AMPL

E' un linguaggio che trasforma il problema in una formulazione che passa ad un solver (programma che ha codificati tutti gli algoritmi che conosciamo per risolvere il problema primale-duale, del simplesso,...) il quale risolve il problema e passa la sol. ad AMPL

$$\min\ c^T x \quad Ax \geq b \quad x \geq 0 \;\Rightarrow\; \boxed{\text{AMPL}} \;\rightleftarrows\; \boxed{\text{SOLVER}}$$

COMMUNITY EDITION (AMPL)

SOLVER: CBC (OPEN SOURCE GRATUITO), GUROBI, CPLEX

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

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

$$x_j \geq 0 \qquad j \in 1,\dots,n$$

MODELLO

Bisogna fare una separazione logica tra modello e dati:

DATI: $c = [c_1 \dots c_n]$

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

$$b = [b_1 \dots b_m]^T$$

Il modello va in un file con estensione .mod, mentre i dati vanno in un file con estensione .dat.

file model.mod:

Scrivo i param (verranno poi quantificati nel file .dat)

```
param n;
param m;    RANGE (INTERVALLO)
param c {1...n};
param b {1...m};
param a {1...m, 1...n};
```

→ AMPL E' UN LINGUAGGIO CASE SENSITIVE ($a \neq A$)

scrivo le variabili x indicizzate sul range 1,...,m.

```
var x {1,..,m} >=0
```

scrivo la funzione obiettivo e gli do il senso di ottimizzazione (min/max)

```
minimize z: sum {J in 1...m} c[J]*x[J];
```

↳ ESPRESSIONE LINEARE (SOMMATORIA TRA VAR E PARAM)

scrivo le espressioni dei vincoli: ci conviene dargli dei nomi per poi vedere se sono saturi o meno o per vedere la var duale corrispondente

```
s.t. v{i in 1...m}: sum {J in 1...m} A[i,J]*[J]>=b[i];
```

file model.dat: SUPPONENDO 3 VAR E 2 VINCOLI

```
param n := 3;
param m := 2;
```

Il range, in realtà, e' un insieme di numeri, quindi si può fare un set (insieme) $J = \{1 \dots n\}$ e $I = \{1 \dots m\}$ e modificare il problema in:

$$\min \sum_{j \in J} c_j x_j$$

$$\text{s.t.} \sum_{j \in J} a_{ij} x_j \geq b_i \quad i \in I$$

$$x_j \geq 0 \quad j \in J$$

e quindi nel file mod:

```
set I;
set J;
param c {J};
param b {I};
param {I,J};
var x {J} >=0;
minimize z: sum {J in J} c[J]*x[J];
s.t. v{i in I}: sum {J in J} a[i,J]*x[J]>= b[i];
```

e quindi model dat:

```
set I = r1 r2;    RIGHE MATRICE
set J = x1 x2 x3;    VARIABILI
param b: =
r1  2
r2  3;
```

param c :=
x1 4
x2 2
x3 3;
param a :
 x1 x2 x3 :=
r1 4 3 1
r2 -4 3 -1;

Nel caso in cui, ad esempio, il secondo vincolo sia $\leq$ e non $\geq$ posso moltiplicare per -1 i coeff, oppure

```
set I1 = r1;
set I2 = r2;
```
} model.dat

```
s.t. v1 {i in I1} : sum {j in J} a[i,j]* x[j] >= b[i];
s.t. v2 {i in I2} : sum {j in J} a[i,j]* x[j] <= b[i];
```
} model.mod

La stessa cosa può valere per J:

```
var x {J1} >= 0;
var x {J2} <= 0;
```
} model.mod : IN MODEL.DAT : set J1; set J2;

Per caricare su AMPL il modello si scrive:

```
model es/model.mod;
```
es/ → CARTELLA/PERCORSO IN CUI C'È IL FILE

```
data es/model.dat;
option solver cplex;
```
PER SCEGLIERE IL SOLVER

`solve;` PER FAR RISOLVERE IL PROBLEMA

`reset;` PER RINIZIALIZZARE IL TUTTO DOPO UN ERR (DEVO CARICARE DI NUOVO IL MODELLO,...)

Per interrogare la sol:

`display x;` MI DÀ IL VALORE DEL VETT. x

`display x[x1];` PERCHÉ VOGLIO LA 1° VAR DEL VETT. x CHE È INDICIZZATO DA J E J HA COME 1° COMP. x1.

`display v[r1];`

`display z;`

∃ un 3° tipo di file che può trattare AMPL: il file dei comandi. Queste istruzioni le posso inserire in un file model.run e faccio eseguire direttamente questo file piuttosto che scrivere tutte le istruzioni scrivendo:

`model run;`

`commands model.run;` CARICA IL FILE E LO ESEGUE es.①:

Dato il seguente problema:

$\min \; 4x_1 + 2x_2 - 3x_3$

s.t.

$r_1 \quad 2x_1 + 3x_2 + x_3 \leq 6$

$r_2 \quad -4x_1 + 3x_2 - x_3 \leq 12$

$x_1 \leq 0, \; x_2 \geq 0, \; x_3 \geq 0$

• es. mod :

set J_1; VARIABILI ≤ 0 OSSIA x_1

set J_2; VARIABILI ≥ 0 OSSIA x_2, x_3

set J; INSIEME VARIABILI : x_1, x_2, x_3

set I; INSIEME VINCOLI SOLO UNO XK ≤ 0

param alfa;

param b{I}; COEFF. DEI TERMINI NOTI (OSSIA 6 e 12)

param c{J}; COEFF. F.O. OSSIA (4, 2, -3)

param a{I,J}; COEF. DELLE VARIABILI NEI VINCOLI CIOÈ $\begin{array}{ccc} 2 & 3 & 1 \\ -4 & 3 & -1 \end{array}$

var x{J}; → VARIABILI F.O. (SOLO LA x)

minimize z: sum{j in J} c[j]* x[j]; → F.O

s.t. v1 {i in I}: sum{j in J} a[i,j]* x[j] <= b[i]+alfa; → VINCOLI (SOLO 1 PERCHÈ ENTRAMBI $\leq$)

s.t. v2 {j in J1}: x[j] <=0; → $x_m \leq 0$ (ossia $x_1 \leq 0$)

s.t. v3 {j in J2}: x[j] >=0; → $x_m \geq 0$ (ossia $x_2, x_3 \geq 0$)

• es. dat:

set J_1 := x_1; $x_1 \leq 0$

set J_2 := x_2, x_3; $x_2, x_3 \geq 0$

set J := x_1, x_2, x_3; tutte le variabili di f.o.

set I := r_1, r_2; righe 2 perchè 2 VINCOLI

param alfa := 5; A SCELTA.

param b := TERMINI NOTI VINCOLI

r_1 6

r_2 12;

```
param c: =          COEFF. F.O.
x1  4
x2  2
x3  -3;
param a:
      x1   x2   x3 =      COEF. VINCOLI.
r1    2    3    1
r2   -4    3   -1 ;
```

es.run:

```
model es1.mod;
data es1.dat;
option solver cplex;
for {i in 1..10} {
    solve;
    let alfa := i;
    display z;          MOSTRA F.O.
    display x;          MOSTRA SOL. OTTIMA
    display v1;         MOSTRA VINCOLI
    display v1.slack;   MOSTRA VARIABILI DI SLACK
};
```

es. 2

Dato il seguente problema:

$$\min \; 2x_1 + x_2 + 3x_4 - x_5 + 4x_6 + x_7 + x_8$$

s.t.

$$x_1 - x_2 - x_3 + x_8 \leq 80$$

$$x_4 + x_5 + x_6 - x_7 \leq 60$$

$$x_1 + x_4 \geq 60$$

$$x_2 + x_5 = 40$$

$$x_3 + x_6 \geq 20$$

$$x_1, x_2, x_4, x_6, x_7, x_8 \geq 0; \; x_5 \leq 0; \; x_3 \text{ libero}$$

·es2.mod:

```
set J1;
set J2;
set J3;
set I1; VINCOLI CON <=
set I2; VINCOLI CON >=
set I3; VINCOLO CON =

param b1 {I1};
param b2 {I2};
param b3 {I3};
param c {J};
param a1 {I1, J};
param a2 {I2, J};
param a3 {I3, J}
var x {J};
minimize z: sum {j in J} c[j]*x[j];
s.t. v1 {i in I1}: sum {j in J} a1[i,j]*x[j] <= b1[i];
s.t. v2 {i in I2}: sum {j in J} a2[i,j]* x[j] >= b2[i];
s.t. v3 {i in I3}: sum {j in J} a3[i,j]*x[j] == b3[i];
s.t. vJ1 {j in J1}: x[j]>=0;
s.t. vJ2 {j in J2}: x[j]<=0;
```

·es2.dat:

```
set J1 := x1, x2, x4, x6, x7, x8;
set J2 := x5;
set J := x1, x2, x3, x4, x5, x6, x7, x8;
set I1 := r1, r2;
set I2 := r3, r5;
set I3 := r4;

param b1 :=
r1    80
r2    60;
```

```
param b2 :=
r3   60
r5   20;
param b3 :=
r4   40;
param c :=
x1  2
x2  1
x3  0
x4  3
x5  -1
x6  4
x7  1
x8  1;
param a1 :
      x1   x2   x3   x4   x5   x6   x7   x8 :=
r1    1    -1   -1   0    0    0    0    1
r2    0    0    0    1    1    1    -1   0;
param a2 :
      x1   x2   x3   x4   x5   x6   x7   x8 :=
r3    1    0    0    1    0    0    0    0
r5    0    0    1    0    0    1    0    0;
param a3 :
      x1   x2   x3   x4   x5   x6   x7   x8 :=
r4    0    1    0    0    1    0    0    0;
```

es2.run:

```
model es2.mod;
data es2.dat;
option solver cplex;
expand > es2.lp;
solve;
display z;
display x;
```

es. 3

	STD	HIQ	MAX
ASM	3	2	80
TST	2	4	60
MAT	4	4	70
P	10	15	

min. $10x_1 + 15x_2$

s.t.

$$3x_1 + 2x_2 \leq 80$$
$$2x_1 + 4x_2 \leq 60$$
$$4x_1 + 4x_2 \leq 70$$
$$x_1, x_2 \geq 0$$

es3. mod:

```
set J;

set I;

param b{I};
param c{J};
param a{I,J};
var x{J};

maximize z: sum{J in J} c[J]*x[J];
s.t. v{i in I} a[i,J]*x[J] <= b[i];
s.t. v1{J in J} x[J] >= 0;
```

es3. dat:

```
set J := x1, x2;

set I := r1, r2, r3;
```

```
param b :=
r1   80
r2   60
r3   70;
param c :=
x1   10
x2   15;
param a:
        x1   x2
  r1    3    2
  r2    2    4
  r3    4    4;
```

es3.run:

```
model es3.mod;
data es3.dat;
option solver cplex;
solve;
display z;
display x;
display v;
display v.slack;
```