Course on mathematical modelling: AMPL and CPLEX

teacher: Giacomo Lanza

Dipartimento di Informatica, Università di Pisa

a.a. 2019-2020

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

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

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. There 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 (costs of transportation of city cars are the same as for luxury cars), 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

An example of Multicommodity flows

$$
\begin{aligned}
min \quad & 30x^1_{12} + 40x^1_{14} + 50x^1_{23} + 35x^1_{35} + 40x^1_{53} + 30x^1_{54} + \\
& + 35x^1_{56} + 25x^1_{65} + 50x^1_{74} + 45x^1_{75} + 50x^1_{76} + 30x^2_{12} + \\
& + 40x^2_{14} + 50x^2_{23} + 35x^2_{35} + 40x^2_{53} + 30x^2_{54} + 35x^2_{56} + \\
& + 25x^2_{65} + 50x^2_{74} + 45x^2_{75} + 50x^2_{76}
\end{aligned}
$$

$$
\begin{aligned}
& -x^1_{12} - x^1_{14} \geq -200 \\
& x^1_{12} - x^1_{23} = 100 \\
& x^1_{23} + x^1_{53} - x^1_{35} = 60 \\
& x^1_{14} + x^1_{54} + x^1_{74} = 80 \\
& x^1_{35} + x^1_{65} + x^1_{75} - x^1_{53} - x^1_{54} - x^1_{56} = 170 \\
& x^1_{56} + x^1_{76} - x^1_{65} = 70 \\
& -x^1_{74} - x^1_{75} - x^1_{76} \geq -300
\end{aligned}
$$

$$
x^1_{12}, x^1_{14}, \ldots \geq 0
$$

$$
\begin{aligned}
& -x^2_{12} - x^2_{14} \geq -100 \\
& x^2_{12} - x^2_{23} = 25 \\
& x^2_{23} + x^2_{53} - x^2_{35} = 35 \\
& x^2_{14} + x^2_{54} + x^2_{74} = 50 \\
& x^2_{35} + x^2_{65} + x^2_{75} - x^2_{53} - x^2_{54} - x^2_{56} = 40 \\
& x^2_{56} + x^2_{76} - x^2_{65} = 0 \\
& -x^2_{74} - x^2_{75} - x^2_{76} \geq -50
\end{aligned}
$$

$$
x^2_{12}, x^2_{14}, \ldots \geq 0
$$

An example of Multicommodity flows

```
set Cities;
set Origins  within (Cities);
set Destinations  within (Cities);
set Product;
set Link  within (Cities cross Cities);

param Cost {Link};
param DemSup {Cities, Product};

var Ship {Link, Product} integer >= 0;

minimize Total_Cost: sum {(i,j) in Link, k in Product} Cost[i,j] * Ship[i,j,k];

subject to Supply {i in Origins, k in Product}: - sum {(i,l) in Link} Ship[i,l, k] >= DemSup[i,k];
subject to Demand {i in Destinations, k in Product}:
sum {(j,i) in Link} Ship[j,i,k] - sum {(i,l) in Link} Ship[i,l,k] == DemSup[i,k];
```

An example of Multicommodity flows

```
set Product:= luxury citycar ;

param DemSup:
                    luxury    citycar:=
Newark              -200      -100
Jacksonville        -300      -50
Boston              100       25
Columbus            60        35
Atlanta             170       40
Richmond            80        50
Mobile              70        0;
```

An example of Multicommodity flows

city cars cannot use the link between Jacksonville and Atlanta;

```
subject to rule1: Ship['Jacksonville', 'Atlanta', ‘citycar'] = 0;
```

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.

```
subject to rule2: 1.3*Ship['Boston', 'Columbus', ‘citycar'] <= Ship['Atlanta', 'Columbus', ‘citycar’];
```

An example of Multicommodity flows

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

```
ampl: include MK.run;
CPLEX 12.6.1.0: optimal integer solution; objective 32975
7 MIP simplex iterations
0 branch-and-bound nodes
Ship [*,*,citycar]
:          Atlanta Boston Columbus Mobile Richmond    :=
Atlanta        .      .      33       0       0
Boston         .      .      25       .       .
Columbus      23      .       .       .       .
Jacksonville   0      .       .      50       0
Mobile        50      .       .       .       .
Newark         .     50       .       .      50;
 [*,*,luxury]
:          Atlanta Boston Columbus Mobile Richmond    :=
Atlanta        .      .      40       0       0
Boston         .      .      20       .       .
Columbus       0      .       .       .       .
Jacksonville 210      .       .      70       0
Mobile         0      .       .       .       .
Newark         .    120       .       .      80;
```

An example of Assignment Problem

A carrier needs to organize a special shipment to one of his clients. The client needs the transportation of 15 different objects, each one having a volume and a weight. The carrier has 4 trucks available. Each truck has an operation cost, a maximum loading weight and its trailer can be loaded till a maximum volume. The carrier needs to decide how many trucks to use to transport all the objects requested, by considering weight and volume constraints, at a minimum cost. In addition:

- objects 1 and 2 need to be shipped together;
- objects 10 and 11 cannot be shipped together.

An example of Assignment Problem

Object	Weight	Volume
1	10	260
2	24	140
3	18	190
4	7	220
5	7	180
6	6	250
7	16	170
8	11	200
9	8	140
10	11	170
11	1	190
12	6	230
13	15	230
14	8	180
15	2	250

Truck	Cost	Weight	Volume
1	900	45	1000
2	1000	50	1000
3	1200	60	1000
4	1300	80	1000

An example of Assignment Problem

Input data:

- I = set of trucks, indexed by i;
- J = set of objects, indexed by j;
- c_i operation cost of truck $i = 1, ..4$;
- w'_i maximum loading weight of truck $i = 1, ..4$;
- v'_i maximum loading volume of truck $i = 1, ..4$;
- w_j weight of object $j = 1, ..15$;
- v_j volume of object $j = 1, ..15$;

An example of Assignment Problem

$$x_{ji} = \begin{cases} 1 & \text{if object j is loaded on truck i} \\ 0 & \text{otherwise} \end{cases}$$

$$j = 1, ..15; \; i = 1, ..4$$

$$y_i = \begin{cases} 1 & \text{if truck i is selected} \\ 0 & \text{otherwise} \end{cases}$$

$$i = 1, ..4$$

An example of Assignment Problem

$$min \sum_{i=1,..4} c_i \cdot y_i$$

Minimize the total cost of operating trucks

$$\sum_{i=1,..4} x_{j,i} = 1 \qquad \forall\, j = 1,..15;$$

Each object (j from 1 to 15) has to be loaded on a truck

$$\sum_{j=1,..15} x_{j,i} \leq M \cdot y_i \qquad \forall\, i = 1,..4;$$

Objects may be loaded only on selected trucks (M can be set to 15). Note that this set of constraints is redundant because of capacity and volume constraints (see next slide).

An example of Assignment Problem

$$\sum_{j=1,..15} w_j x_{j,i} \leq w'_i y_i \qquad \forall\, i = 1,..4;$$

Maximum loading weight of trucks cannot be violated

$$\sum_{j=1,..15} v_j x_{j,i} \leq v'_i y_i \qquad \forall\, i = 1,..4;$$

Maximum loading volume of trucks cannot be violated

$$y_i \in \{0,1\} \qquad \forall\, i = 1,..4;$$

$$x_{j,i} \in \{0,1\} \qquad \forall\, j = 1,..15;\ i = 1,..4;$$

Variables are binary

An example of Assignment Problem

$$x_{1,i} = x_{2,i} \qquad \forall\, i = 1, ..4;$$

Objects 1 and 2 need to be shipped together

$$x_{10,i} + x_{11,i} \leq 1 \qquad \forall\, i = 1, ..4;$$

Objects 10 and 11 cannot be shipped together

An example of Assignment Problem

```
# model (I/II)
param MaxTrucks;
param MaxObjects;
set Objects:= 1.. MaxObjects;                    # index j
set Trucks:= 1..MaxTrucks;                       # index i
param Cost {Trucks} ;
param MaxWeight {Trucks};
param Weight {Objects} ;
param Volume {Objects} ;
param MaxVolume;

var AssignmentOT {Objects, Trucks} binary;
var AssignmentT {Trucks} binary;
```

An example of Assignment Problem

```
# model (I/II)
param MaxTrucks;
param MaxObjects;
set Objects:= 1.. MaxObjects;                    # index j
set Trucks:= 1..MaxTrucks;                       # index i
param Cost {Trucks}              >= 0;
param MaxWeight {Trucks}         > 0;
param Weight {Objects}           > 0 ;
param Volume {Objects}           > 0;
param MaxVolume                  > 0;

var AssignmentOT {Objects, Trucks} binary;
var AssignmentT {Trucks} binary;
```

An example of Assignment Problem

```
# model (I/II)
param MaxTrucks;
param MaxObjects;
set Objects:= 1.. MaxObjects;                    # index j
set Trucks:= 1..MaxTrucks;                       # index i
param Cost {Trucks}             >= 0;
param MaxWeight {Trucks}        > 0;
param Weight {Objects}          > 0 ;
param Volume {Objects}          > 0;
param MaxVolume                 > 0;

var AssignmentOT {Objects, Trucks} binary;
var AssignmentT {Trucks} binary;

check: sum {j in Objects} Volume[j] <= MaxTruck*MaxVolume;
check: sum {j in Objects} Weight[j] <= sum {i in Truck} MaxWeight[i];
```

An example of Assignment Problem

```
# model (II/II)
minimize  Total_Cost: sum {i in Trucks} Cost[i] * AssignmentT[i];

subject to AssigneObjToCont {j in Objects}: sum {i in Trucks} AssignmentOT[j,i] == 1;

subject to LinkingObjcont {i in Trucks}: sum {j in Objects} AssignmentOT[j,i] <= MaxObjective*AssignmentT[i];

subject to CapacityW {i in Trucks}: sum {j in Objects} Weight[j]*AssignmentOT[j,i] <= MaxWeight[i];

subject to CapacityV {i in Trucks}: sum {j in Objects} Volume[j]*AssignmentOT[j,i] <= MaxVolume;

subject to rule1 {i in Trucks}: AssignmentOT[1,i] == AssignmentOT[2,i];

subject to rule2 {i in Trucks}: AssignmentOT[10,i] + AssignmentOT[11,i] <= 1;
```

An example of Assignment Problem

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

param MaxTrucks := 4;
param MaxObjects:= 15;
param MaxVolume := 1000;

param:    Cost, MaxWeight:=
1         900       45
2         1000      50
3         1200      60
4         1300      80;
```

```
# data (II/II)
param:    Weight, Volume:=
1         10        260
2         24        140
3         18        190
4         7         220
5         7         180
6         6         250
7         16        170
8         11        200
9         8         140
10        11        170
11        1         190
12        6         230
13        15        230
14        8         180
15        2         250;
```

An example of Assignment Problem

```
# run
reset;
model Assign.mod;
data Assign.dat;
option solver cplex amp;
solve;

display AssignmentT;
display AssignmentOT;
```

```
# solution (I/II)
AssignmentT [*] :=
1  1
2  1
3  1
4  0;
```

```
# solution (II/II)
AssignmentOT [*,*]
:   1   2   3   4   :=
1   0   0   1   0
2   0   0   1   0
3   0   1   0   0
4   1   0   0   0
5   1   0   0   0
6   0   1   0   0
7   0   0   1   0
8   1   0   0   0
9   0   1   0   0
10  1   0   0   0
11  0   1   0   0
12  1   0   0   0
13  0   1   0   0
14  0   0   1   0
15  0   0   1   0;
```

Modelling commands: Write on File

```
# run
reset;
model Assign.mod;
data Assign.dat;
option solver cplex amp;
solve;

display AssignmentT > filename.txt;
display AssignmentOT > filename.txt;
```

Modelling commands

```
# run
reset;
model Assign.mod;
data Assign.dat;
option solver cplex amp;

for {a in 1..2} {
printf "Test \n" > filename.txt;
display a > filename.txt;
solve;
display AssignmentT > filename.txt;
}
```

```
Test
a = 1
AssignmentT [*] :=
1  1
2  1
3  1
4  0
;

Test
a = 2
AssignmentT [*] :=
1  1
2  1
3  1
4  0
;
```

Modelling commands

```
# run
reset;
model Assign.mod;
data Assign.dat;
option solver cplex amp;

for {a in 1..2} {
printf "Test \n" > filename.txt;
display a > filename.txt;

if a == 2 then {
[...]
}
solve;
display AssignmentT > filename.txt;
}
```

Modelling commands

- The **let** command permits you to **change particular data values** while leaving the model the same (it is more convenient for small or easy-to-describe changes than reset data or update data)
- The **drop** command instructs AMPL to **ignore certain constraints** or objective of the current model
- The **fix** command **fixes specified variables at a specified values** (as if there was a constraint imposing that the specified variables must be equal to that values)

Modelling commands

```
# run (I/II)
reset;
model Assign.mod;
data Assign.dat;
option solver cplex amp;

for {a in 1..2} {
    printf "Test \n" > filename.txt;
    display a > filename.txt;
        if a == 2 then {
            let MaxVolume := 800;
            let Cost[1] := 1400;
            drop CapacityW;
            fix AssignmentOT[9,1] := 1
        }
```

```
# run (II/II)
    solve;

    display MaxVolume > filename.txt;
    display Cost[1] > filename.txt;

    display AssignmentT > filename.txt;
    display AssignmentOT > filename.txt;
}
```

Modelling commands

```
# filename.txt

Test
a = 1
MaxVolume = 1000
Cost[1] = 900
AssignmentT [*] :=
1  1
2  1
3  1
4  0;
```

```
 filename.txt

Test
a = 2
MaxVolume = 800
Cost[1] = 1400
AssignmentT [*] :=
1  1
2  1
3  1
4  1;
```

Modelling commands

A comprehensive guide to building optimization models, for beginning or experienced users:

https://ampl.com/resources/the-ampl-book/

Written by the creators of AMPL, this book is a complete guide for modellers at all levels of experience.

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