Course on mathematical modelling: AMPL and CPLEX

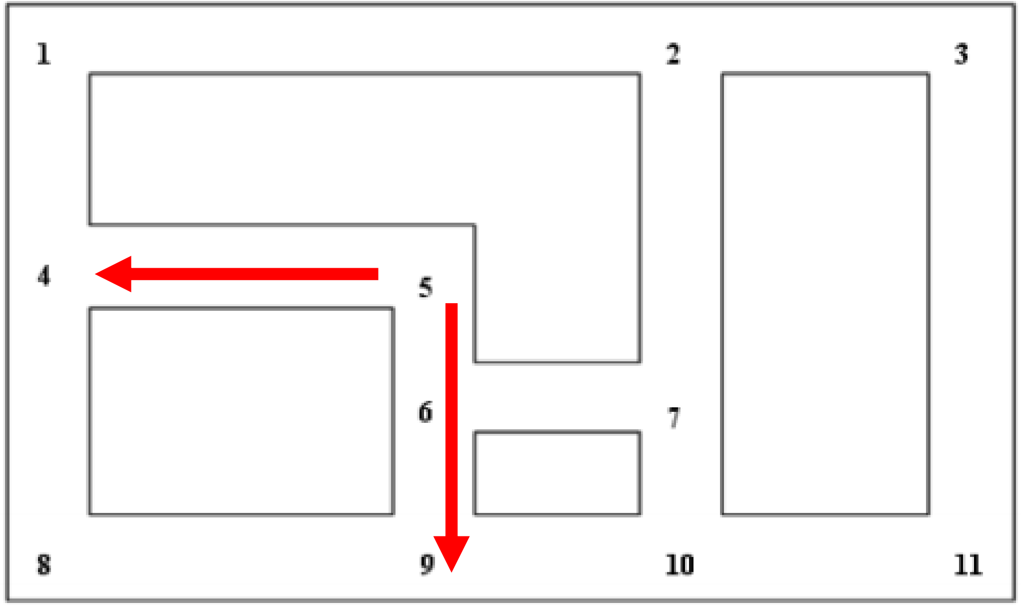
teacher: Giacomo Lanza

Dipartimento di Informatica, Università di Pisa

a.a. 2019-2020

SCLP: a security example

A security officer has to install a new security system on the ground floor of a museum. He has to install a set of cameras so as to control a set of 11 strategic points of the floor. Security cameras may be installed at each of these points (in the map indexed from 1 to 11). Each camera can cover the entire visual plane (from the front to the back). For instance, if a camera is installed at point 5, it controls points 4, 5, 6 and 9. Considering that the costs of installing cameras are [5, 5, 4, 4, 4, 4, 4, 6, 6, 3, 3, 5], define the problem of installing a suitable number of cameras, to control all the strategic points of the ground floor of the museum at minimum cost, as an ILP model



SCLP: a security example

Input data:

- I = set of strategic points (or demand nodes), indexed by i;
- J = set of candidate camera locations, indexed by j;
- $a_{i,j}$ coverage parameters, $i = 1, ..11$, $j = 1, ..11$;

$$a_{i,j} = \begin{cases} 1 & \text{if camera at location } j \text{ covers strategic point } i \\ 0 & \text{otherwise} \end{cases}$$

- c_j, $j = 1, ..11$ cost of installation of camera at location j

SCLP: a security example

Variables:

$$x_j = \begin{cases} 1 & \text{if camera at locatation } j \text{ is installed} \\ 0 & \text{otherwise} \end{cases}$$

$$j = 1, ..11$$

SCLP: a security example

ILP model

$$min \sum_{j=1,..11} c_j \cdot x_j$$

Minimize the total cost of installing cameras

$$\sum_{j=1,..11} a_{i,j} \cdot x_j \geq 1 \qquad \forall\, i = 1,..11;$$

Each strategic point (i from 1 to 11) needs to be covered by a camera

$$x_j \in \{0, 1\} \qquad \forall\, j = 1,..11;$$

Variables are binary

SCLP: a security example

```
# model
param TotPoints ;
param TotCameras;
set Points := 1.. TotPoints ;
set Cameras:= 1..TotCameras;
param Coverage {Points , Cameras};
param InstallingCost {Cameras};

var InstalCam{Cameras} binary;

minimize Total_InstallingCost:  sum {j in Cameras}
InstallingCost[j]* InstalCam[j];

subject to CoveringConstr{i in Points }:
  sum {j in Cameras} Coverage[i,j] * InstalCam[j] >= 1;
```

```
param TotCameras = 11;
param TotPoints  = 11;
param InstallingCost:=
1 5
2 5
3 4
4 4
5 4
6 4
7 4
8 6
9 6
10 3
11 5;
```

SCLP: a security example

```
# model
param TotPoints ;
param TotCameras;
set Points := 1.. TotPoints ;
set Cameras:= 1..TotCameras;
param Coverage {Points , Cameras};
param InstallingCost {Cameras};

var InstalCam{Cameras} binary;

minimize Total_InstallingCost:  sum {j in Cameras}
InstallingCost[j]* InstalCam[j];

subject to CoveringConstr{i in Points }:
  sum {j in Cameras} Coverage[i,j] * InstalCam[j] >= 1;
```

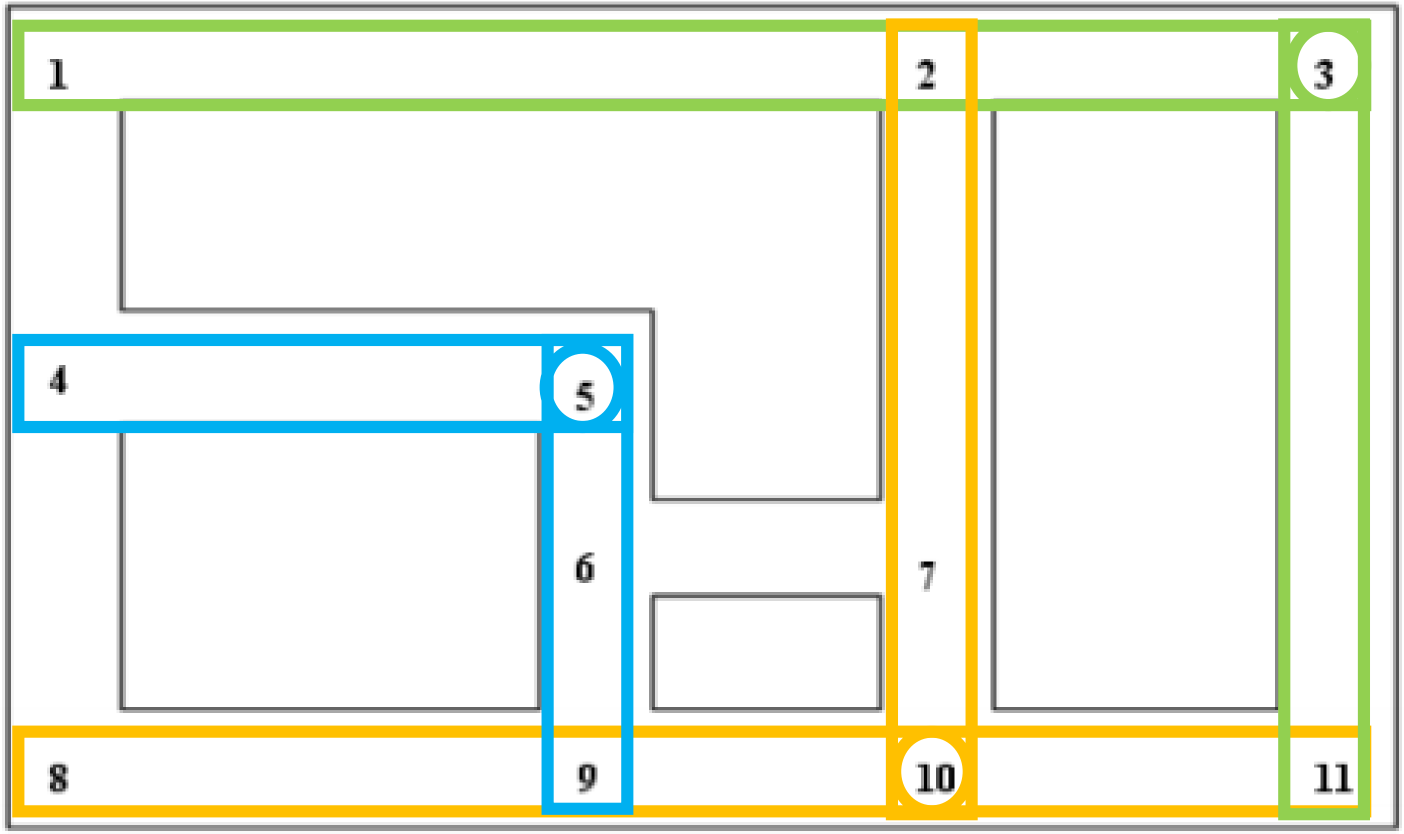
```
param Coverage:
  1 2 3 4 5 6 7 8 9 10 11:=
1 1 1 1 1 0 0 0 1 0 0 0
2 1 1 1 0 0 0 1 0 0 1 0
3 1 1 1 0 0 0 0 0 0 0 1
4 1 0 0 1 1 0 0 1 0 0 0
5 0 0 0 1 1 1 0 0 1 0 0
6 0 0 0 0 1 1 1 0 1 0 0
7 0 1 0 0 0 1 1 0 0 1 0
8 1 0 0 1 0 0 0 1 1 1 1
9 0 0 0 0 1 1 0 1 1 1 1
10 0 1 0 0 0 0 1 1 1 1 1
11 0 0 1 0 0 0 0 1 1 1 1
;
```

SCLP: a security example

```
# Console
ampl: reset;
ampl: model Museum.mod;
ampl: data Museum.dat;
ampl: option solver cplex amp;
ampl: solve;
CPLEX 12.6.1.0: optimal integer solution;
objective 11
15 MIP simplex iterations
0 branch-and-bound nodes
```

```
# Console
ampl: display InstalCam;
InstalCam [*] :=
 1  0
 2  0
 3  1
 4  0
 5  1
 6  0
 7  0
 8  0
 9  0
10  1
11  0;
```

SCLP: a security example




```
# Console
ampl: display InstalCam;
InstalCam [*] :=
 1  0
 2  0
 3  1
 4  0
 5  1
 6  0
 7  0
 8  0
 9  0
10  1
11  0;
```

An example of Minimum Cost Flow Problem

Bavarian Motor Company (BMC) manufactures luxury cars in Germany and exports them in the U.S.; they are currently holding 200 cars available at the port in Newark and 300 cars available at the port in Jacksonville. From there, the cars are transported (by rail or truck) to five distributors having a specific requirement of cars (see Figure). In the network, Newark and Jacksonville are supply nodes (or origins): negative numbers (e.g. -200) represent their supply; Boston, Columbus, Atlanta, Richmond and Mobile are demand nodes (or destinations): positive numbers (e.g. +100) represent their demand.

The problem is to determine how to transport (flowing) cars along the arcs of the network to satisfy the demands at a minimum cost.

An example of Minimum Cost Flow Problem

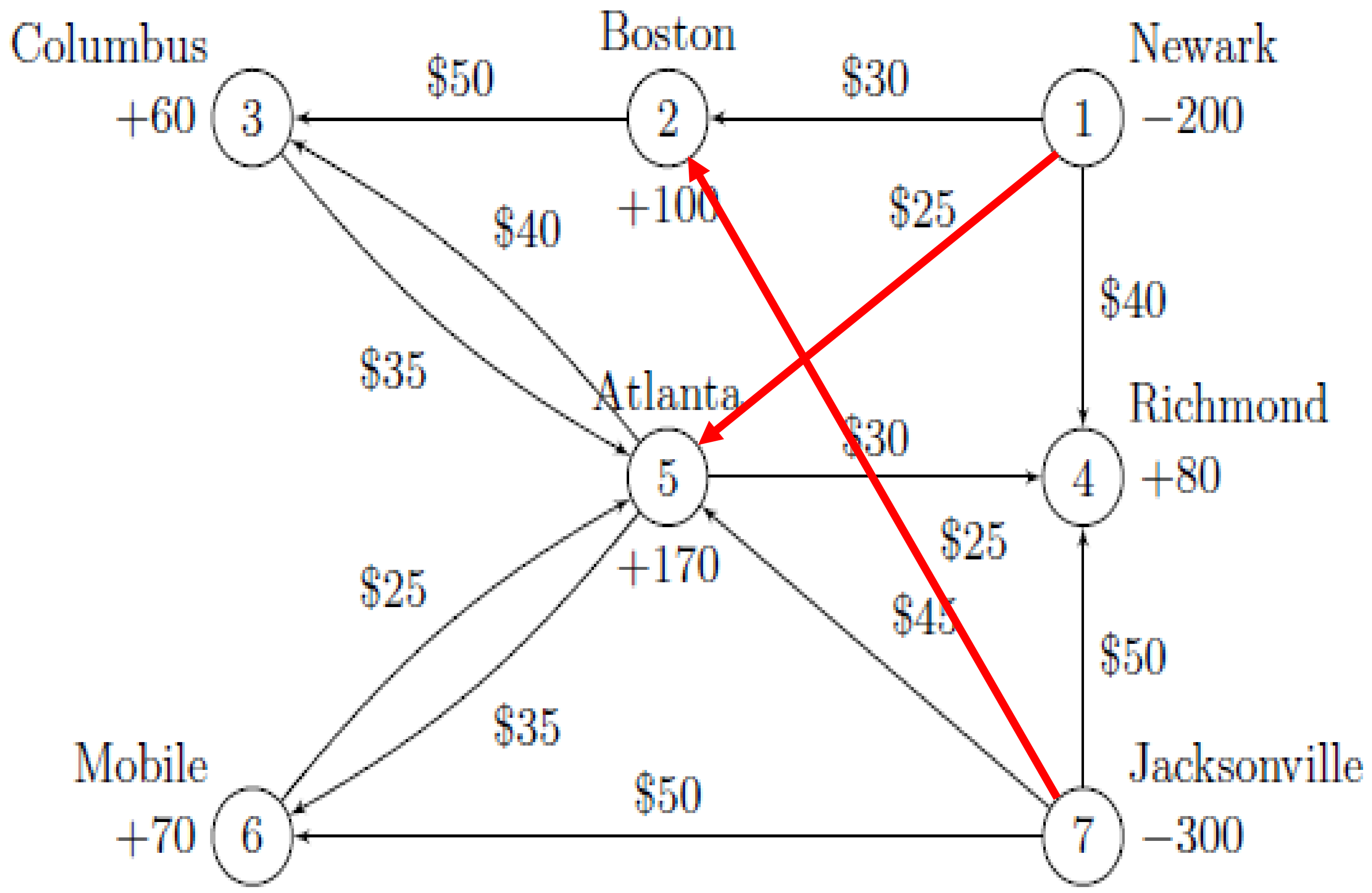
$$
\begin{aligned}
&\min\ 30x_{12} + 40x_{14} + 50x_{23} + 35x_{35} + 40x_{53} + 30x_{54} + \\
&\quad + 35x_{56} + 25x_{65} + 50x_{74} + 45x_{75} + 50x_{76} \\
&-x_{12} - x_{14} \geq -200 \\
&x_{12} - x_{23} = 100 \\
&x_{23} + x_{53} - x_{35} = 60 \\
&x_{14} + x_{54} + x_{74} = 80 \\
&x_{35} + x_{65} + x_{75} - x_{53} - x_{54} - x_{56} = 170 \\
&x_{56} + x_{76} - x_{65} = 70 \\
&-x_{74} - x_{75} - x_{76} \geq -300 \\
&x_{12}, x_{14}, \ldots \geq 0
\end{aligned}
$$

A constrained Minimum Cost Flow Problem

Considering the same problem as before, take into account the following additional constraints:

- Each link has a capacity, that cannot be exceeded (see the figure);
- Two additional links are available (Jacksonville-Boston and Newark-Atlanta), but BMC has to but BMC has to pay a (fixed) activation cost to use them (see the figure);
- By considering the demand of Boston fixed to 100 suppose that the number of cars allowed to pass through the city is limited to 30;
- Columbus is allowed to increase its original demand (i. e. 60), but it has to pay 500 to BMC for any additional car that it will receive (with respect to the original demand).

A constrained Minimum Cost Flow Problem



Link	Capacity
Jacksonville, Richmond	90
Jacksonville, Atlanta	180
Jacksonville, Mobile	50
Newark, Boston	50
Newark, Richmond	90
Atlanta Columbus	50
All other links	100

Link	Activation Cost
Jacksonville, Boston	300
Newark, Atlanta	800

A constrained Minimum Cost Flow Problem

- Each link has a capacity, that cannot be exceeded

$$x_{12} \leq 50$$

$$x_{14} \leq 90$$

$$x_{15} \leq 100$$

[...]

A constrained Minimum Cost Flow Problem

- Each link has a capacity, that cannot be exceeded

$$x_{12} \leq 50$$

$$x_{14} \leq 90$$

$$x_{15} \leq 100$$

[…]

```
# model
[…]
param Cap {Link};
[…]
subject to Capacity {(i,j) in Link}: Ship[i,j] <= Cap[i,j];
```

A constrained Minimum Cost Flow Problem

- Two additional links are available (Jacksonville-Boston and Newark-Atlanta), but BMC has to pay a (fixed) activation cost to use them

Variables:

$$y_{15} = \begin{cases} 1 & \text{if link from 1 to 5 is activated} \\ 0 & \text{otherwise} \end{cases} \qquad y_{72} = \begin{cases} 1 & \text{if link from 7 to 2 is activated} \\ 0 & \text{otherwise} \end{cases}$$

$$min \ldots + 25x_{72} + 25x_{15} + 300y_{72} + 800y_{15}$$

$$x_{15} \leq 100y_{15}$$

$$x_{72} \leq 100y_{72}$$

A constrained Minimum Cost Flow Problem

- Two additional links are available (Jacksonville-Boston and Newark-Atlanta), but BMC has to pay a (fixed) activation cost to use them

```
# model
[…]
set AdditionalLink  within (Link);
param CostAct {AdditionalLink};
var additionalservice {AdditionalLink} binary;
[…]
minimize Total_Cost: sum {(i,j) in Link} Cost[i,j] * Ship[i,j] + sum {(i,j) in AdditionalLink} CostAct[i,j] *additionalservice[i,j]
subject to LinkingActivation {(i,j) in AdditionalLink}:  Ship[i,j] <= Cap[i,j]*additionalservice[i,j];
```

A constrained Minimum Cost Flow Problem

- By considering the demand of Boston fixed to 100 suppose that the number of cars allowed to pass through the city is limited to 30

$$x_{12} + x_{72} \leq 130$$

A constrained Minimum Cost Flow Problem

- By considering the demand of Boston fixed to 100 suppose that the number of cars allowed to pass through the city is limited to 30

$$x_{12} + x_{72} \leq 130$$

```
# model
[…]
set Transfer  within (Cities);
[…]
subject to TransferConstr {i in Transfer}: sum {(j,i) in Link} Ship[j,i] <= DemSup[i]  + 30;
```

A constrained Minimum Cost Flow Problem

- Columbus is allowed to increase its original demand (i. e. 60), but it has to pay 500 to BMC for any additional car that it will receive (with respect to the original demand)

$$min \ldots - 500z$$

$$x_{23} + x_{53} - x_{35} = 60 + z$$

$$z \geq 0$$

A constrained Minimum Cost Flow Problem

- Columbus is allowed to increase its original demand (i. e. 60), but it has to pay 500 to BMC for any additional car that it will receive (with respect to the original demand)

```
# model
[…]
set AddDemand  within (Cities);
var additionaldemand {AddDemand} >= 0;
[…]
subject to AdditionalDemand {i in AddDemand}:
sum {(j,i) in Link} Ship[j,i] - sum {(i,k) in Link} Ship[i,k] == DemSup[i] + additionaldemand[i];
```

A constrained Minimum Cost Flow Problem

```
# model (I/II)
set Cities;
set Origins  within (Cities);
set Destinations  within (Cities);
set Transfer  within (Cities);
set AddDemand  within (Cities);
set Link  within (Cities cross Cities);
set AdditionalLink  within (Link);
param Cost {Link};
param CostAct {AdditionalLink};
param Cap {Link};
param DemSup {Cities};

var Ship {Link} >= 0;
var additionalservice {AdditionalLink} binary;
var additionaldemand {AddDemand} >= 0;
```

A constrained Minimum Cost Flow Problem

```
# model (II/II)
minimize Total_Cost:
sum {(i,j) in Link} Cost[i,j] * Ship[i,j]  + sum {(i,j) in AdditionalLink} CostAct[i,j] * additionalservice[i,j]
- sum {i in AddDemand} 500*(additionaldemand[i]);

subject to Supply {i in Origins}: - sum {(i,k) in Link} Ship[i,k] >= DemSup[i];

subject to Demand {i in Destinations}: sum {(j,i) in Link} Ship[j,i] - sum {(i,k) in Link} Ship[i,k] == DemSup[i];

subject to AdditionalDemand {i in AddDemand}:
  sum {(j,i) in Link} Ship[j,i] - sum {(i,k) in Link} Ship[i,k] == DemSup[i] + additionaldemand[i];

subject to Capacity {(i,j) in Link}: Ship[i,j] <= Cap[i,j];

subject to TransferConstr {i in Transfer}: sum {(j,i) in Link} Ship[j,i] <= DemSup[i] + 30;

subject to LinkingActivation {(i,j) in AdditionalLink}: Ship[i,j] <= Cap[i,j]*additionalservice[i,j];
```

A constrained Minimum Cost Flow Problem

```
# data (I/III)
data;

set Cities := Newark Jacksonville Boston Columbus Atlanta Richmond Mobile;
set Origins := Newark Jacksonville;
set Destinations := Boston Atlanta Richmond Mobile;
set Transfer := Boston;
set AddDemand := Columbus;

param:          DemSup:=
Newark          -200
Jacksonville    -300
Boston          100
Columbus        60
Atlanta         170
Richmond        80
Mobile          70;
```

A constrained Minimum Cost Flow Problem

```
# data (II/III)
set Link :=  (Jacksonville, Richmond)
             (Jacksonville, Atlanta)
             (Jacksonville, Mobile)
             (Jacksonville, Boston)
             (Newark, Richmond)
             (Newark, Boston)
             (Newark, Atlanta)
             (Mobile, Atlanta)
             (Atlanta, Mobile)
             (Atlanta, Columbus)
             (Atlanta, Richmond)
             (Columbus, Atlanta)
             (Boston, Columbus);

set AdditionalLink :=   (Jacksonville, Boston)
                        (Newark, Atlanta);
```

A constrained Minimum Cost Flow Problem

```
# model (III/III)
param CostAct :=
Jacksonville, Boston    300
Newark, Atlanta         800;

param:                      Cost,       Cap:=
Jacksonville, Richmond      50          90
Jacksonville, Atlanta       45          180
Jacksonville, Mobile        50          50
Jacksonville, Boston        50          100
Newark, Richmond            40          90
Newark, Boston              30          50
Newark, Atlanta             40          100
Mobile, Atlanta             25          100
Atlanta, Mobile             35          100
Atlanta, Columbus           40          50
Atlanta, Richmond           30          100
Columbus, Atlanta           35          100
Boston, Columbus            50          100;
```

A constrained Minimum Cost Flow Problem

```
# run
reset;
model SND.mod;
data SND.dat;
option solver cplexamp;
solve;
display Ship;
display additionaldemand;
display additionalservice;
```

```
# solution  (I/II)

ampl: include SND.run;
CPLEX 12.6.1.0: optimal integer solution; objective 16950
8 MIP simplex iterations
0 branch-and-bound nodes
additionalservice :=
Jacksonville Boston    1
Newark       Atlanta   1;
```

```
# solution (II/II)

Ship :=
Atlanta      Columbus    50
Atlanta      Mobile      20
Atlanta      Richmond     0
Boston       Columbus    30
Columbus     Atlanta      0
Jacksonville Atlanta    170
Jacksonville Boston      80
Jacksonville Mobile      50
Jacksonville Richmond     0
Mobile       Atlanta      0
Newark       Atlanta     70
Newark       Boston      50
Newark       Richmond    80;

additionaldemand [*] :=
Columbus  20;
```

An example of Multicommodity flows

Bavarian Motor Company (BMC) manufactures also City cars, and exports them in the U.S., in addition to the Luxury cars - as previously specified. They are 100 City cars available at the port in Newark and 50 City cars available at the port in Jacksonville. From there both Luxury cars and City cars are transported (by rail or truck) to the five distributors having a specific requirement of each type of car (in the next table the additional request of City cars is specified, whereas requirements of Luxury cars are the same as previously defined).

An example of Multicommodity flows

The problem is to determine how to transport both types of cars along the arcs of the network to satisfy the demands at a minimum cost, considering that for marketing reasons:

- City cars cannot use the link between Jacksonville and Atlanta;
- The number of City cars traveling from Atlanta to Columbus has to be at least 30% higher than the number of cars traveling from Boston to Columbus.

City	City cars
Boston	25
Columbus	35
Atlanta	40
Richmond	50
Mobile	0
Jacksonville	-50
Newark	-100

AMPL Main Commands:

- reset; # reset the environment
- model *modelfilename*.mod; # model upload
- data *datafilename*.dat; # data upload
- option solver *nameofsolver*; # optimizer selection
- solve; # solve
- display *nameofvariables*; # display variables