0	1	-2/3	0	10/3
x_3	0	0	0	1	7	-8	0	2
x_6	0	0	0	0	0	-1/3	1	-1/3

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

Applying the dual simplex method we obtain:

	Z	x_1	x_2	x_3	x_4	x_5	x_6	RHS
Z	1	0	0	0	-1	0	-3	-39
x_1	0	1	0	0	-1	0	3	3
x_2	0	0	1	0	1	0	-2	4
x_3	0	0	0	1	7	0	-24	10
x_5	0	0	0	0	0	1	-3	1

Optimal solution: $x_1^* = 3, x_2^* = 4$; value of objective function: $z^* = -39$.

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

Notes:

- From the original constraints and the first Gomory cut, we have:

$$x_3 = 52 - (10x_1 + 3x_2); \qquad x_4 = 18 - (2x_1 + 3x_2)$$

$$x_5 = -\frac{1}{4} + \frac{x_3 + x_4}{8} = 22 - 3(x_1 + x_2)$$

Hence, the first and the second cutting planes are equivalent to:

$$x_1 + x_2 \leq {}^{22}/_{3}$$

$$x_1 + x_2 \leq 7$$

The cutting planes are parallel.

- The cutting plane algorithm will converge after a finite number of iterations

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

Theorem: If LP relaxation problem is unbounded, the IP problem is either unbounded or infeasible

Proof: If the LP relaxation problem is unbounded, then

$$\exists j:\ y_{0j} > 0\ and\ y_{ij} \leq 0\ \forall i = 1,2,\dots,m$$

Consider a direction $\mathbf{d} > \mathbf{0}$ defined by $\begin{cases} d_{\boldsymbol{B}_i} = -y_{ij} > 0 & i = 1,2,\dots,m \\ d_j = 1 & \\ d_k = 0 & k \neq i,j \end{cases}$

We have:

$$\mathbf{c}^T\mathbf{d} = \sum_{i=1}^{m} c_{x_{\boldsymbol{B}_i}}\left(-y_{ij}\right) + c_j = -\mathbf{c}_\mathbf{B}^T\mathbf{B}^{-1}\mathbf{A}_j + c_j = -y_{oj} < 0$$

$$\mathbf{Ad} = \mathbf{A}_j - \sum_{i=1}^{m} y_{ij}\mathbf{A}_{\boldsymbol{B}_i} = \mathbf{0}$$

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
CUTTING PLANE ALGORITHM

It is noted that we can define an infinite series $\{\lambda_i\}$, $\lambda_i < \lambda_{i+1}$ such that $\lambda_i \mathbf{d}$ is integer for $\forall i$. Therefore, if the IP program is feasible, i.e., there exists an integer feasible solution $\bar{\mathbf{x}}$ ($\mathbf{A}\bar{\mathbf{x}} = \mathbf{b}$ and integer) then $\bar{\mathbf{x}} + \lambda_i \mathbf{d}$ is also an integer feasible solution for all i and:

$$\mathbf{c}^T(\bar{\mathbf{x}} + \lambda_i \mathbf{d}) = \mathbf{c}^T\bar{\mathbf{x}} + \lambda_i \mathbf{c}^T\mathbf{d} = \mathbf{c}^T\bar{\mathbf{x}} - \lambda_i y_{0j} < \mathbf{c}^T\bar{\mathbf{x}}$$

Therefore, the IP is also unbounded.

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

The branch and bound algorithm can be applied for both pure IP and mixed IP programs. However, in many practical applications, the algorithm is developed based on the specific structure of the IP problem under consideration

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

In general, the branch and bound procedure using LP relaxation method for a minimization problem is as follows:

1. Branching

Divide the feasible region into a finite number of subregions

2. Bounding

Develop bounds for the optimal objective value z^* at each subdivided subregion: $\underline{z} \leq z^* \leq \overline{z}$. The upper bound $\overline{z}$ is the smallest objective value of any *feasible integer solution* encountered and the lower bound $\underline{z}$ is the smallest objective value at any *active* (or *unfathomed*) subregion.

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

3. Fathoming

The subregion L_j is fathomed if

a. LP relaxation over L_j is infeasible (*infeasibility*)

b. The optimal LP relaxation over L_j is integer (*integrality*)

c. The optimal LP relaxation solution over L_j is greater than or equal to $\overline{z}$ (*dominance*)

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

Remark:

In case of maximization problem:

- The upper bound $\bar{z}$ is the largest objective value at any *active* (or *unfathomed*) subregion
- The lower bound $\underline{z}$ is the largest *feasible integer solution* encountered.
- Fathoming criteria 3c. is: The optimal LP relaxation solution over L_j is less than or equal to $\underline{z}$ (dominance)

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

Branching procedure:

1. Depth-first search or LIFO

If the current node is not fathomed, the next node considered is one of its two sons. We can:

- Arbitrarily select the left son or the right son
- Select the son that has the smaller lower bound (for minimization problem) or the larger upper bound (for maximization problem)

When a node is fathomed, go back toward the root until we find the first node that has a son that has not yet been considered

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

2. Breadth-first search

All the nodes at a given level are considered before any nodes at the next lower level. Note that the level of a node in the enumeration tree is the number of arcs in the unique path between it and the root.

3. Best lower bound

When a node has been fathomed, select the next node from all active nodes that has the smallest lower bound (for minimization problem) or the largest upper bound (for minimization problem)

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

Example 8: Consider the problem

$$
\begin{aligned}
\text{(IP0)}\quad \text{Min}\quad & z = -5x_1 - 4x_2 \\
\text{s.t.}\quad & x_1 + x_2 \leq 5 \\
& 10x_1 + 6x_2 \leq 45 \\
& x_1, x_2 \geq 0 \text{ and integer}
\end{aligned}
$$

Solution of the LP relaxation problem (LP0):

$$x_1 = 3.75, x_2 = 1.25 \text{ and } z = -23.75$$

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

Branching variable: $x_1 \Rightarrow$ The feasible region is divided into two subregions

1. Subregion 1: defined by original constraints and $x_1 \leq 3$
2. Subregion 2: defined by original constraints and $x_1 \geq 4$

If the integer requirement of variables is taken into consideration, the problem (LP0) is equivalent to the two subproblems (LP1) and (LP2) associated with the two subregions defined above.

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

Consider

$$\begin{aligned} \text{(LP1)}: \ \text{Min} \ & z = -5x_1 - 4x_2 \\ \text{s.t.} \ & x_1 + x_2 \leq 5 \\ & 10x_1 + 6x_2 \leq 45 \\ & x_1 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Solution of (LP1): $x_1 = 3, x_2 = 2$ and $z = -23$

(LP1) is fathomed due to integrality and $\bar{z} = -23$, i.e., $z^* \leq -23$

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

The next step should be to consider (LP2). However, due to the fact that all coefficients of the objective function are integers and that the optimal objective value of (LP0) is -23.75, there exists no subproblem of (LP0) that can give a better solution than -23. Therefore, there is no need to consider (LP2).

Solution of (LP1) is then the optimal solution of the original (IP0) problem

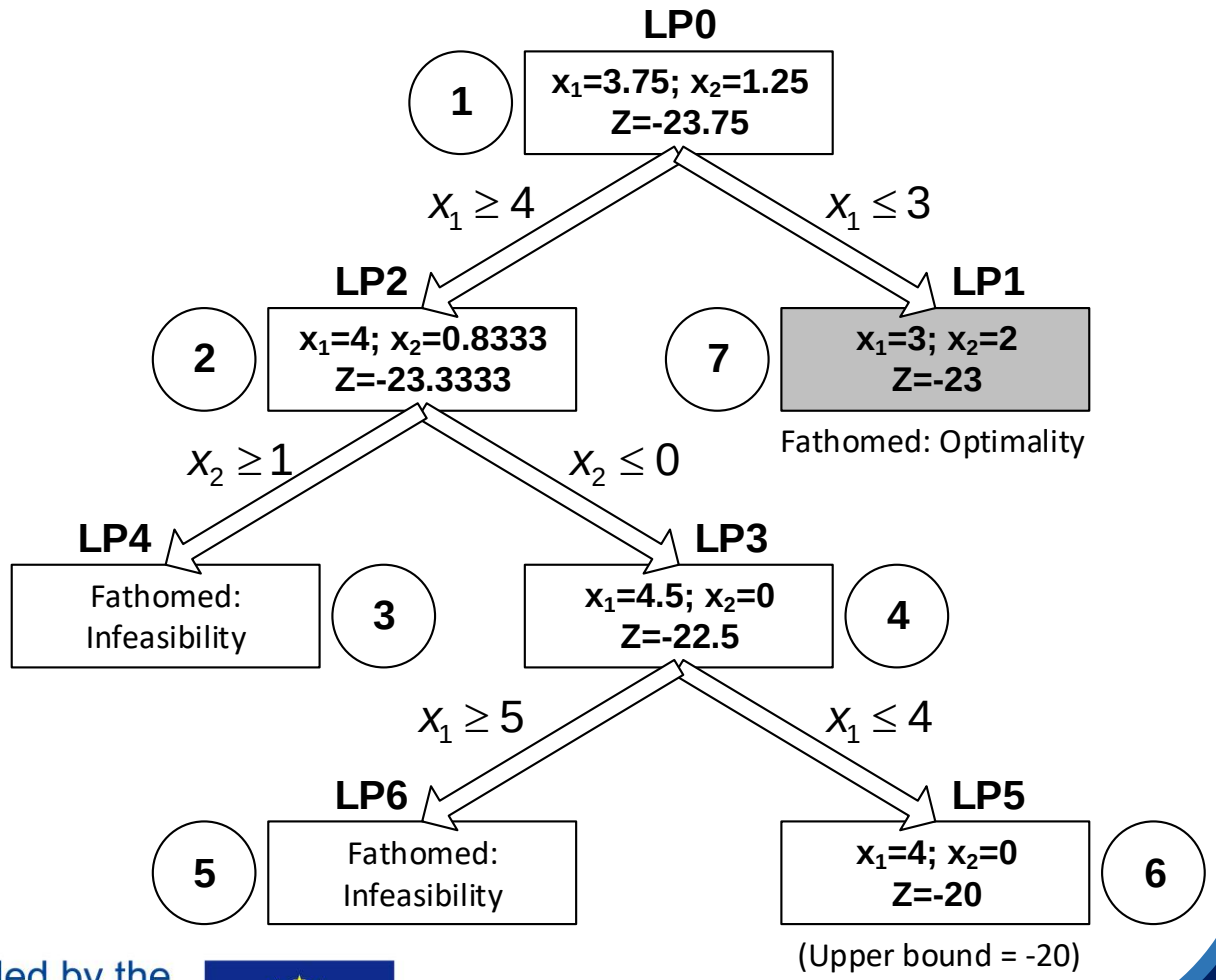
Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

If (LP2) is examined before (LP1): using depth-first search technique, the solution can be determined as follows:

Instead of using x_1 as branching variable, we can use x_2 (the student is recommended to examine this case).

Note: There is no general guidance for the selection of branching variable



Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM

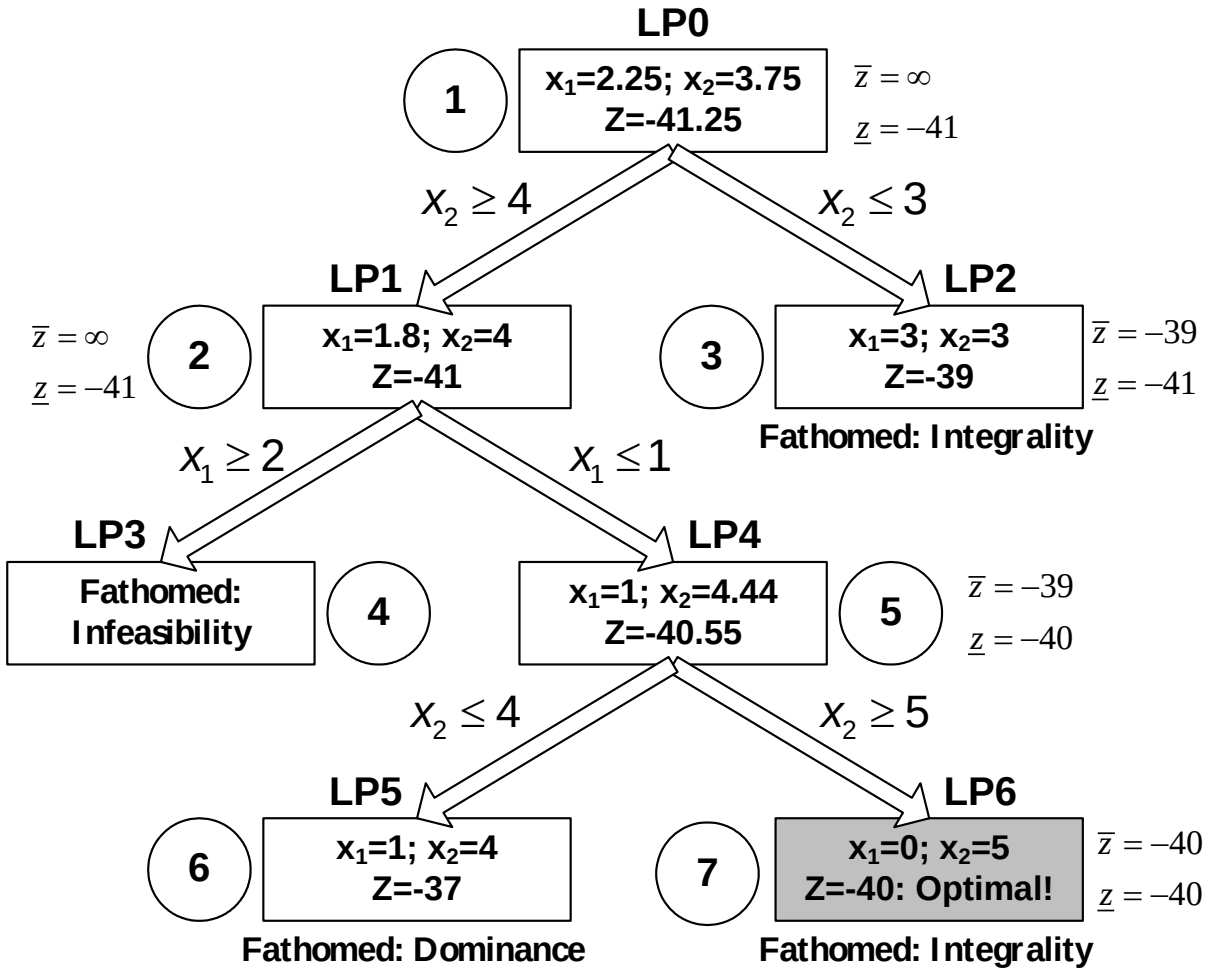
Example 9: Consider the problem

$$
\begin{aligned}
\text{(IP0)}\quad \text{Min}\quad & z = -5x_1 - 8x_2 \\
\text{s.t.}\quad & x_1 + x_2 \leq 6 \\
& 5x_1 + 9x_2 \leq 45 \\
& x_1, x_2 \geq 0 \text{ and integer}
\end{aligned}
$$

Applying the breadth-first search technique, the upper & lower bounds of each subproblem and the final solution of the problem can be illustrated in the following picture

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM



Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

LP Relaxation Technique: (for minimization problem)

The B&B algorithm applied for 0-1 integer program is performed as follows:

- Assign the current best objective value $z^* = \infty$. Solve the LP relaxation problem (LP0) associated with the original (IP0) problem. If (LP0) is not fathomed then start the iterative process

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

- In each iteration, perform the following steps

Branching:

Among unfathomed subproblems, select the most recently emerging subproblem (depth-first search). Breaking the tie by selecting the subproblem with smaller lower bound. Branching by use of some 0-1 variable of the selected subproblem.

Defining the lower bound:

For each subproblem, solve the associated LP relaxation problem in order to determine the lower bound and update the current best objective value z^* (if obtain integer solution from LP relaxation problem)

Checking fathoming condition:

Use fathoming criteria to check if the investigated subproblem is fathomed.

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

The above procedure is performed until all subproblems are fathomed. The current best objective value is the optimal objective value.

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

Fathoming criteria:

1. Solution of LP relaxation problem of the subproblem is integer
2. Lower bound of the subproblem is greater than z^*
3. LP relaxation problem is infeasible.

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

Example 10: Consider

$$
\begin{aligned}
\text{(IP0)}\quad \text{Min}\quad & z = -9x_1 - 5x_2 - 6x_3 - 4x_4 \\
\text{s.t.}\quad & 6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10 \\
& x_3 + x_4 \leq 1 \\
& -x_1 + x_3 \leq 0 \\
& -x_2 + x_4 \leq 0 \\
& x_j = 0,1 \ \ \forall j = 1,2,3,4
\end{aligned}
$$

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

LP relaxation problem:

$$\begin{aligned}
\text{(LP0)}\ \text{Min}\quad & z = -9x_1 - 5x_2 - 6x_3 - 4x_4 \\
\text{s.t.}\quad & 6x_1 + 3x_2 + 5x_3 + 2x_4 \le 10 \\
& x_3 + x_4 \le 1 \\
& -x_1 + x_3 \le 0 \\
& -x_2 + x_4 \le 0 \\
& 0 \le x_j \le 1 \ \ \forall j = 1,2,3,4
\end{aligned}$$

Solution of (LP0): $(x_1, x_2, x_3, x_4) = (\frac{5}{6}, 1, 0, 1);\ \ z = -16\frac{1}{2}$

Lower bound of (IP0): $\underline{z}_{IP0} = -16$

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

- Iteration 1: Branching variable x_1. The two subproblems:

(IP1):

$$\text{Min} \quad Z = -5x_2 - 6x_3 - 4x_4$$

s.t.

$$3x_2 + 5x_3 + 2x_4 \le 10$$
$$x_3 + x_4 \le 1$$
$$x_3 \le 0$$
$$-x_2 + x_4 \le 0$$
$$x_j = 0,1 \ \forall j = 2,3,4$$

(IP2):

$$\text{Min} \quad Z = -9 - 5x_2 - 6x_3 - 4x_4$$

s.t.

$$3x_2 + 5x_3 + 2x_4 \le 4$$
$$x_3 + x_4 \le 1$$
$$x_3 \le 1$$
$$-x_2 + x_4 \le 0$$
$$x_j = 0,1 \ \forall j = 2,3,4$$

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

Solution of (LP1): $(x_1, x_2, x_3, x_4) = (0,1,0,1);\ z = -9$

Solution of (LP2): $(x_1, x_2, x_3, x_4) = (1, \frac{4}{5}, 0, \frac{4}{5});\ z = -16\frac{1}{5}$

Lower bounds: $\underline{z}_{IP2} = -16$

(LP1) has integer solution ⇒ Update the current best objective value $z^* = -9$.

(IP1) is fathomed due to integrality. Hence, (IP2) will be considered for branching in the next iteration

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

- Iteration 2: Branching subproblem (IP2) based on x_2

(IP3): Min $Z = -9 - 6x_3 - 4x_4 \quad (x_2 = 0)$

s.t.

$$5x_3 + 2x_4 \le 4$$
$$x_3 + x_4 \le 1$$
$$x_3 \le 1$$
$$x_4 \le 0$$
$$x_j = 0,1 \ \forall j = 3,4$$

(IP4): Min $Z = -14 - 6x_3 - 4x_4 \quad (x_2 = 1)$

s.t.

$$5x_3 + 2x_4 \le 1$$
$$x_3 + x_4 \le 1$$
$$x_3 \le 1$$
$$x_4 \le 1$$
$$x_j = 0,1 \ \forall j = 3,4$$

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

Solution of (LP3): $(x_1, x_2, x_3, x_4) = (0,0,\frac{4}{5},0);\ z = -13\frac{4}{5}$

Solution of (LP4): $(x_1, x_2, x_3, x_4) = (1,1,0,\frac{1}{2});\ z = -16$

Lower bounds: $\underline{z}_{IP3} = -13$ and $\underline{z}_{IP4} = -16$

(IP4) will be considered for branching in the next iteration (smaller lower bound).

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

Iteration 3: Branching subproblem (IP4) based on x_3

(IP5): Min $Z = -14 - 4x_4$ $(x_3 = 0)$

s.t.

$$2x_4 \leq 1$$

$$x_4 \leq 1$$

$$x_4 = 0,1$$

(IP6): Min $Z = -20 - 4x_4$ $(x_3 = 1)$

s.t.

$$2x_4 \leq -4$$

$$x_4 \leq 0$$

$$x_4 = 0,1$$

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

Solution of (LP5): $(x_1, x_2, x_3, x_4) = (1,1,0,\frac{1}{2})$; $z = -16$

Solution of (LP6): Not exist! Infeasible

Lower bound: $\underline{z}_{IP5} \geq -16$

(IP6) is fathomed due to infeasibility. Hence, (IP5) will be considered for branching in the next iteration.

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
BRANCH AND BOUND ALGORITHM FOR 0-1 IP

- Iteration 4: Branching subproblem (IP5) based on x_4

(IP7) - $x_4 = 0$ and (IP8) - $x_4 = 1$ are not mathematical programs
(IP7) $\Rightarrow (x_1, x_2, x_3, x_4) = (1,1,0,0)$; $z = -14$: integer solution.
(IP8) $\Rightarrow$ $(x_1, x_2, x_3, x_4) = (1,1,0,1)$; infeasible.

Update the current best objective value $z^* = -14$.

With $z^* = -14$, subproblem (IP3) is also fathomed due to $\underline{z}_{IP3} \geq z^*$

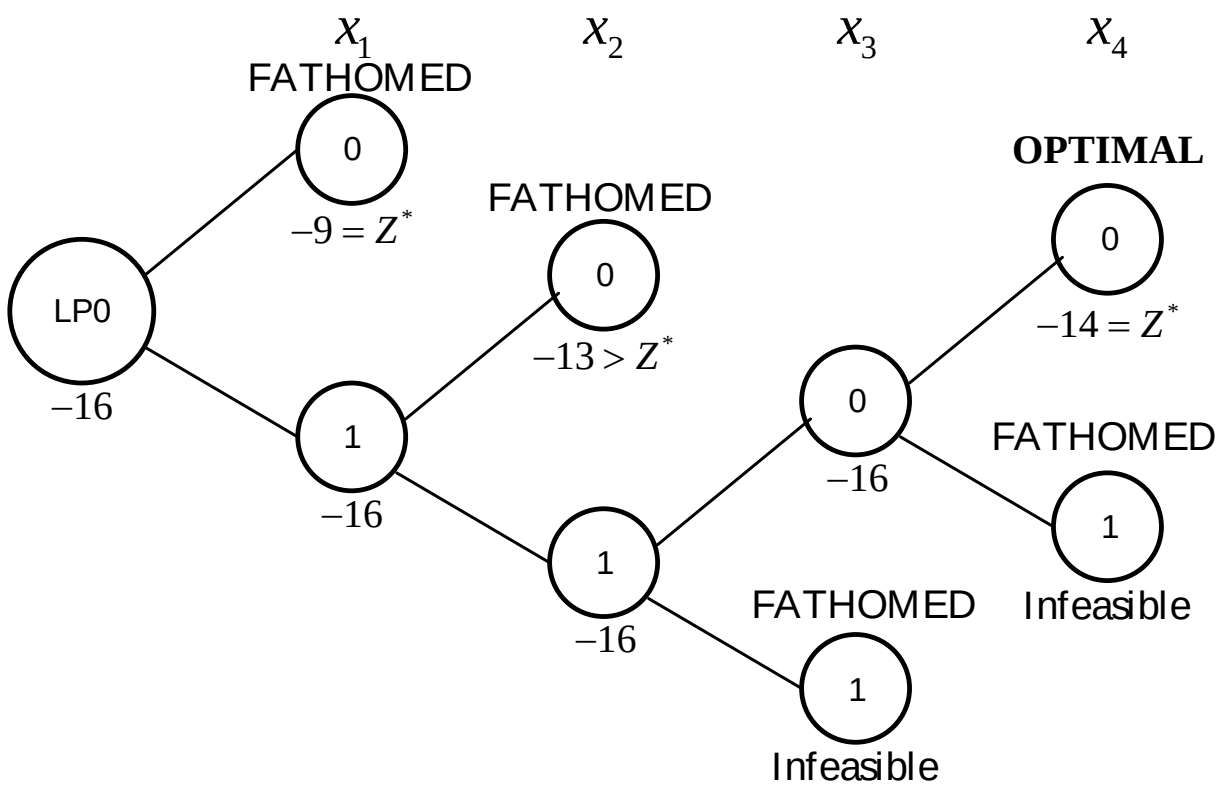
Solution of (IP0): $(x_1, x_2, x_3, x_4) = (1,1,0,0)$;$z^* = -14$

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach

BRANCH AND BOUND ALGORITHM FOR 0-1 IP

The solution procedure can be summarized in the following graph:



Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
IMPLICIT ENUMERATION METHOD FOR 0-1 IP

Consider the 0-1 IP program:

$$\begin{aligned} \text{Min} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i \qquad \forall i = 1,2,\dots,m \\ & x_j = 0,1 \qquad\qquad \forall j = 1,2,\dots,n \end{aligned}$$

Without loss of generality, we can assume that $c_j \geq 0$ (if $c_j < 0$ then define $x_j' = 1 - x_j \Rightarrow c_j x_j = c_j + c_j' \, x_j'$ with $c_j' = -c_j \geq 0$)

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
IMPLICIT ENUMERATION METHOD FOR 0-1 IP

Definitions:

- Fixed variable: the variable that has been assigned a value (either 0 or 1)
- Free variable: the variable that has not been assigned a value.

If the inequality constraints are ignored, the objective function is minimized by setting all free variables to 0 (since all $c_j \geq 0$).

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
IMPLICIT ENUMERATION METHOD FOR 0-1 IP

At any node of the branching tree, the *i*-th constraint is equivalent to:

$$\sum_{j \in FREE} a_{ij} x_j \geq b_i - \sum_{j \in FIXED} a_{ij} x_j = (\text{const})$$

$$\Rightarrow \quad \sum_{j \in FREE} a_{ij}^{+} x_j \geq b_i - \sum_{j \in FIXED} a_{ij} x_j \qquad (*)$$

where $a_{ij}^{+} = max\{0, a_{ij}\}$

Hence, if $\sum_{j \in FREE} a_{ij}^{+} x_j < b_i - \sum_{j \in FIXED} a_{ij} x_j$, the *i*-th constraint will never be satisfied regardless of any value assigned to the free variables ⇒ The current node is fathomed.

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
IMPLICIT ENUMERATION METHOD FOR 0-1 IP

Fathoming criteria:

1. The infeasibility test is positive, i.e.,

$$\sum_{j \in FREE} a_{ij}^{+} x_j < b_i - \sum_{j \in FIXED} a_{ij} x_j$$

1. Feasible integer solution is found
2. The objective value > upper bound $\bar{z}$

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
IMPLICIT ENUMERATION METHOD FOR 0-1 IP

Example 11: Consider

(IP0) Min $z = 8x_1 + 2x_2 + 4x_3 + 7x_4 + 5x_5$

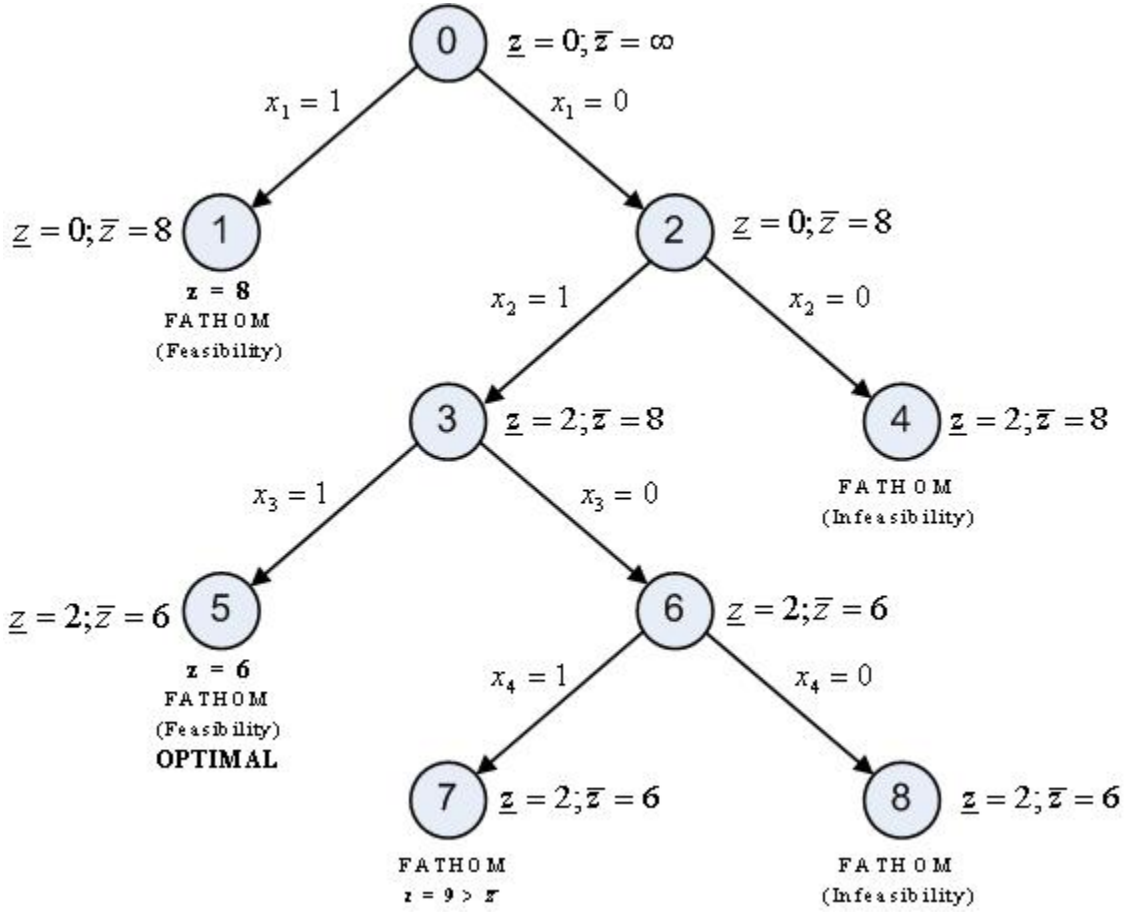
s.t.

$$3x_1 + 3x_2 - x_3 - 2x_4 - 3x_5 \geq 2$$
$$5x_1 + 3x_2 + 2x_3 + x_4 - x_5 \geq 4$$
$$x_j = 0,1 \quad \forall j = 1,2,3,4,5$$

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
IMPLICIT ENUMERATION METHOD FOR 0-1 IP

Solution Diagram:



Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
IMPLICIT ENUMERATION METHOD FOR 0-1 IP

Node 0: No fixed variables
If we set all variables equal zero then $z = 0$ but the constraints are not satisfied. Hence, $\underline{z} = 0$ and $\overline{z} = \infty$

Node 1: $x_1 = 1$
Minimum value = 8 by setting $x_2 = x_3 = x_4 = x_5 = 0$. Both constraints are satisfied ⇒ Node 1 is fathomed. Update $\overline{z} = 8$.

Node 2: $x_1 = 0$
Minimum value = 0 by setting $x_2 = x_3 = x_4 = x_5 = 0$. Both constraints are not satisfied.
Checking (*): $3 \geq 2$ & $6 \geq 4$ - Both constraints satisfy (*) ⇒ Further branching

Node 3: $x_1 = 0$, $x_2 = 1$
Minimum value = 2 by setting $x_3 = x_4 = x_5 = 0$. Constraint 2 is not satisfied. Update $\underline{z} = 2$.
Checking (*): $3 \geq 1$ - Constraint 2 satisfies (*) ⇒ Further branching

Co-funded by the
Erasmus+ Programme
of the European Union

Solution Approach
IMPLICIT ENUMERATION METHOD FOR 0-1 IP

Node 4: $x_1 = 0$, $x_2 = 0$
Minimum value = 0 by setting $x_3 = x_4 = x_5 = 0$. Both constraints are not satisfied.
Checking (*): $0 \geq 2$ & $3 \geq 4$ - Both constraints do not satisfy (*) ⇒ Node 4 is fathomed. (In fact, if any constraint does not satisfy (*) then the node is fathomed)

Node 5: $x_1 = 0$, $x_2 = 1$, $x_3 = 1$
Minimum value = 6 by setting $x_4 = x_5 = 0$. Both constraints are satisfied ⇒ Node 5 is fathomed. Update $\bar{z} = 6$.

Node 6: $x_1 = 0$, $x_2 = 1$, $x_3 = 0$
Minimum value = 2 by setting $x_4 = x_5 = 0$. Constraint 2 is not satisfied.
Checking (*): $1 \geq 1$ - Constraint 2 satisfies (*) ⇒ Further branching..

Co-funded by the Erasmus+ Programme of the European Union

Solution Approach
IMPLICIT ENUMERATION METHOD FOR 0-1 IP

Node 7: $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$

Minimum value = 9 by setting $x_5 = 0$. Node 7 is fathomed because $z = 9 > \overline{z} = 6$.

Node 8: $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$

Minimum value = 2 by setting $x_5 = 0$. Constraint 2 is not satisfied.

Checking (*): $1 \geq 1$ - Constraint 2 does not satisfy (*) ⇒ Node 8 is fathomed.

Optimal solution: $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 0$, $x_5 = 0$ and $z^* = 6$

Co-funded by the Erasmus+ Programme of the European Union

Combinatorial Optimization

- Combinatorial analysis is the study of the arrangement, grouping, or selection of discrete objects, usually finite in number.

- In Operations Research, combinatorial optimization aims at searching for the best element among a finite set of elements

- Integer programming together with combinatorial optimization are considered as “discrete optimization”

Co-funded by the
Erasmus+ Programme
of the European Union

Combinatorial Optimization

Common problems involving combinatorial optimization:

- Knapsack Problem
- Travelling Salesman Problem
- Assignment Problem
- Transportation Problem
- Vehicle Routing Problem
- Shortage Path Problem
- Maximum Flow Problem
- Minimum Spanning Tree Problem
- Set Covering Problem
- Cutting Stock Problem
- etc

Co-funded by the
Erasmus+ Programme
of the European Union

Combinatorial Optimization

- In this section, solution techniques for a few network flow problems will be discussed. It should be noted that these techniques are efficient only for problems with small dimensionality

- Most of the combinatorial problems are difficult to solve. In many practical applications, metaheuristic algorithms (e.g., GA, ACO, SA, PSO,…) should be used to help find good solution for combinatorial optimization problems

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem

Find the least-cost transportation plan from a set of supply nodes with supply capacities s_i $(i = 1,2,\dots,m)$ to a set of demand nodes with required demand quantities d_j $(j = 1,2,\dots,n)$

Minimize $Z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$

s.t. $\sum_{j=1}^{n} x_{ij} \le s_i \quad (i = 1,2,\dots,m)$

$\sum_{i=1}^{m} x_{ij} \ge d_j \quad (j = 1,2,\dots,n)$

$x_{ij} \ge 0 \quad \forall i,j$

Where: x_{ij} - the amount to be transported from supply node *i* to demand node *j*

c_{ij} - unit transportation cost form supply node *i* to demand node *j*

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem

Feasibility condition:

$$\sum_{i=1}^{m} s_i \geq \sum_{j=1}^{n} d_j$$

The balanced transportation model:

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$$

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem

- If the demand exceeds the supply, a dummy supply node with a capacity $\sum_{j=1}^{n} d_j - \sum_{i=1}^{m} s_i$ is added to balance the transportation model. The unit transportation cost from the dummy supply node to any demand node is zero
- If the supply exceeds the demand, a dummy demand node with a required demand quantity $\sum_{i=1}^{m} s_i - \sum_{j=1}^{n} d_j$ is added to balance the transportation model. The unit transportation cost from any supply node to the dummy demand node is zero.
- The balanced transportation problem always has optimal solution.

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem

Example 12:

Supply	Demand			
	1	**2**	**3**	**4**
1	5	7	9	6
2	6	7	10	5
3	7	6	8	1

Minimize $5x_{11}+7x_{12}+9x_{13}+6x_{14}+6x_{21}+7x_{22}+10x_{23}+5x_{24}+7x_{31}+6x_{32}+8x_{33}+x_{34}$

s.t.

$$x_{11}+x_{12}+x_{13}+x_{14}=120$$
$$x_{21}+x_{22}+x_{23}+x_{24}=140$$
$$x_{31}+x_{32}+x_{33}+x_{34}=100$$
$$x_{11}+x_{21}+x_{31}=100$$
$$x_{12}+x_{22}+x_{32}=60$$
$$x_{13}+x_{23}+x_{33}=80$$
$$x_{14}+x_{24}+x_{34}=120$$
$$x_{ij}\geq 0 \qquad i=1,2,3;\ j=1,2,3,4$$

Note that there exists a redundant constraint!

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem

Transportation tableau:

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Demand Node 4	Supply
1	5 x_{11}	7 x_{12}	9 x_{13}	6 x_{14}	120
2	6 x_{21}	7 x_{22}	10 x_{23}	5 x_{24}	140
3	7 x_{31}	6 x_{32}	8 x_{33}	1 x_{34}	100
Demand	100	60	80	120	360

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Transportation Algorithm

The transportation algorithm follows the exact steps of the simplex method:

Step 1: Determine a starting basic feasible solution and go to step 2

Step 2: Use the optimality condition to determine the entering variable from among all nonbasic variables. If the optimality condition is satisfied, stop. Otherwise, go to step 3

Step 3: Use the feasibility condition to determine the leaving variable from among all basic variables and find the new basic variable. Return to step 2

However, with the specific structure of the transportation model, the computations will be organized in a more convenient form

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem
Determine the Starting Solution

THE NORTHWEST-CORNER METHOD

Start at the northwest-corner cell (route) of the tableau

1. Allocate as much as possible from a supply node (a row) to the demand nodes with the priority order from *left to right* until the supply capacity of the current supply node is exhausted. Start with a new supply node (a new row)

1. Allocate as much as possible to a demand node (a column) from the supply nodes with the priority order from *top to bottom* until the demand of the current demand node is satisfied. Start with a new demand node (a new column)

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem
Determine the Starting Solution

Example 13: Consider the problem in Ex.12

Supply Node	Demand Node 1	2	3	4	Supply
1	5 → 100	7 → 20	9	6	120
2	6	7 → 40	10 → 80	5 → 20	140
3	7	6	8	1 → 100	100
Demand	100	60	80	120	360

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Determine the Starting Solution

THE LEAST-COST METHOD

1. Assigning as much as possible to the cell with the smallest unit transportation cost (break ties arbitrarily).
2. Discard the satisfied row or column and adjust the amounts of supply and demand. Redetermine the cell with smallest unit transportation cost.
3. Repeat until all demand nodes are satisfied (all supply nodes are exhausted)

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem
Determine the Starting Solution

Example 14: Consider the problem in Ex.12

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Demand Node 4	Supply
1	**5** 100	**7**	**9** 20	**6**	**120**
2	**6**	**7** 60	**10** 60	**5** 20	**140**
3	**7**	**6**	**8**	**1** 100	**100**
Demand	**100**	**60**	**80**	**120**	**360**

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Determine the Starting Solution

Note: We have a better starting solution

$$C_{\text{Northwest}} = 100*5 + 20*7 + 40*7 + 80*10 + 20*5 + 100*1 = 1920$$

$$C_{\text{Least-cost}} = 100*5 + 20*9 + 60*7 + 60*10 + 20*5 + 100*1 = 1900$$

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem
Determine the Starting Solution

THE VOGEL APPROXIMATION METHOD (VAM)

VAM is an improved version of the least-cost method in order to produce a better starting solution

1. For each row/column, determine a *penalty measure* by subtracting the *smallest* unit cost element in the row/column from the *next smallest* unit cost element in the same row/column
2. Identify the row or column with the largest penalty (break ties arbitrarily or select the row/column such that the allocated amount can be maximized). Allocate as much as possible to the cell with the least unit transportation cost in the selected row or column
3. Discard the satisfied row or column.
4. Go to step 1

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem
Determine the Starting Solution

Example 15: Consider the problem in Ex.12

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Demand Node 4	Supply	
1	5	7	9	6	120	1
2	6	7	10	5	140	1
3	7 X	6 X	8 X	1 100	100	5 *
Demand	100	60	80	120	360	
	1	1	1	4		

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Determine the Starting Solution

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Demand Node 4	Supply	
1	**5** 100	**7**	**9**	**6**	**120**	1 *
2	**6** X	**7**	**10**	**5**	**140**	1
3	**7** X	**6** X	**8** X	**1** 100	**100**	
Demand	**100**	**60**	**80**	**120**	**360**	
	1*	0	1	1		

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Determine the Starting Solution

Supply Node	Demand Node 1	2	3	4	Supply	
1	**5** 100	**7**	**9**	**6** X	**120**	1
2	**6** X	**7**	**10**	**5** 20	**140**	2 *
3	**7** X	**6** X	**8** X	**1** 100	**100**	
Demand	**100**	**60**	**80**	**120**	**360**	
		0	1	1		

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Determine the Starting Solution

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Demand Node 4	Supply	
1	**5** 100	**7** X	**9**	**6** X	**120**	2
2	**6** X	**7** 60	**10**	**5** 20	**140**	3 *
3	**7** X	**6** X	**8** X	**1** 100	**100**	
Demand	**100**	**60**	**80**	**120**	**360**	
		0	1			

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Determine the Starting Solution

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Demand Node 4	Supply
1	**5** 100	**7** X	**9** 20	**6** X	**120** #
2	**6** X	**7** 60	**10** 60	**5** 20	**140** #
3	**7** X	**6** X	**8** X	**1** 100	**100**
Demand	**100**	**60**	**80**	**120**	**360**

#

Total transportation cost of this starting solution:

$$C_{\text{VAM}} = 100*5 + 20*9 + 60*7 + 60*10 + 20*5 + 100*1 = 1900$$

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

In this section, the methods to find optimal solution for *non-degenerate* transportation problems will be discussed. It is noted that a transportation problem is non-degenerate if there are exactly $(m + n - 1)$ cells assigned *positive* values in any iteration.

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem
Finding Optimal Solution

The Stepping-Stone Method

Find the starting solution. At each iteration, perform the following steps:

1. Compute the *improvement indices* I_{ij} for all empty cells (i, j) in the tableau:

a. For each empty cell (i, j), draw a closed path which is comprised of horizontal and vertical segments connecting that cell with other non-empty cells.
b. Alternatively assign the *plus/minus* signs for the vertices of the path, starting with *plus* sign at the current cell.
c. The improvement index I_{ij} of cell (i, j) is defined as the algebraic sum of unit transportation costs on the path.

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem
Finding Optimal Solution

2. If all improvement indices are nonnegative, the current solution is optimal. Otherwise, the empty cell with the most negative value of I_{ij} will be selected for improvement by reallocating the assigned value for all cells on the path as follows:

- Determine the minimum assigned value among all cells with *minus* sign: x_{ij}^{min}
- The value of the cells with minus sign will be subtracted by an amount x_{ij}^{min}.
- The value of the cells with plus sign will be added an amount x_{ij}^{min}.

3. Construct the new transportation tableau and return to step 1.

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem
Finding Optimal Solution

Notes:

- There exists a unique path for each empty cell.
- Compared to the simplex method, the cells assigned positive values are basic variable while the empty cells are nonbasic variable. The improvement index I_{ij} associated with the empty cell (i, j) is the reduced cost associated with that cell.
- After each iteration, there will be one nonempty cell become empty. There is no need to investigate this new empty cell in the next iteration.

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem
Finding Optimal Solution

Example 16:

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Demand Node 4	Supply
1	**5** 100	**7** 20	**9**	**6**	**120**
2	**6**	**7** 40	**10** 80	**5** 20	**140**
3	**7**	**6**	**8**	**1** 100	**100**
Demand	**100**	**60**	**80**	**120**	**360**

Empty cells: (1,3), (1,4), (2,1), (3,1), (3,2), (3,3)

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

Iteration 1: Consider cell (1,3): (Why?)

Supply Node	Demand Node 1	2	3	4	Supply
1	5 100	- 7 20	+ 9	6	120
2	6	7 40 +	10 80 -	5 20	140
3	7	6	8	1 100	100
Demand	100	60	80	120	360

$I_{13} = +9 - 10 + 7 - 7 = -1$: Not optimal; $x_{min} = min(20,80) = 20$ (Note that: $I_{14} = 1, I_{21} = 1, I_{31} = 6, I_{32} = 3, I_{33} = 2$). Hence, $x_{12} = 0, x_{13} = 20, x_{22} = 60, x_{23} = 60$

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

Iteration 2:

Supply Node	Demand Node 1	2	3	4	Supply
1	**5** 100	**7**	**-** **9** 20	+ **6**	**120**
2	**6**	**7** 60	**10** 60 +	**5** 20 -	**140**
3	**7**	**6**	**8**	**1** 100	**100**
Demand	**100**	**60**	**80**	**120**	**360**

Cell (1,4): $I_{14} = +6 - 5 + 10 - 9 = +2$ ⇒ Cell (1,4) satisfies the optimality condition

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

Supply Node	Demand Node 1	2	3	4	Supply
1	- 5 100	7	+ 9 20	6	120
2	6 +	7 60	10 60 -	5 20	140
3	7	6	8	1 100	100
Demand	100	60	80	120	360

Cell (2,1): $I_{21} = +6 - 5 + 9 - 10 = 0$ ⇒ Cell (2,1) satisfies the optimality condition

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

Supply Node	Demand Node 1	2	3	4	Supply
1	- 5 100	7	+ 9 20	6	120
2	6	7 60	- 10 60	+ 5 20	140
3	7 +	6	8	1 100 -	100
Demand	100	60	80	120	360

Cell (3,1): $I_{31} = +7 - 5 + 9 - 10 + 5 - 1 = 5 \Rightarrow$ Cell (3,1) satisfies the optimality condition

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Demand Node 4	Supply
1	5 100	7	9 20	6	120
2	6	7 - 60	10 60	5 + 20	140
3	7	6 +	8	1 100 -	100
Demand	100	60	80	120	360

Cell (3,2): $I_{32} = +6 - 7 + 5 - 1 = 3$ ⇒ Cell (3,2) satisfies the optimality condition

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Demand Node 4	Supply
1	5 100	7	9 20	6	120
2	6	7 60	- 10 60	+ 5 20	140
3	7	6	8 +	1 100 -	100
Demand	100	60	80	120	360

Cell (3,3): $I_{33} = +8 - 10 + 5 - 1 = 2$ ⇒ Cell (3,3) satisfies the optimality condition

We have optimal solution!

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

The Modified Distribution Method (MODI) - The Method of Multipliers

In this method, the improvement indices are computed based on the dual variables of the transportation problem. The principle of this method is as follows:

Denote $u_i\ (i = 1,2,\dots,m)$, $v_j\ (j = 1,2,\dots,n)$: dual variables associated with supply node *i* and demand node *j* . We have:

1. $I_{ij} = c_{ij} - (u_i + v_j) = 0$ at nonempty cells.
2. $I_{ij} = c_{ij} - (u_i + v_j) \neq 0$ at empty cells.

From the set of equations developed from the values of nonempty cells, we can determine all values of $u_i\ (i = 1,2,\dots,m)$ and $v_j\ (j = 1,2,\dots,n)$

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

Example 17:

Supply Node	Demand Node				Supply
	1 (v_1)	**2** (v_2)	**3** (v_3)	**4** (v_4)	
1 (u_1)	**10** 5	**0** 10	**20**	**11**	**15**
2 (u_2)	**12**	**7** 25	**9** 15	**20** 35	**75**
3 (u_3)	**0**	**14**	**16**	**18** 20	**20**
Demand	**5**	**35**	**15**	**55**	**110**

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

$$u_1 + v_1 = 10, u_1 + v_2 = 0$$
$$u_2 + v_2 = 7, u_2 + v_3 = 9$$
$$u_2 + v_4 = 20, u_3 + v_4 = 18$$

select $u_1=0$ $\Rightarrow$

$$u_1 = 0, u_2 = 7, u_3 = 5$$
$$v_1 = 10, v_2 = 0, v_3 = 2, v_4 = 13$$

$$\Rightarrow I_{13} = 18, I_{14} = -2, I_{21} = -5, I_{31} = -15, I_{32} = 9, I_{33} = 9 \ (Z = 1420)$$

$\Rightarrow$ The first iteration: Select cell (3,1)

Co-funded by the
Erasmus+ Programme
of the European Union

Transportation Problem
Finding Optimal Solution

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Demand Node 4	Supply
1	**10**	**0** 15	**20**	**11**	**15**
2	**12**	**7** 20	**9** 15	**20** 40	**75**
3	**0** 5	**14**	**16**	**18** 15	**20**
Demand	**5**	**35**	**15**	**55**	**110**

This tableau is optimal!

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

Degeneracy

1. If degeneracy occurs at starting tableau ⇒ Assign 0 to an empty cell and deal with this cell as a nonempty one.

2. If degeneracy occurs at an iteration (when there are two nonempty cells associated with the minimum value on the path) ⇒ Assign to one of them and deal with this cell as a nonempty one (usually select the cell with smaller unit transportation cost).

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

Example 18:

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Supply
1	8 70	5	16	70
2	15 80	10 50	7	130
3	3	9 30	10 50	80
Demand	150	80	50	280

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

After iteration 1 (pivot at cell (3,1)):

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Supply
1	**8** 70	**5**	**16**	**70**
2	**15** 50	**10** 80	**7**	**130**
3	**3** 30	**9**	**10** 50	**80**
Demand	**150**	**80**	**50**	**280**

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

After iteration 2 (pivot at cell (2,3)):

Supply Node	Demand Node 1	Demand Node 2	Demand Node 3	Supply
1	**8** 70	**5**	**16**	**70**
2	**15** ***	**10** 80	**7** 50	**130**
3	**3** 80	**9**	**10** ***	**80**
Demand	**150**	**80**	**50**	**280**

Assign 0 to cell (3,3)!

Co-funded by the Erasmus+ Programme of the European Union

Transportation Problem
Finding Optimal Solution

Multiple Solutions

The transportation problem will has multiple solutions if the improvement index of an empty cell equals to 0 at the optimal tableau

Reason: Pivoting at that empty cell will result in a new solution with the same (optimal) objective function.

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment Problem

- A specific type of LP that aims at assigning a set of *n* jobs to a set of *n* machines/workers/employees.

- Objective: minimize the total cost or total time

Assignment problem can be considered as a balanced transportation problem in which all supplies and demands are equal to 1. However, the methods developed for balanced transportations problems are very inefficient in this case because the problem is highly degenerate.

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment Problem

Denote c_{ij}: cost of assigning job *j* to machine *i*

Decision variables

$$x_{ij} = \begin{cases} 1 & \text{if job } j \text{ is assigned to machine } i \\ 0 & \text{otherwise} \end{cases}$$

Problem:

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

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i$$

$$x_{ij} = 0 \text{ or } 1 \qquad \forall i, j$$

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment Problem
The Hungarian Method

Determine the cost matrix of the assignment problem

Step 1: Determine the reduced cost matrix of the problem

- Find the minimum element in each row of the cost matrix; subtract this minimum cost from each cost in the same row of the cost matrix
- Find the minimum element in each column of the new matrix; subtract this minimum cost from each cost in the same column of the matrix

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment Problem
The Hungarian Method

Step 2:

- Draw the minimum number of lines (horizontal or vertical) that are needed to cover all the zeros in the reduced cost matrix. If n lines are drawn, stop: the optimal solution is found and determined from the covered zeros. Otherwise, go to step 3.

Note: The optimal solution is defined from a set of independent zeros in the final matrix, which is the set of zeros such that there is no horizontal or vertical line that covers more than one of them.

Step 3:

- Find the smallest uncovered element in the reduced cost matrix. Subtract this value from each uncovered element and add this value to each element that is covered by two lines. Return to step 2

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment Problem
The Hungarian Method

Example 19:

M/C	Job				
	1	**2**	**3**	**4**	**5**
1	2	3	5	1	4
2	-1	1	3	6	2
3	-2	4	3	5	0
4	1	3	4	1	4
5	7	1	2	1	2

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment Problem
The Hungarian Method

Reduced cost matrix:

M/C	Job					$\bar{u}_i$ = Min c_{ij}
	1	**2**	**3**	**4**	**5**	
1	2	3	5	1	4	1
2	-1	1	3	6	2	-1
3	-2	4	3	5	0	-2
4	1	3	4	1	4	1
5	7	1	2	1	2	1

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment Problem
The Hungarian Method

$\Rightarrow$

M/C	Job				
	1	**2**	**3**	**4**	**5**
1	1	2	4	0	3
2	0	2	4	7	3
3	0	6	5	7	2
4	0	2	3	0	3
5	6	0	1	0	1
$\bar{u}_j = \mathbf{Min}\ \{c_{ij} - \bar{u}_i\}$	0	0	1	0	1

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment Problem
The Hungarian Method

$\Rightarrow$

M/C	Job				
	1	**2**	**3**	**4**	**5**
1	1	2	3	0	2
2	0	2	3	7	2
3	0	6	4	7	**1**◯
4	0	2	2	0	2
5	~~6~~	~~0~~	~~0~~	~~0~~	~~0~~

(Minimum number of lines: 3 < 5)

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment Problem
The Hungarian Method

M/C	Job				
	1	2	3	4	5
1	1	1	2	0	1
2	0	1	2	7	1
3	0	5	3	7	0
4		1	1	0	1
5	7	0	0	1	0

(Minimum number of lines: 4 < 5)

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment Problem
The Hungarian Method

M/C	Job				
	1	2	3	4	5
1	~~1~~	~~0~~	~~1~~	~~0~~	~~0~~
2	~~0~~	~~0~~	~~1~~	~~7~~	~~0~~
3	1	5	3	8	0
4	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~
5	~~8~~	~~0~~	~~0~~	~~2~~	~~0~~

Optimal Solution

Co-funded by the
Erasmus+ Programme
of the European Union

Assignment Problem
The Hungarian Method

There are multiple solutions:

1. $x_{12} = x_{21} = x_{35} = x_{44} = x_{53} = 1$
2. $x_{14} = x_{22} = x_{35} = x_{41} = x_{53} = 1$
3. $x_{14} = x_{21} = x_{35} = x_{42} = x_{53} = 1$
4. $x_{14} = x_{21} = x_{35} = x_{43} = x_{52} = 1$

Optimal objective value = 5

Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem

Consider the following network's problem: Transport the maximum amount of flow from a starting point (the source node) to a terminal point (the sink node)

The network with directed arcs will be considered first and expansion to networks with undirected arcs will then be investigated.

Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Directed Arcs

Consider the network with m nodes and n arcs. The capacity of arc (i,j) is c_{ij} $(i,j = 1,2,..,m)$

Decision variables: x_{ij} - flow through arc (i,j)
Denote f : total flow from source node 1 to sink node m

The problem can be formulated as an LP:

$$\text{Max} \quad f$$

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = \begin{cases} f & \text{if} \quad i = 1 \\ 0 & \text{if} \quad i = 2,3,\dots,m-1 \\ -f & \text{if} \quad i = m \end{cases}$$

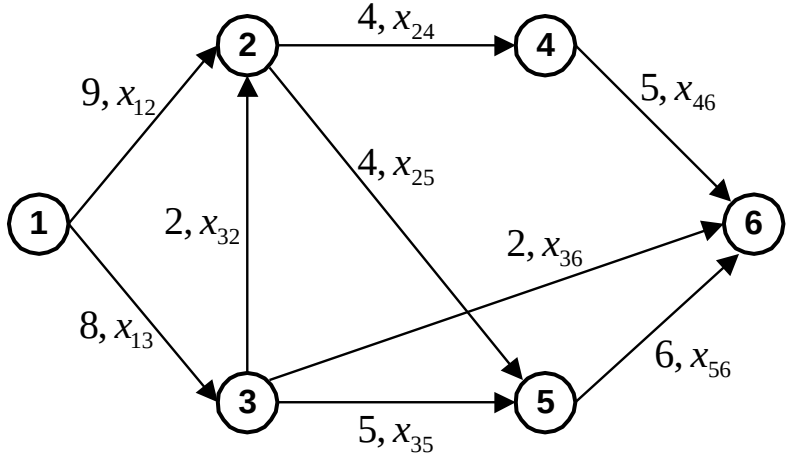
$$x_{ij} \le c_{ij} \qquad \forall (i,j)$$

$$x_{ij} \ge 0 \qquad \forall (i,j)$$

Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Directed Arcs

Example 20:



Max f

s.t.

$$x_{12} + x_{13} = f \qquad x_{12} \le 9; x_{13} \le 8; x_{32} \le 2$$

$$x_{24} + x_{25} - x_{12} - x_{32} = 0 \qquad x_{24} \le 4; x_{25} \le 4; x_{35} \le 5$$

$$x_{32} + x_{35} + x_{36} - x_{13} = 0 \qquad x_{36} \le 2; x_{46} \le 5; x_{56} \le 6$$

$$x_{46} - x_{24} = 0, \; x_{56} - x_{25} - x_{35} = 0$$

$$-x_{36} - x_{46} - x_{56} = -f$$

$$x_{12}, x_{13}, x_{32}, x_{24}, x_{25}, x_{35}, x_{36}, x_{46}, x_{56} \ge 0$$

Co-funded by the Erasmus+ Programme of the European Union

Maximum Flow Problem
Network with Directed Arcs

Ford-Fulkerson's Labeling Algorithm

Can be applied when c_{ij}'s are integers or rational numbers

General Principle: An unlabeled node *j* will be labeled based on a labeled node *i* which is connected to *j* by arc (i, j). The label of node *j* will have the form $[i\pm, v_j]$, in which:

- v_j : the amount of flow changed between *i* and *j*.
- $i+$: if the flow from *i* to *j* increases by an amount v_j.
- $i-$: if the flow from i to j decreases by an amount v_j.

Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Directed Arcs

Step 1:

1. Find a starting feasible solution for the problem (for instance, $x_{ij} = 0 \ \forall(i,j)$).
2. Assign label $[-,\infty]$ to node 1 (the source node): this means that the source node has unlimited supply capacity.

Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Directed Arcs

Step 2: Successively assign label for unlabeled nodes as follows

1. Select a labeled node – At the beginning, only node 1 is labeled node.
2. Consider an unlabeled node *j* that is directly connected to *i*:

a. If $x_{ij} \leq c_{ij}$: assign the label $[i+, v_j]$ to node *j*, in which $v_j = Min\{v_i, c_{ij} - x_{ij}\}$. Meaning: the maximum increase of amount of flow from *i* to *j* is the smaller value between the amount of flow that can be transported from *i* and the remaining capacity of arc (i, j).

b. If $x_{ji} > 0$: assign the label $[i-, v_j]$ to node *j*, in which $v_j = Min\{v_i, x_{ji}\}$. Meaning: take back to node *j* a portion of the amount of flow that has been transported to *i*.

This step is repeated until:

- Node *m* has been labeled ⇒ A ***breakthrough path*** exists between the source node and the sink node. Then, go to step 3. Or,
- Cannot find a breakthrough path. Then, go to step 4.

Co-funded by the Erasmus+ Programme of the European Union

Maximum Flow Problem
Network with Directed Arcs

Step 3:

The breakthrough path defined in step 3 is a *flow augmenting path*. The labeling procedure has assigned a label $[i+, v_m]$ to node m. Revise the starting feasible solution to get a better feasible solution as follows:

- $x'_{ij} = x_{ij} + v_m$ if (i, j) is on the flow augmenting path and node j has the label $[i+, v_j]$
- $x'_{ji} = x_{ji} - v_m$ if (i, j) is on the flow augmenting path and node j has the label $[i-, v_j]$

Remove all labels and return to step 2.

Co-funded by the Erasmus+ Programme of the European Union

Maximum Flow Problem
Network with Directed Arcs

Step 4:

Optimal solution is achieved. The maximum flow can be determined from the sum of all labels v_m's received in step 3 (only if the starting solution is $x_{ij} = 0 \ \forall(i,j)$).

Notes:

1. The purpose of step 2 is to find a breakthrough path that connects the source node and the sink node. Therefore, it is not necessary to determine the labels of all nodes. The target should be finding a path such that the augmented flow is as large as possible

⇒ Rule of thumb: starting from a labeled node, select an unlabeled node *j* such that the assigned label v_j to that node is largest.

Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Directed Arcs

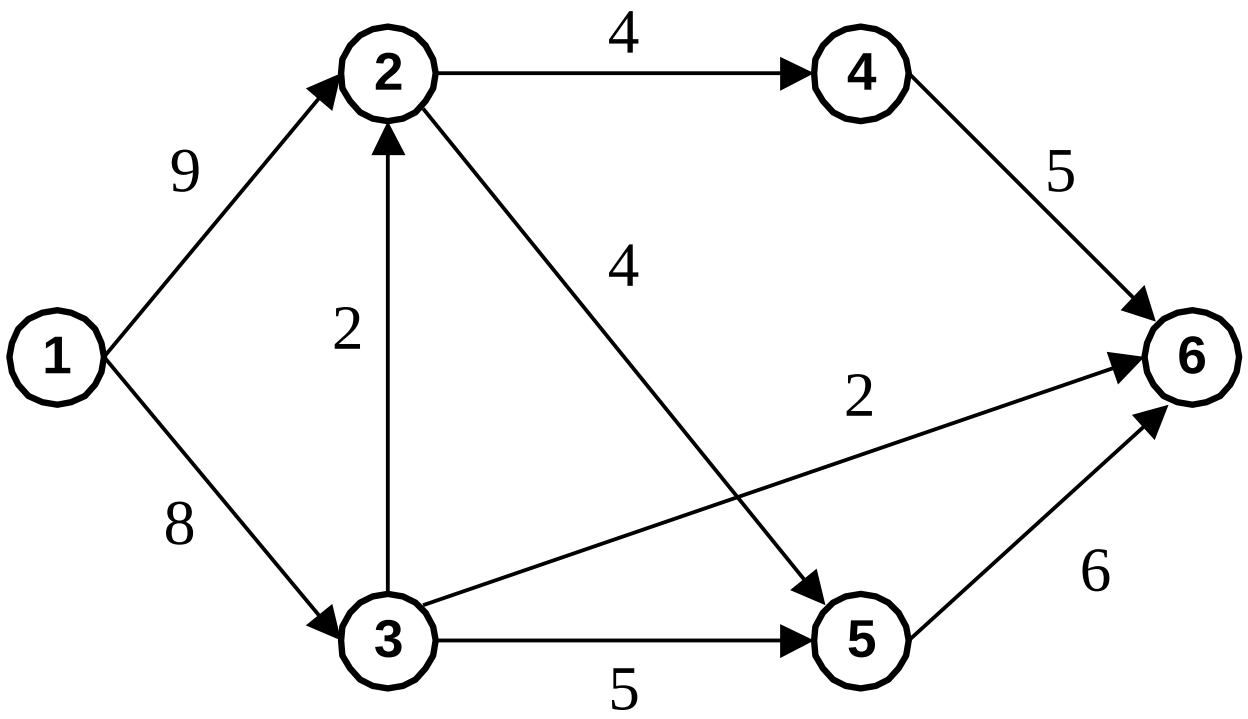
2. In each iteration (step 2), each node has a unique label.

3. Other ways to determine maximum flow
   a. Total algebraic flows at the source node.
   b. Total algebraic flows at the sink node.
   c. When the breakthrough path does not exist (the last iteration), the network's nodes can be divided into two sets: Set X includes the source node and set $\overline{X}$ includes the sink node. The arcs that connect X and $\overline{X}$ is called a *cut set*. Total transport capacity of the cut set $c(X,\overline{X})$ is the maximum amount of flow that can be transported from the source node to the sink node.

Co-funded by the Erasmus+ Programme of the European Union

Maximum Flow Problem
Network with Directed Arcs

Example 21:



Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Directed Arcs

Iteration 1: starting feasible solution: zeros for all arcs

Node 1: $[-, \infty]$ Node 2: $[1+, 9]$ Node 4: $[2+, 4]$ Node 6: $[4+, 4]$

Breakthrough path: 1-2-4-6.
New feasible solution: $x_{12} = x_{24} = x_{46} = 4$, other variables = 0

Iteration 2:

Node 1: $[-, \infty]$ Node 3: $[1+, 8]$ Node 5: $[3+, 5]$ Node 6: $[5+, 5]$

Breakthrough path: 1-3-5-6.
New feasible solution: $x_{12} = x_{24} = x_{46} = 4$, $x_{13} = x_{35} = x_{56}$ =5, other variables = 0

Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Directed Arcs

Iteration 3:

Node 1: $[-, \infty]$ Node 2: $[1+, 5]$ Node 5: $[2+, 4]$ Node 6: $[5+, 1]$

Breakthrough path: 1-2-5-6.

New feasible solution: $x_{12} = 5$, $x_{24} = x_{46} = 4$, $x_{13} = x_{35} = 5$, $x_{56} = 6$, $x_{25} = 1$, other variables = 0

Iteration 4:

Node 1: $[-, \infty]$ Node 3: $[1+, 3]$ Node 6: $[3+, 2]$

Breakthrough path: 1-3-6.

New feasible solution: $x_{12} = 5$, $x_{24} = x_{46} = 4$, $x_{13} = 7$, $x_{35} = 5$, $x_{56} = 6$, $x_{36} = 2$, $x_{25} = 1$, $x_{32} = 0$

Co-funded by the Erasmus+ Programme of the European Union

Maximum Flow Problem
Network with Directed Arcs

Iteration 5:

Node 1: $[-, \infty]$ Node 2: $[1+, 4]$ Node 5: $[2+, 3]$ Node 3: $[1+, 1]$

The breakthrough path does not exist: $X = \{1,2,3,5\}$ $\overline{X} = \{4,6\}$

The current solution is optimal. The maximum flow from node 1 to node 6 can be determined as follows:

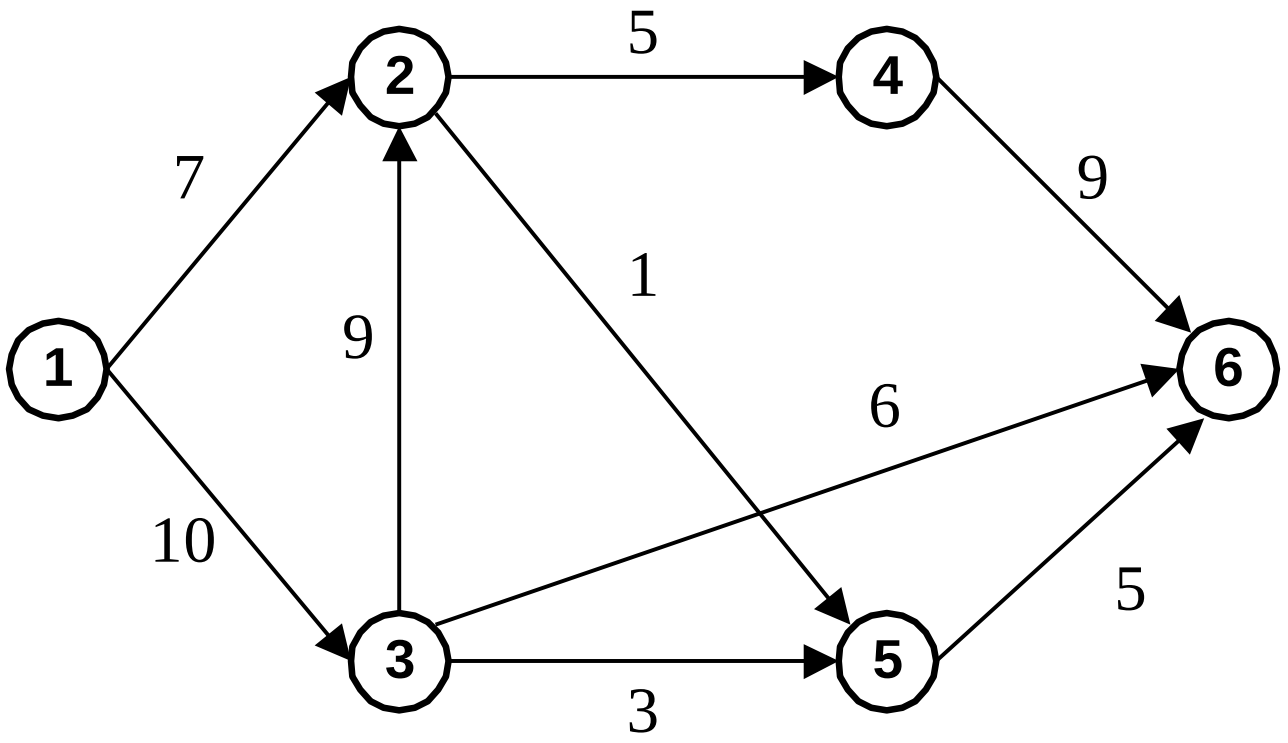
$\sum v_m = 4 + 5 + 1 + 2 = 12$ or $x_{46} + x_{36} + x_{56} = 4 + 2 + 6 = 12$

or $x_{12} + x_{13} = 5 + 7 = 12$ or $c(X, \overline{X}) = c_{24} + c_{36} + c_{56} = 4 + 2 + 6 = 12$

Co-funded by the Erasmus+ Programme of the European Union

Maximum Flow Problem
Network with Directed Arcs

Example 22:



Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Directed Arcs

Iteration 1: starting feasible solution: zeros for all arcs

Node 1: $[-, \infty]$ Node 3: $[1+, 10]$ Node 2: $[3+, 9]$ Node 4: $[2+, 5]$ Node 6: $[4+, 5]$

Breakthrough path: 1-3-2-4-6.
New feasible solution: $x_{13} = x_{32} = x_{24} = x_{46} = 5$, other variables = 0

Iteration 2:

Node 1: $[-, \infty]$ Node 2: $[1+, 7]$ Node 5: $[2+, 1]$ Node 6: $[5+, 1]$

Breakthrough path: 1-2-5-6.
New feasible solution: $x_{13} = x_{32} = x_{24} = x_{46} = 5$, $x_{12} = x_{25} = x_{56} = 1$, other variables = 0

Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Directed Arcs

Iteration 3:

Node 1: $[-, \infty]$ Node 3: $[1+, 5]$ Node 6: $[3+, 5]$

Breakthrough path: 1-3-6.

New feasible solution: $x_{13} = 10,\ x_{32} = x_{24} = x_{46} = 5,\ x_{12} = x_{25} = x_{56} = 1,\ x_{36} = 5$, other variables = 0.

Iteration 4:

Node 1: $[-, \infty]$ Node 2: $[1+, 6]$ Node 3*: $[2-, 5]$ Node 5: $[3+, 3]$ Node 6: $[5+, 3]$

Breakthrough path: 1-2-3-5-6.

New feasible solution: $x_{13} = 10,\ x_{32} = 2,\ x_{24} = x_{46} = 5,\ x_{12} = 4,\ x_{25} = 1,\ x_{56} = 4\ x_{36} = 5,\ x_{35} = 3$

Co-funded by the Erasmus+ Programme of the European Union

Maximum Flow Problem
Network with Directed Arcs

Iteration 5:
Node 1: $[-, \infty]$ Node 2: $[1+, 3]$ Node 3*: $[2-, 2]$ Node 6: $[3+, 1]$

Breakthrough path: 1-2-3-6.
New feasible solution: $x_{13} = 10$, $x_{32} = 1$, $x_{24} = x_{46} = 5$, $x_{12} = 5$, $x_{25} = 1$, $x_{56} = 4$ $x_{36} = 6$, $x_{35} = 3$

Iteration 6:
Node 1: $[-, \infty]$ Node 2: $[1+, 2]$ Node 3*: $[2-, 1]$

The breakthrough path does not exist: $X = \{1,2,3\}$ $\overline{X} = \{4,5,6\}$
The current solution is optimal.
The maximum flow: $c(X, \overline{X}) = c_{24} + c_{25} + c_{35} + c_{36} = 5 + 1 + 3 + 6 = 15$

Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Directed Arcs

Note:

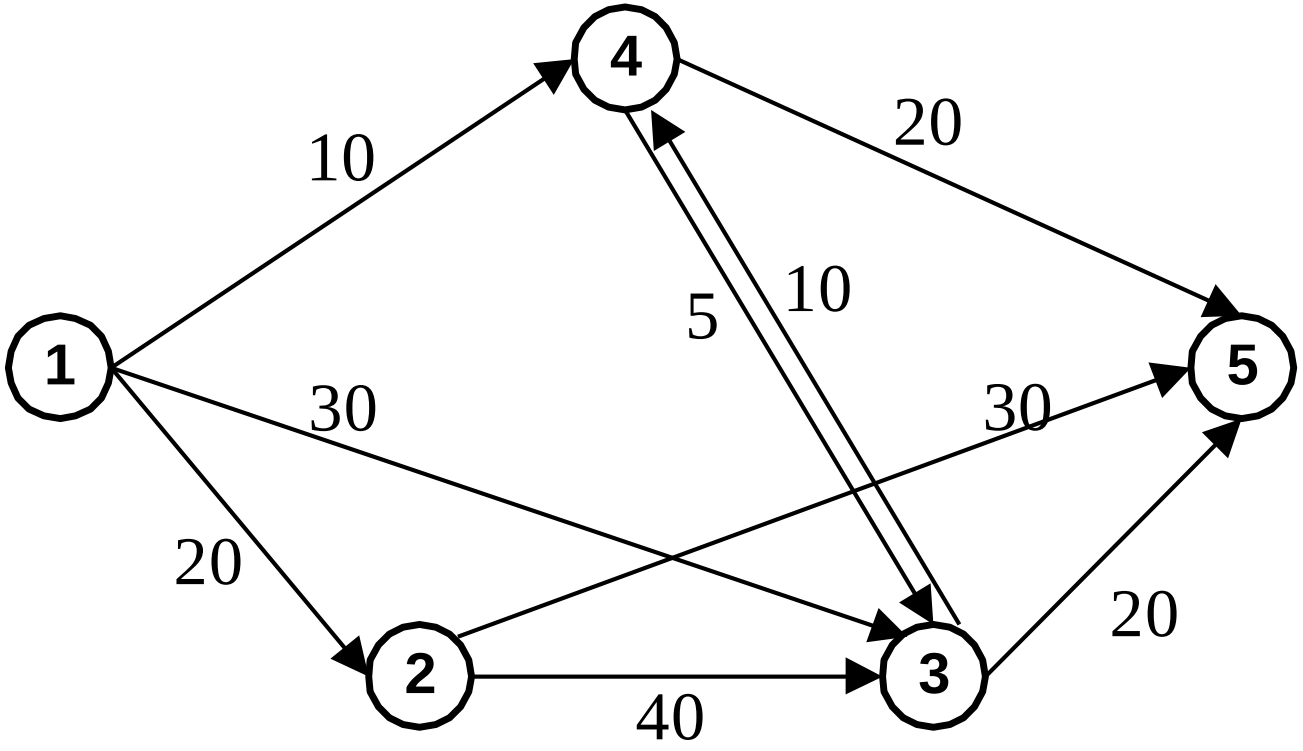
For problem with multiple source nodes with limited supply capacity and multiple sink nodes with limited demand, an artificial source node and an artificial sink node will be introduced and the Ford-Fulkerson's algorithm can be applied to find optimal solution. It is also noted the maximum flow problem is a transportation problem in which the objective is to maximize the amount of goods transported from supply nodes to demand nodes.

Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Undirected Arcs

Ford-Fulkerson's algorithm can be applied. However, it is noted that there exist 2 variables for an undirected arc

Example 23:



Co-funded by the
Erasmus+ Programme
of the European Union

Maximum Flow Problem
Network with Undirected Arcs

Iteration 1: starting feasible solution: zeros for all arcs
Node 1: $[-, \infty]$ Node 3: $[1+, 30]$ Node 5: $[3+, 20]$

Breakthrough path: 1-3-5.
New feasible solution: $x_{13} = x_{35} = 20$, other variables = 0.

Iteration 2:
Node 1: $[-, \infty]$ Node 2: $[1+, 20]$ Node 3: $[2+, 20]$ Node 4: $[3+, 10]$
Node 5: $[4+, 10]$
Breakthrough path: 1-2-3-4-5.
New feasible solution: $x_{12} = x_{23} = x_{34} = x_{45} = 10$, $x_{13} = x_{35} = 20$, other variables = 0.

Co-funded by the Erasmus+ Programme of the European Union

Maximum Flow Problem
Network with Undirected Arcs

Iteration 3:

Node 1: $[-, \infty]$ Node 4: $[1+, 10]$ Node 5: $[4+, 10]$

Breakthrough path: 1-4-5.

New feasible solution: $x_{14} = 10,\ x_{12} = x_{23} = x_{34} = 10,\ x_{45} = 20,\ x_{13} = x_{35} = 20,$ other variables = 0.

Iteration 4:

Node 1: $[-, \infty]$ Node 2: $[1+, 10]$ Node 5: $[2+, 10]$

Breakthrough path: 1-4-5.

New feasible solution: $x_{14} = 10, x_{12} = 20,\ x_{23} = x_{34} = 10, x_{45} = 20, x_{13} = x_{35} = 20, x_{25} = 10, x_{43} = 0.$

Co-funded by the Erasmus+ Programme of the European Union

Maximum Flow Problem
Network with Undirected Arcs

Iteration 5:

Node 1: $[-,\infty]$ Node 3: $[1+,10]$ Node 2*: $[3-,10]$ Node 5: $[2+,10]$

Breakthrough path: 1-3-2-5.

New feasible solution: $x_{14}=10, x_{12}=20,\ x_{23}=0,\ x_{34}=10,\ x_{45}=20,\ x_{13}=30, x_{35}=20,\ x_{25}=20,\ x_{43}=0.$

Iteration 6:

Node 1: $[-,\infty]$

The breakthrough path does not exist: $X=\{1\}\quad \overline{X}=\{2,3,4,5\}$

The current solution is optimal.

Maximum flow: $c(X,\overline{X})=c_{12}+c_{13}+c_{14}=20+30+10=60$

Co-funded by the Erasmus+ Programme of the European Union

Shortage Path Problem

Consider a network with *m* nodes and *n* directed arcs and a cost c_{ij} is associated with each arc (i,j). The shortage path problem is to find the shortage (least costly) path from node 1 to node *m*.

The problem can be formulated as an LP as follows:

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

$$\text{s.t.} \quad \sum_j x_{ij} - \sum_k x_{ki} = \begin{cases} 1 & \text{if} \quad i = 1 \\ 0 & \text{if} \quad i = 2,3,\dots,m-1 \\ -1 & \text{if} \quad i = m \end{cases}$$

$$x_{ij} = 0 \text{ or } 1 \qquad \forall (i,j)$$

In which: $x_{ij} = 0$: (i,j) is not included in the path,

$x_{ij} = 1$: (i,j) is included in the path.

Co-funded by the
Erasmus+ Programme
of the European Union

Shortage Path Problem
Dijkstra's Labeling Algorithm

Step 1:

Assign label $L_i = [-, v_i]$ for network's nodes, starting with $v_1 = 0$, $v_2 = v_3 = \cdots = v_m = \infty$.

Step 2:

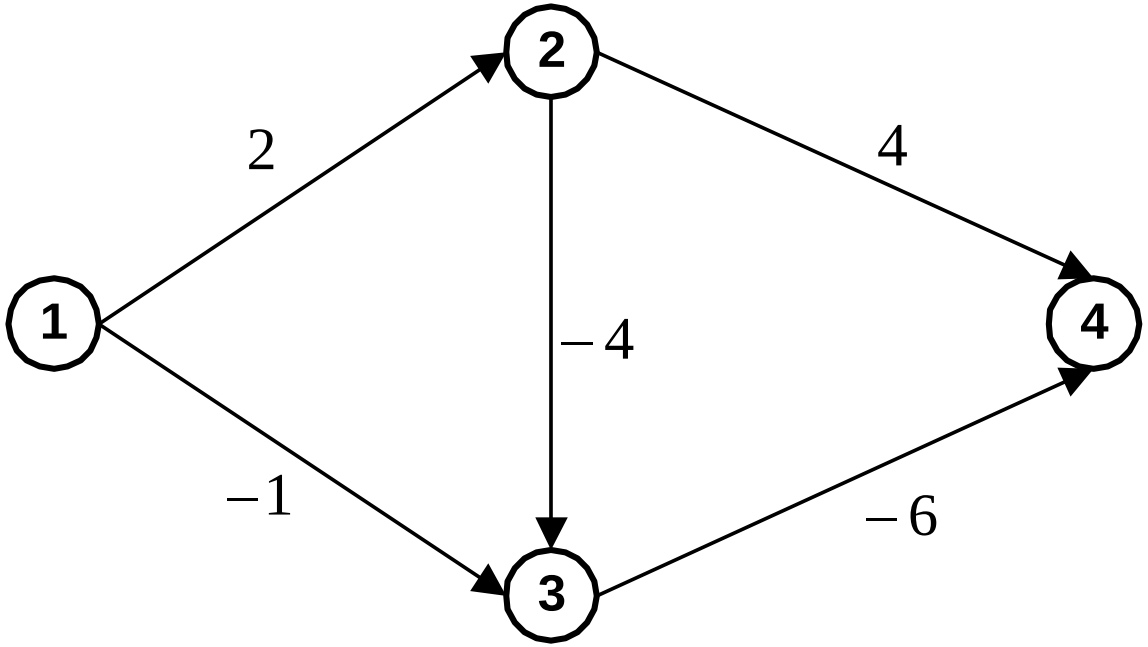
1. If $v_j \leq v_i + c_{ij}\ \forall(i,j)\ \Rightarrow$ The current solution is optimal.
2. If there exist an arc (p,q) such that $v_q \leq v_p + c_{pq} \Rightarrow$ reassign the label of node q: $L_q = [p, v_q = v_p + c_{pq}]$. Return to check 2.1

Co-funded by the Erasmus+ Programme of the European Union

Shortage Path Problem
Dijkstra's Labeling Algorithm

Example 24:

Find the shortage path from 1 to 4 for the following network



Co-funded by the
Erasmus+ Programme
of the European Union

Shortage Path Problem
Dijkstra's Labeling Algorithm

Iteration 1: $v_1 = 0, v_2 = v_3 = v_4 = \infty$

Iteration 2: $\infty = v_3 > v_1 + c_{13} = -1 \quad \Rightarrow v_3 = -1, L_3 = [1, -1]$

Iteration 3: $\infty = v_2 > v_1 + c_{12} = 2 \Rightarrow v_2 = 2, L_2 = [1,2]$

Iteration 4: $-1 = v_3 > v_2 + c_{23} = -2 \quad \Rightarrow v_3 = -2, L_3 = [2, -2]$

Iteration 5: $\infty = v_4 > v_2 + c_{24} = 6 \Rightarrow v_4 = 6, L_4 = [2,6]$:

(redundant step!)

Iteration 6: $6 = v_4 > v_3 + c_{34} = -8 \quad \Rightarrow v_4 = -8, L_4 = [3, -8]$

The current solution is optimal: $v_j \leq v_i + c_{ij}\ \forall (i,j)$.

The shortest path: 1-2-3-4. Minimum "cost" $v_4 = -8$.

Co-funded by the
Erasmus+ Programme
of the European Union

Shortage Path Problem

Labeling Algorithm with Nonnegative Costs

Let $N = \{1,2,3,\dots,m\}$ be the set of network's nodes.

Step 1: Set $v_1 = 0$ and $X = \{1\}$.

Step 2: Set $\overline{X} = N - X$, $(X, \overline{X}) = \{(i,j) | i \in X, j \in \overline{X}\}$.
Select $(i,j) \in (X, \overline{X})$ such that:

$$v_p + c_{pq} = \min_{(i,j)\in(X,\overline{X})} \{v_i + c_{ij}\}$$

Assign $v_q = v_p + c_{pq}$ and adjust $X = X \cup \{1\}$.

Repeat step 2 exactly $(m-1)$ times to get the optimal solution.

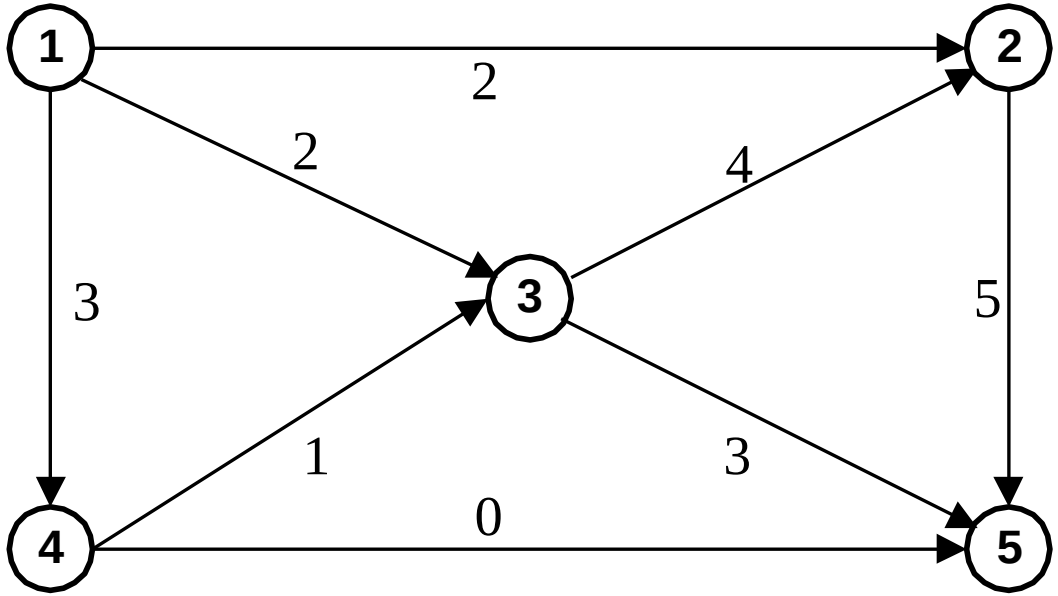
Co-funded by the
Erasmus+ Programme
of the European Union

Shortage Path Problem

Labeling Algorithm with Nonnegative Costs

Example 24:

Consider the shortage path problem from node 1 to node 5 for the following network



Co-funded by the
Erasmus+ Programme
of the European Union

Shortage Path Problem

Labeling Algorithm with Nonnegative Costs

Iteration 1: $v_1 = 0, X = \{1\}, \overline{X} = \{2,3,4,5\}$

$\Rightarrow (X, \overline{X}) = \{(1,2), (1,3), (1,4)\}$: $v_1 + c_{12} = 2$; $v_1 + c_{13} = 2$; $v_1 + c_{14} = 3$

Select $(p, q) = (1,2)$.

Iteration 2: $v_2 = 2, X = \{1,2\}, \overline{X} = \{3,4,5\}$

$\Rightarrow (X, \overline{X}) = \{(1,3), (1,4), (2,5)\}$: $v_1 + c_{13} = 2$; $v_1 + c_{14} = 3$; $v_2 + c_{25} = 7$

Select $(p, q) = (1,3)$.

Iteration 3: $v_3 = 2, X = \{1,2,3\}, \overline{X} = \{4,5\}$

$\Rightarrow (X, \overline{X}) = \{(1,4), (2,5), (3,5)\}$: $v_1 + c_{14} = 3$; $v_2 + c_{25} = 7$; $v_3 + c_{35} = 7$

Select $(p, q) = (1,4)$.

Co-funded by the
Erasmus+ Programme
of the European Union

Shortage Path Problem

Labeling Algorithm with Nonnegative Costs

Iteration 4: $v_4 = 3$, $X = \{1,2,3,4\}$, $\overline{X} = \{5\}$

$\Rightarrow (X, \overline{X}) = \{(2,5), (3,5), (4,5)\}$: $v_2 + c_{25} = 7$; $v_3 + c_{35} = 7$; $v_4 + c_{45} = 3$

Select $(p, q) = (4,5)$.

Iteration 5: $v_5 = 3$, $X = \{1,2,3,4,5\}$, $\overline{X} = \{\emptyset\}$

The shortage path: 1-4-5. Minimum "cost": $v_5 = 3$

Co-funded by the
Erasmus+ Programme
of the European Union

Shortage Path Problem

Labeling Algorithm with Nonnegative Costs

Note:

- In some cases, we have to deal with the longest path problem (the case of profit instead of cost)
- To find the solution for the longest path problem, change the sign of all “profits” on arcs and apply the labeling algorithms for the shortage path problem.

Co-funded by the
Erasmus+ Programme
of the European Union