

Riferimento rapidissimo alla sintassi AMPL

Due file:

- `.mod` file: contiene la descrizione del problema in un programma matematico *puramente simbolico*, i.e. senza dati numerici
- `.dat` file: contiene tutti i dati numerici che descrivono la specifica *istanza* del problema che vogliamo risolvere

Model file

Contiene definizione di *indici*, *parametri*, *variabili di decisione*, *funzione obiettivo*, *vincoli*.

1. Insiemi degli indici: contengono gli indici su cui variabili e parametri sono definiti. Esempio: se la variabile x è in \mathbb{R}^n , per un dato n , l'insieme I conterrà tutti gli indici per cui x è definita (ad esempio i naturali da 1 a 5). Il contenuto di I *non viene specificato* nel file di modello. Keyword: `set`. Sintassi:

```
Set I;
```

2. Parametri: tutto ciò che fa parte del problema ma non è una variabile di decisione. Keyword: `param`.

Se il parametro è uno scalare:

```
param myScalar.
```

Se è un vettore o una matrice, è necessario specificare gli insiemi di indici su cui corre, i.e.

```
param myVector{I};  
param myMatrix{I, J};
```

Nell'ultimo caso, `myMatrix` viene definita per ogni indice nel prodotto cartesiano $I \times J$.

3. Variabili di decisione. Keyword: `var`. Gli indici su cui la variabile è definita vengono specificati come per i parametri. Sintassi:

```
var myVar{I};
```

Positività, negatività, vincoli di box (ossia lower e upper bound), integralità o binarietà della variabile vengono imposti a seguito, come nell'esempio:

```
var myVar{I} >= 0;  
var myVar{I} >= 5, <= 18;  
var myVar{I,J}, integer;  
var myVar{I,J}, binary;  
var myVar{I,J} >= 0, integer;
```

4. Funzione obiettivo. Keyword: `minimize` o `maximize`. Esempio:

```
minimize myFunction: sum{i in I} myVar[i]*myVector[i];
```

NOTA: I “:” precedono, sia per vincoli che per funzione obiettivo, la definizione algebrica degli stessi.

NOTA: ogni funzione (vincolo o funzione obiettivo) in AMPL deve avere un nome *unico*.

NOTA: E’ buona regola, se vogliamo scrivere un modello di *basso livello*, ossia simile alla sua definizione algebrica, utilizzare lettere *MAIUSCOLE* per gli insiemi di indici, i.e. J , e *minuscole*, i.e. j per gli indici che su esso corrono. SI NOTI CHE nulla vieta l’uso di, ad esempio, $j \text{ in } I$ o $i \text{ in } J$.

5. Vincoli. Keyword: `subject to`. È possibile ripeterla prima di ogni vincolo o solo una volta all’inizio di questi.

Se vogliamo esprimere una famiglia di vincoli, ossia un insieme di vincoli in bi-iezione con un insieme di indici, tale insieme viene specificato tra `{ }` dopo il nome del vincolo. Lo specifico indice appartenente all’insieme che verrà utilizzato nella definizione algebrica del vincolo stesso va indicato con l’usuale sintassi $j \text{ in } J$, dove j è l’indice e J l’insieme su cui esso corre. Esempio:

```
subject to myConstraint{j in J}: sum{i in I}myMatrix[i,j]*x[i] >= 0;
```

Data file

1. Insiemi degli indici. Esempio:

```
set I := Gennaio Febbraio Marzo;
set I := 1 .. 10;
```

2. Parametri scalari:

```
param myScalar := 100;
```

3. Parametri vettoriali:

```
param myVector1 :=
Gennaio 12
Febbraio 31
Marzo 51;
```

NOTA: Gennaio, Febbraio, Marzo sono *indici*!

4. Parametri vettoriali, definendo due vettori alla volta:

```
param: myVector1 myVector2 :=
Gennaio 12 0.1
Febbraio 31 0.15
Marzo 51 0.98;
```

NOTA: osservare la presenza dei “:”.

5. Matrici:

```
param myMatrix:
      Rossi   Bianchi :=
Gennaio  0.12   0.31
Febbraio  0.51   0.41
Marzo     0.61   0.98;
```

NOTA: osservare la diversa posizione dei “:” rispetto al caso precedente.

Chiamare AMPL

Con MS Windows XP/Vista è possibile: fare doppio click su `ampl.exe`, oppure premere [WINDOWS]+r, digitare `cmd` o `cmd.exe`, navigare fino alla cartella di AMPL, ad esempio con

```
c:
cd USER_DATA
cd ampl
ampl
```

Nota: il comando `cd` ha semantica *change directory*. È seguito dal nome della directory, contenuta in quella corrente, a cui si vuole accedere.

NOTA: la seconda opzione è preferibile: il prompt di comandi (`cmd.exe`) tiene in memoria i comandi digitati *anche all'interno di AMPL*. Se dovete chiudere AMPL e farlo ripartire, potete riottenere i comandi digitati in precedenza mediante i tasti FRECCIA SU e FRECCIA GIÙ della tastiera.

Caricare .mod e .dat, chiamare il risolutore

Sintassi di chiamata:

```
model mymodel.mod;
data mymodel.dat;
option solver cplex;
solve;
display x;
```

La stringa `option solver cplex` obbliga AMPL ad utilizzare il risolutore CPLEX (di ILOG) anziché MINOS, chiamato di default. A differenza di MINOS, CPLEX permette di risolvere efficientemente problemi di programmazione lineare intera (oltre che continua).

La stringa `display x`; stampa il valore della variabile `x` in corrispondenza della soluzione trovata. Nel caso ci fossero più variabili di decisione, ad esempio `x` e `y`, usare semplicemente `display x; display y`;

File `.run`: E' consigliabile, per comodità, creare un file `.run` (ad esempio `myproblem.run`) contenente i comandi di chiamata sopra indicati. Sarà possibile trasmettere tali comandi ad AMPL mediante la stringa

```
type myproblem.run | ampl
```