INTRODUCTION TO THE AMPL LANGUAGE

Esteve Codina

Universitat Politècnica de Catalunya

Grau d'Enginyeria Informàtica **FIB**

CONTENTS

- Language Basics
  - Elements: variables, constraints, objective function
  - How to run the system
- Problems with Network Flow constraints
  - The simplest problem (the min-cost problem)
  - The node and arc statements

Linear Objective Function

Affine constraints

$$
\begin{aligned}
Min_x \quad & c_1 x_1 + \ldots + c_n x_n \\
s.a: \quad & a_{11} x_1 + \ldots + a_{1n} x_n \geq b_1 \\
& \ldots \\
& a_{p1} x_1 + \ldots + a_{pn} x_n \geq b_p \\
& d_{11} x_1 + \ldots + d_{1n} x_n \leq e_1 \\
& \ldots \\
& d_{q1} x_1 + \ldots + d_{qn} x_n \leq e_q \\
& g_{11} x_1 + \ldots + g_{1n} x_n = h_1 \\
& \ldots \\
& g_{r1} x_1 + \ldots + g_{rn} x_n = h_r
\end{aligned}
$$

MATRIX NOTATION

$$
\begin{aligned}
Min_x \quad & c^\top x \\
s.a: \quad & Ax \geq b \\
& Dx \leq e \\
& Gx = h
\end{aligned}
$$

ALGEBRAIC LANGUAJES FOR OPTIMIZATION PROBLEMS: AMPL

Oriented to forrmulate optimization problems very efficiently.

- They are oriented to the development of models commonly used in OR
- The formulations are: understandable to other users and can be easily modified and extended.
- For solving the models, general purpose SOLVERS are used
- Usually there are environments suited for developement/debugging.
- AMPL language is very well suited and featured for complex data structures in large/speceialized models
- Other languajes: GAMS, CAMPS, LINGO.
- http://www.ampl.com/

$$
\begin{aligned}
Max_x \quad & \sum_{i=1}^{n} c_i x_i \\
s.a: \quad & \sum_{i=1}^{n} a_i x_i \leq b \\
& 0 \leq x_i \leq u_i, \quad i = 1, 2, ..., n
\end{aligned}
$$

$$
\begin{aligned}
Max_x \quad & c^\top x \\
s.a: \quad & a^\top x \leq b \\
& 0 \leq x \leq u
\end{aligned}
$$

$$
\begin{aligned}
Max_x \quad & \sum_{i \in P} c_i x_i \\
s.a: \quad & \sum_{i \in P} a_i x_i \leq b \\
& 0 \leq x_i \leq u_i, \quad i \in P
\end{aligned}
$$

$$P = \{1, 2, ..., n\}$$

$$P = \{\texttt{bandas},\ \texttt{bobinas}\}$$

Parameters

c – cost vector

a – resource consumption vector

b - Amount of resource

u – vector of upper bounds

File prod.mod

```
set P;
param a {j in P};
param b;
param c {j in P};
param u {j in P};

var X {j in P};

maximize beneficio: sum {j in P} c[j] * X[j];

subject to tiempo: sum {j in P} a[j] * X[j]<=b;
subject to Limites {j in P}: 0 <= X[j] <= u[j];
```

$$
\begin{aligned}
Max_x \quad & \sum_{i \in P} c_i x_i \\
s.a: \quad & \sum_{i \in P} a_i x_i \leq b \\
& 0 \leq x_i \leq u_i, \quad i \in P
\end{aligned}
$$

File prod.dat

```
set P := bandas bobinas;
param:      a      c      u   :=
bandas     200    25    6000
bobinas    140    30    4000  ;
param b := 40;
```

```
set P := bandas bobinas;
param: a:= bandas   200 bobinas  140;
param: c:= bandas    25 bobinas   30;
param: u:= bandas  6000 bobinas 4000;
param b := 40;
```

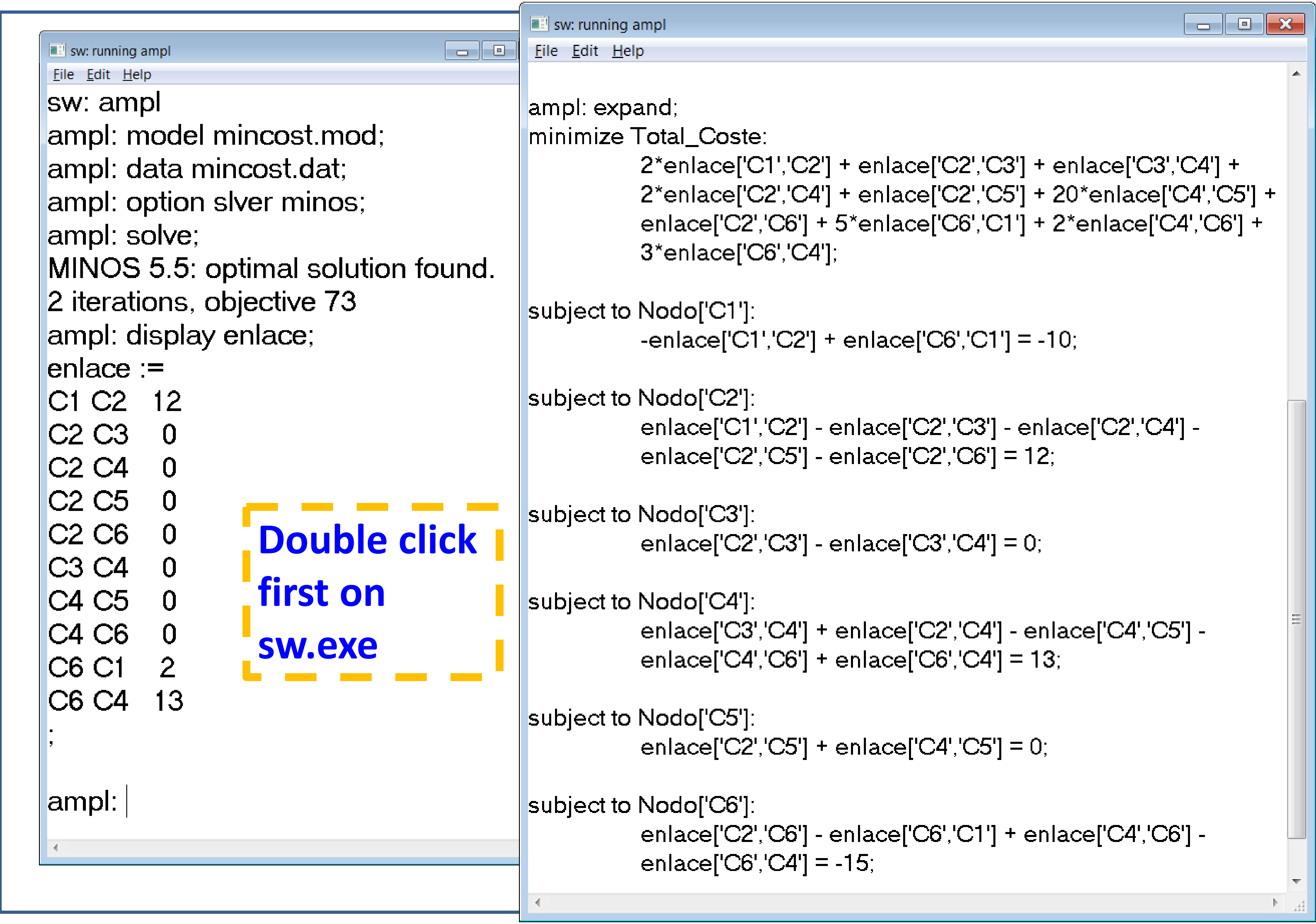
- **Download from ATENEA the zip file containing the AMPL Student versión.**
- **Extract its contents on a physical directory (your 'AMPL directory').**
- **That directory must contain also your working files:**
  - .mod files, (containing models)
  - .dat files, (containing data sets for model instances)
  - .run files (scripts)

  and thus, will have both functions of AMPL and working directory.
- **If the IDE system is going to be used (optional) then, download it into a subdirectory of your 'AMPL directory'.**
- **If you prefer not to use the IDE system then, you may:**
  - Work directly with an MS-DOS command window
  - Use sw.exe to create your command window

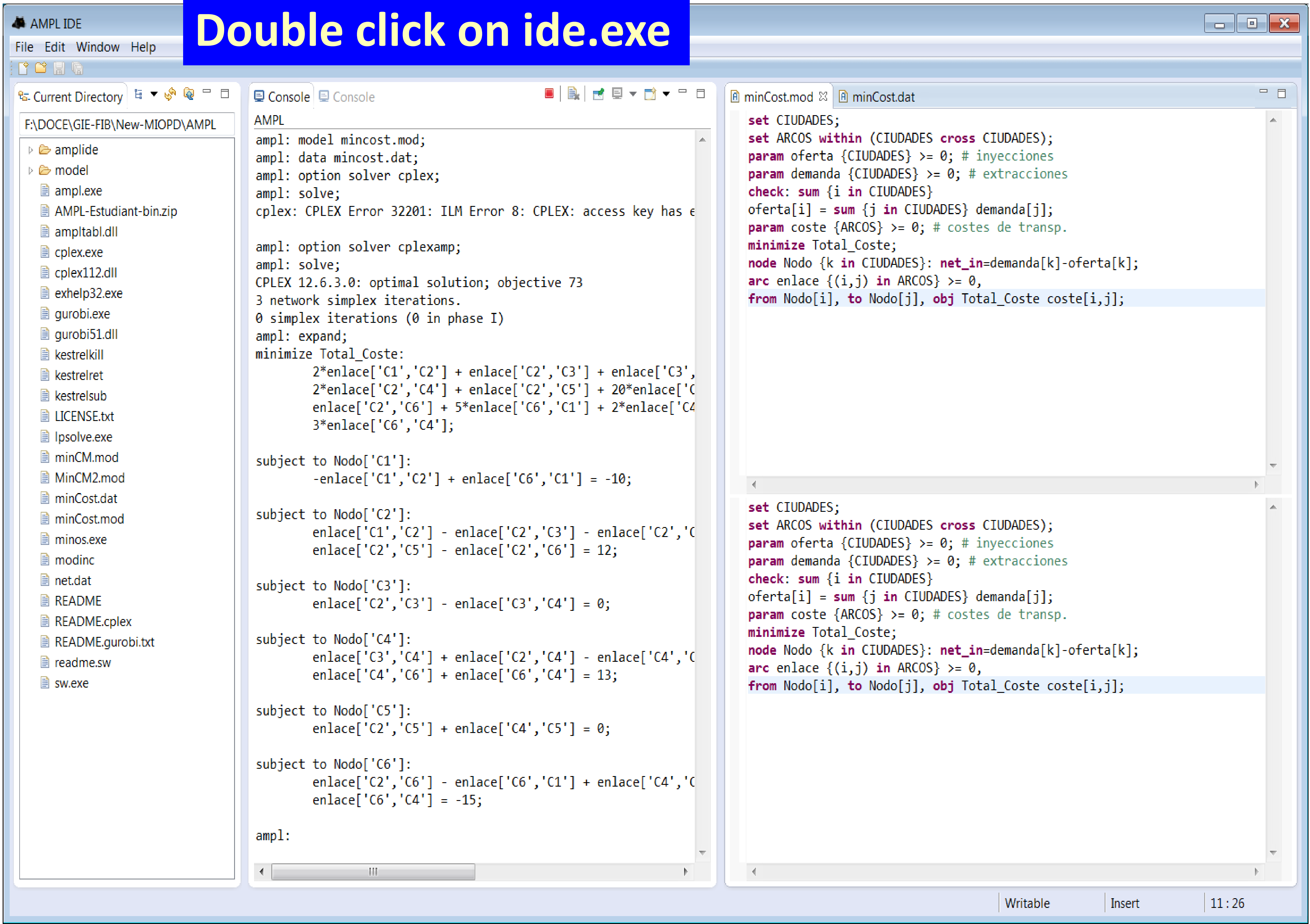
C:\windows\system32\cmd.exe - ampl

```
C:\Users\Esteve>cd F:\DOCE\GIE-FIB\New-MIOPD\AMPL

C:\Users\Esteve>F:

F:\DOCE\GIE-FIB\New-MIOPD\AMPL>ampl
ampl: model mincost.mod;
ampl: data mincost.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.1.0: optimal solution; objective 73
1 simplex iterations
ampl: show;

parameters:   coste   demanda   oferta

sets:   ARCOS   CIUDADES

variable:   enlace

constraint:   Nodo

objective:   Total_Coste

checks: one, called check 1.
ampl: reset;
ampl: _
```

The MS-DOS command window


sw: running ampl
File Edit Help
sw: ampl
ampl: model mincost.mod;
ampl: data mincost.dat;
ampl: option slver minos;
ampl: solve;
MINOS 5.5: optimal solution found.
2 iterations, objective 73
ampl: display enlace;
enlace :=
C1 C2 12
C2 C3 0
C2 C4 0
C2 C5 0
C2 C6 0
C3 C4 0
C4 C5 0
C4 C6 0
C6 C1 2
C6 C4 13
;
ampl:
Double click first on sw.exe
sw: running ampl
File Edit Help
ampl: expand;
minimize Total_Coste:
2*enlace['C1','C2'] + enlace['C2','C3'] + enlace['C3','C4'] +
2*enlace['C2','C4'] + enlace['C2','C5'] + 20*enlace['C4','C5'] +
enlace['C2','C6'] + 5*enlace['C6','C1'] + 2*enlace['C4','C6'] +
3*enlace['C6','C4'];
subject to Nodo['C1']:
-enlace['C1','C2'] + enlace['C6','C1'] = -10;
subject to Nodo['C2']:
enlace['C1','C2'] - enlace['C2','C3'] - enlace['C2','C4'] -
enlace['C2','C5'] - enlace['C2','C6'] = 12;
subject to Nodo['C3']:
enlace['C2','C3'] - enlace['C3','C4'] = 0;
subject to Nodo['C4']:
enlace['C3','C4'] + enlace['C2','C4'] - enlace['C4','C5'] -
enlace['C4','C6'] + enlace['C6','C4'] = 13;
subject to Nodo['C5']:
enlace['C2','C5'] + enlace['C4','C5'] = 0;
subject to Nodo['C6']:
enlace['C2','C6'] - enlace['C6','C1'] + enlace['C4','C6'] -
enlace['C6','C4'] = -15;

Double click on ide.exe



EXAMPLE: Production Problems

The amounts $x_i \geq 0$, $i = 1, \ldots, n$ for n products must be determined so that the overall benefit of the production is maximized.

Benefits per unit of product i is c_i, $i = 1, \ldots, n$.

Total benefit: $c_1 x_1 + \ldots + c_n x_n$ (→ maximize)

m resoures are required, each of them available in quantities b_j , $j = 1, 2, \ldots, m$.

Each unit of product i requires a_{ij} units of resource j :

$$a_{j1} x_1 + \ldots + a_{jn} x_n \leq b_j$$

$$\begin{aligned} Max_x \quad & c_1 x_1 + \ldots + c_n x_n \\ s.a: \quad & a_{11} x_1 + \ldots + a_{1n} x_n \leq b_1 \\ & \ldots \\ & a_{m1} x_1 + \ldots + a_{mn} x_n \leq b_m \\ & x_1, \ldots, x_n \geq 0 \end{aligned}$$

$$\begin{aligned} Max_x \quad & c^\top x \\ s.a: \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

	Amount of each resource needed to produce one unit of the product A		Amount of each resource needed to produce one unit of the product B		Total Availability of each resource (per week))
	Process 1	Process 2	Process 3	Process 4	
Person / Week	1	1	1	1	15
Kg Material Y	7	5	3	2	120
Kg Material Z	3	5	10	15	100
Unit profit	4	5	9	11	

SOLUTION:

$$\begin{aligned}
\textbf{Max} \quad & 4x_1 + 5x_2 + 9x_3 + 11x_4 \\
& x_1 + x_2 + x_3 + x_4 \leq 15 \\
& 7x_1 + 5x_2 + 3x_3 + 2x_4 \leq 120 \\
& 3x_1 + 5x_2 + 10x_3 + 15x_4 \leq 100 \\
& x_1, x_2, x_3, x_4 \geq 0
\end{aligned}$$

plan_prod.mod

```
set PRODUCTS;
set RESOURCES;

param profit{PRODUCTS}>=0;
param resources_requirements{PRODUCTS, RESOURCES} >=0;
param max_resource{RESOURCES} >=0;

var X {PRODUCTS} >=0;

# Objective function
maximize total:
         sum {i in PRODUCTS} (profit[i]*X[i]);

# Constraints
# Resources
subject to resource_availability {j in RESOURCES}:
   sum {i in PRODUCTS} (resources_requirements[i,j]*X[i]) <= max_resource[j];
```

```
set PRODUCTS:= 1 2 3 4;
set RESOURCES:= 1 2 3;
param profit:=
1 4
2 5
3 9
4 11;

param resources_requirements [*,*]
: 1 2 3 :=
1 1 7 3
2 1 5 5
3 1 3 10
4 1 2 15;

param max_profit:=
1 15
2 120
3 100;
```

File plan_prod.dat

```
# Reset previously commands in AMPL
reset;

# Model Load
model plan_prod.mod;

# Data Load
data plan_prod.dat;

# Selection of the solver: CPLEX
option solver cplex;

#Solve the problem
solve;

# Show the obtained results
display total;
display X;
```

File
plan_prod.run

include

D:\USUARIS\mari.paz.l

```
ampl: include plan_prod.run;
CPLEX 12.6.0.1: optimal solution
3 dual simplex iterations (1 in
total = 99.2857

X [*] :=
1  7.14286
2  0
3  7.85714
4  0
;

ampl: _
```

DOUBLE INDEXES

BOUNDS ON DECISION VARIABLES

CONDITIONS ON PARAMETER VALUES

INDEXED CONSTRAINTS

```
set P;
set ETAPA;

param tasa{P,ETAPA} > 0;
param recurso{ETAPA} >= 0;
param benef_u{P};
param x_min{P} >= 0;
param x_mercado{P} >= 0;

var X {p in P} >= x_min[p], <= x_mercado[p];

maximize total_benef_u: sum {p in P} benef_u[p]*X[p];

subject to Tiempo {s in ETAPA}:
   sum {p in P} tasa[p,s]* X[p] <= recurso[s];
```

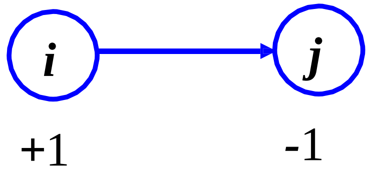
CONTENTS

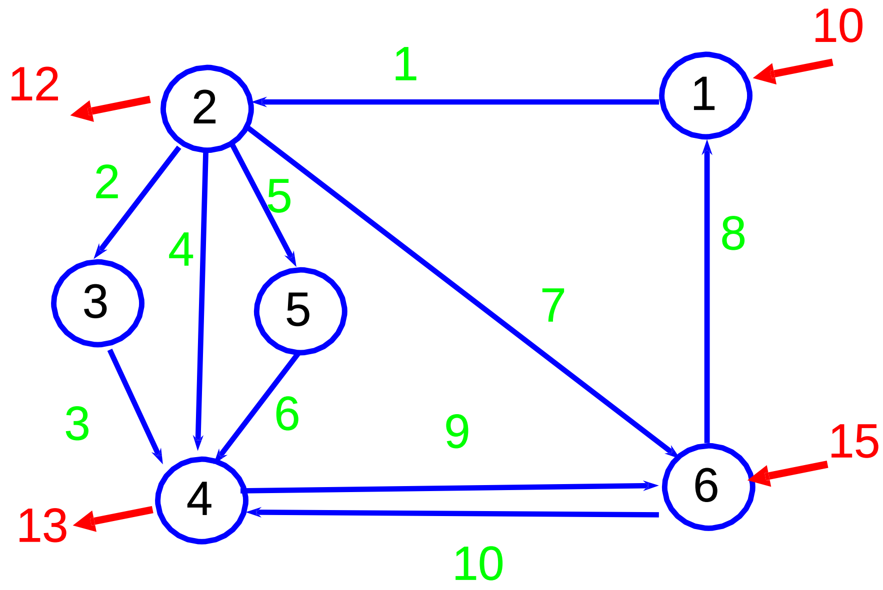
- Language Basics
  - Elements: variables, constraints, objective function
  - How to run the system
- Problems with Network Flow constraints
  - The simplest problem (the min-cost problem)
  - The node and arc statements

NETWORK FLOWS

Node 1: $x_{12} - x_{61} = 10$

Node 2: $x_{23} + x_{24} + x_{25} + x_{26} - x_{12} = -12$





$$\sum_{(i,j)\in E(i)} x_{ij} - \sum_{(j,i)\in I(i)} x_{ij} = \mathbf{b}_i\ ,\ i \in \mathrm{N}$$

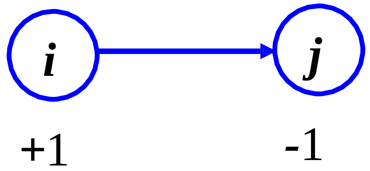
Notice that, for a feasible problem, $\sum_{i\in \mathrm{N}} \mathbf{b}_i = 0$

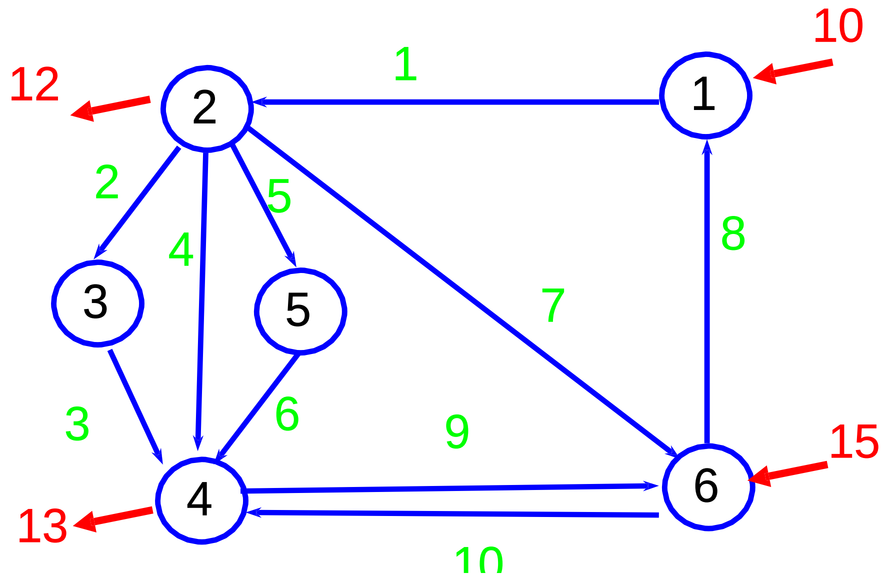
NETWORK FLOWS

Nudo 1: $x_1 - x_8 = 10$

Nudo 2: $x_2 + x_4 + x_5 + x_7 - x_1 = -12$

…





$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ -1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_9 \\ x_{10} \end{pmatrix} = \begin{pmatrix} 10 \\ -12 \\ 0 \\ -13 \\ 0 \\ 15 \end{pmatrix}$$

NODE-LINK INCIDENCE MATRIX *D*

FLOW VECTOR *x*

INP./OUTP. *b*

$x_i \geq 0, \ i = 1, \ldots, 10$

Sometimes upper bounds are required on the flows: $x_i \leq u_i$

MIN-COST FLOW PROBLEM: DEFINITION

D - **NODE-LINK INCIDENCE MATRIX**

x - **FLOW VECTOR (**DECISION VARIABLES**)**

b - **INJECTIONS/EXTRACTIONS VECTOR**

l, u – **LOWER-UPPER BOUNDS VECTORS**

$$\mathrm{Min}_{x}\ c^{\mathrm{T}} x$$

$$Dx = b$$

$$l \leq x \leq u$$

Typically $l = 0;$

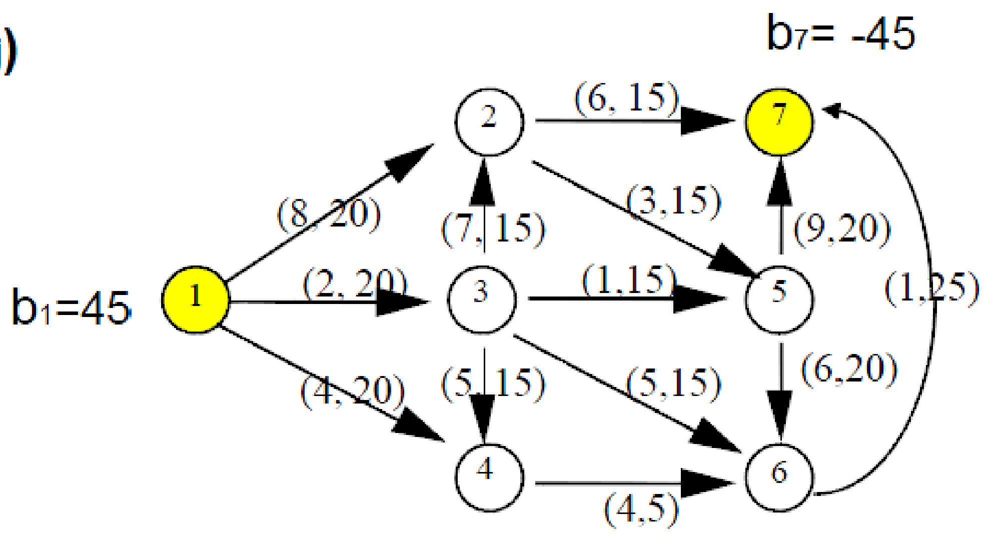
if $l \neq 0$ the problema can be easily reformulated using new decision variables $y = x - l$

Typically $c \geq 0;$

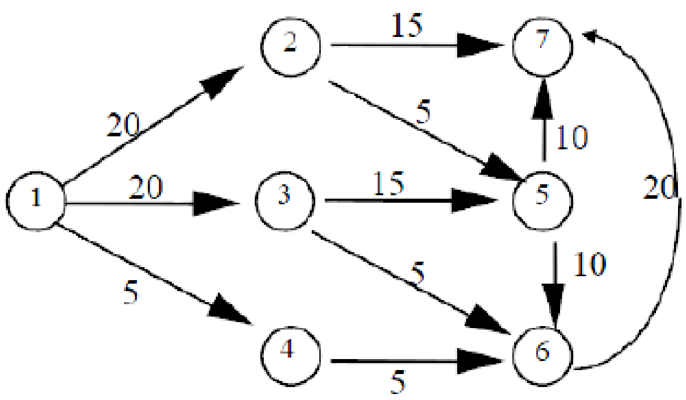
AMPL LANGUAGE: DECLARATIONS NODE, ARC

```
set CIUDADES;

set ARCOS within (CIUDADES cross CIUDADES);

param oferta {CIUDADES} >= 0;    # injections
param demanda {CIUDADES} >= 0;   # extractions

param coste {ARCOS} >= 0; # costs   of transp.

minimize Total_Coste;

node Nodo {k in CIUDADES}: net_in=demanda[k]-oferta[k];

arc enlace {(i,j) in ARCOS} >= 0,
  from Nodo[i], to Nodo[j], obj Total_Coste coste[i,j];

check: sum {i in CIUDADES}
       oferta[i] = sum {j in CIUDADES}    demanda[j];
```

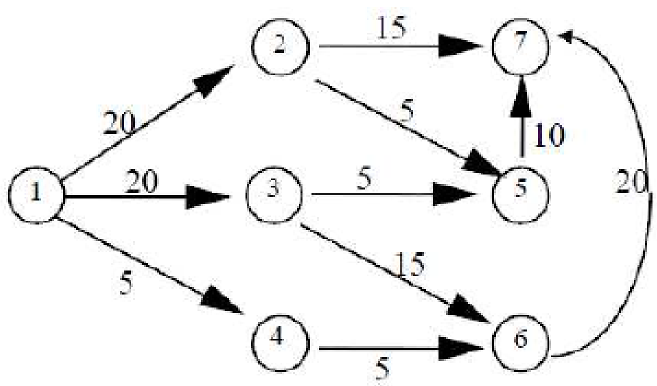
Example: (c_{ij}, d_{ij})



$b_i = 0,\ i=2, \ldots, 6$



Coste 555

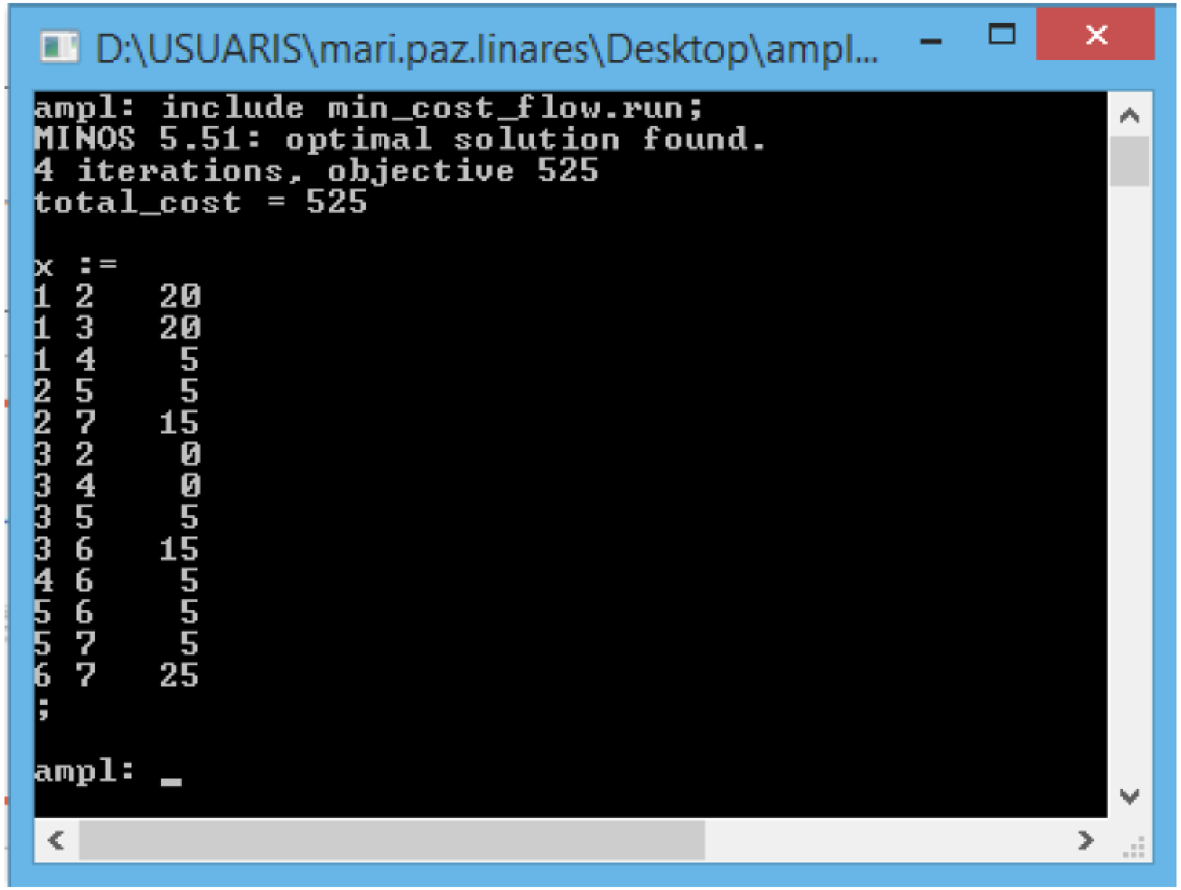


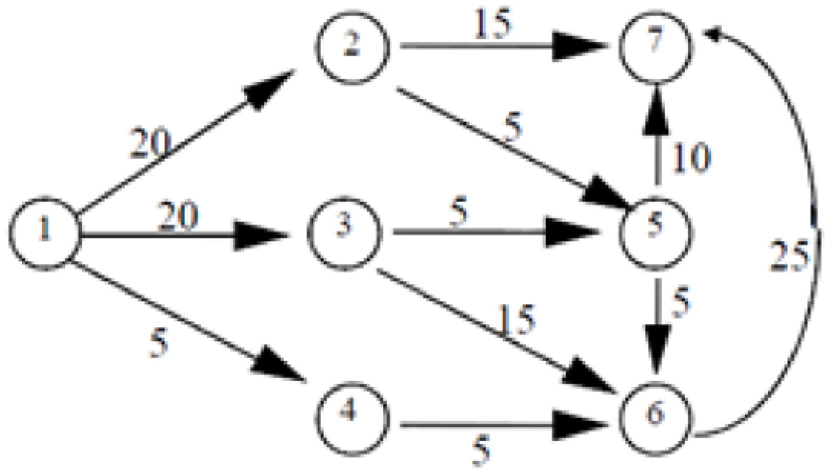
Coste 535

File min_flow_cost.mod

```
# Minimum cost flow problem

#Number of nodes
param n;
set NODES:=1..n;
set ARCS within {NODES,NODES};
param flow{NODES};
param cost{ARCS}>=0;
param capacity{ARCS}>=0;
var x {(i,j) in ARCS}>=0,<=capacity[i,j];

#Objective function
minimize total_cost:
sum{(i,j) in ARCS} cost[i,j]*x[i,j];

# Constraints
subject to cons_nodes{k in NODES}:
(sum{(k,j)in ARCS} x[k,j] - sum{(i,k)in ARCS} x[i,k])=flow[k];
```

```
# Min cost flow problem
param n:=7;
param flow:=
7 -45
6 0
5 0
4 0
3 0
2 0
1 45;

param: ARCS: capacity cost:=
1 2 20 8
1 3 20 2
1 4 20 4
2 5 15 3
2 7 15 6
3 2 15 7
3 4 15 5
3 5 15 1
3 6 15 5
4 6 5 4
5 6 20 6
5 7 20 9
6 7 25 1;
```


D:\USUARIS\mari.paz.linares\Desktop\ampl...
ampl: include min_cost_flow.run;
MINOS 5.51: optimal solution found.
4 iterations, objective 525
total_cost = 525
x :=
1 2 20
1 3 20
1 4 5
2 5 5
2 7 15
3 2 0
3 4 0
3 5 5
3 6 15
4 6 5
5 6 5
5 7 5
6 7 25
;
ampl:


1
2
3
4
5
6
7
20
20
5
15
5
5
15
10
5
5
25

Cost: 525

ANOTHER PROBLEM OF NETWORK FLOWS:
THE MAX-FLOW PROBLEM

Find the maximum flow that can be send from the source node "*s*" to the target node "*t*" in a capacitated network.

$$\max \quad v$$

$$s.t. \quad \sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = 0 \ \forall i \in N \setminus \{s,t\}$$

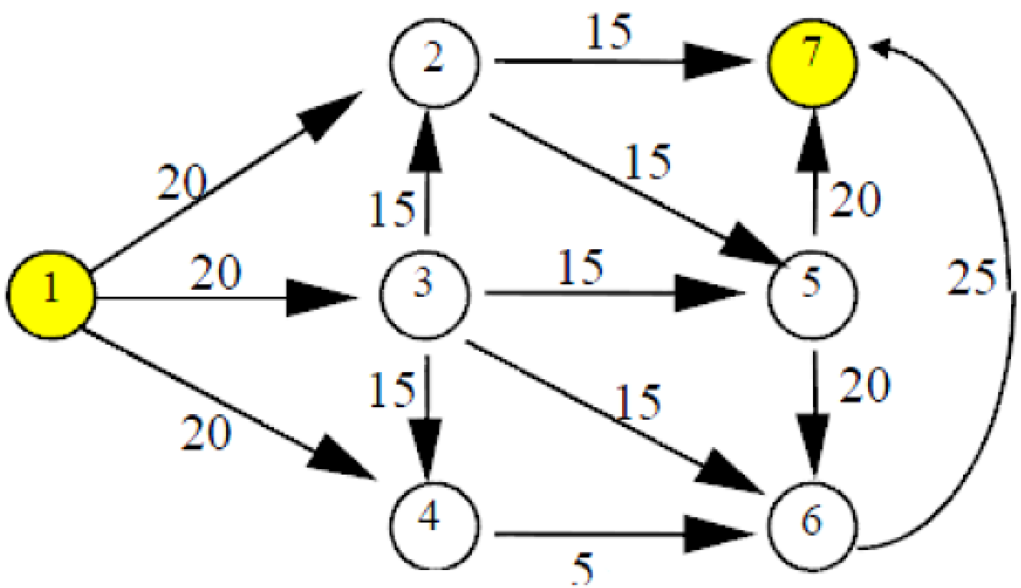
$$\sum_{\{j:(s,j)\in A\}} x_{sj} = v$$

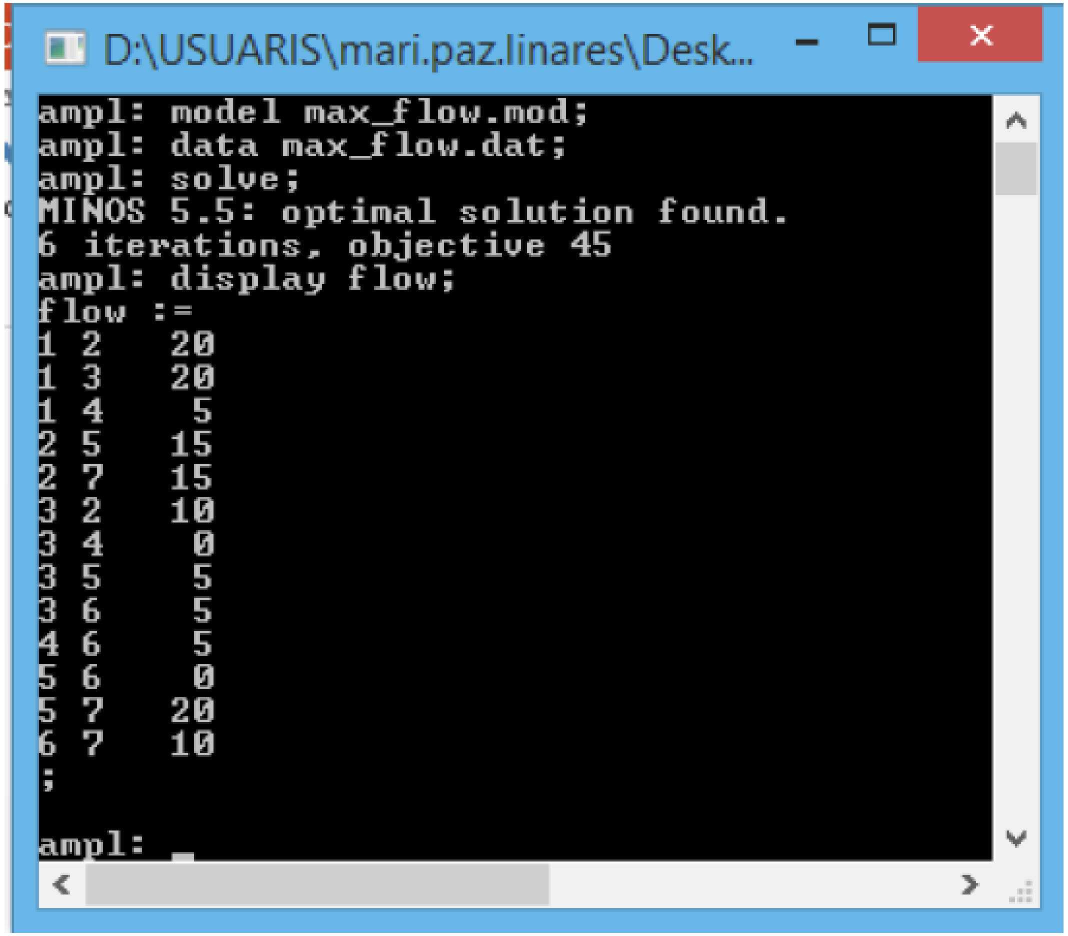
$$\sum_{\{j:(j,t)\in A\}} x_{jt} = v$$

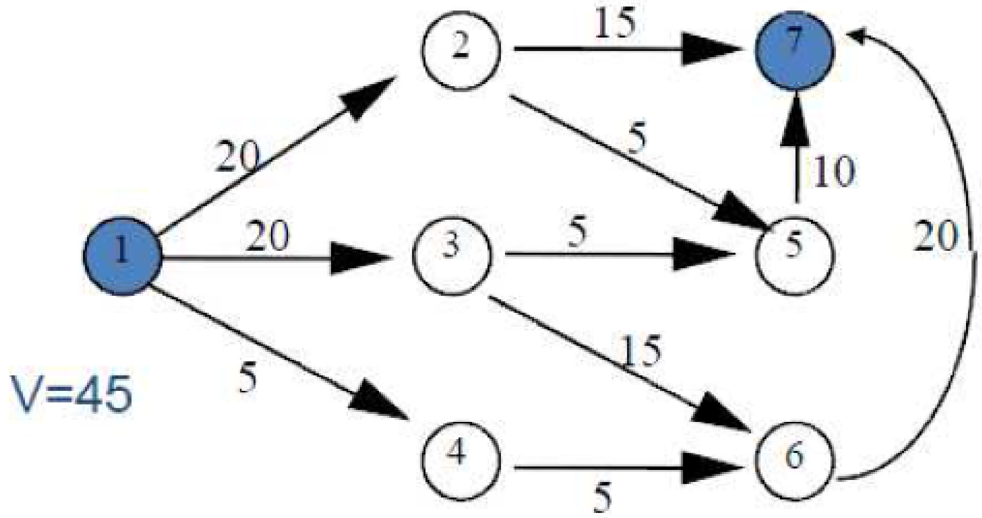
$$x_{ij} \le d_{ij} \ \forall (i,j) \in A$$

$$x_{ij} \ge 0 \ \forall (i,j) \in A, v \ge 0$$

Exercise: Formulate with AMPL the following example and solve it.




D:\USUARIS\mari.paz.linares\Desk...
ampl: model max_flow.mod;
ampl: data max_flow.dat;
ampl: solve;
MINOS 5.5: optimal solution found.
6 iterations, objective 45
ampl: display flow;
flow :=
1 2 20
1 3 20
1 4 5
2 5 15
2 7 15
3 2 10
3 4 0
3 5 5
3 6 5
4 6 5
5 6 0
5 7 20
6 7 10
;
ampl:


V=45
1
2
3
4
5
6
7
20
20
5
15
5
5
15
10
5
20