

# Lezioni su AMPL & CPLEX

Corso di Ricerca Operativa · Prof. Gianpaolo Oriolo

Gianmaria Leo

Università di Roma “Tor Vergata”

17 Ottobre 2014

# Informazioni generali (Updated)

**Contatti:** Gianmaria Leo ([gianmaria.leo@optrail.com](mailto:gianmaria.leo@optrail.com))  
Prof. Gianpaolo Oriolo ([oriolo@disp.uniroma2.it](mailto:oriolo@disp.uniroma2.it))

**Lezioni:** venerdì, ore 16:00 - 17:30, aula B4

- ① Presentazioni
- ② Esercitazioni al computer

**Materiale:** Inviare una mail a [ricerca14.15@gmail.com](mailto:ricerca14.15@gmail.com)

**Esame:** In fase di definizione:

- ① 6 CFU: Prova al computer
- ② 9 CFU: Progetto (Modeling/Computational Challenge)

## Nella puntata precedente... (1/3)

- lanciare il programma
- specificare un solutore
- dichiarare le variabili
- dichiarare la funzione obiettivo
- dichiarare i vincoli
- risolvere
- visualizzare la soluzione

```
ampl: option solver <path2solver>;  
ampl: var <variable>;  
ampl: minimize <objFunc_label>: <objFunc_expr>;  
ampl: subject to <constr_label>: <constr_expr>;  
ampl: solve;  
ampl: display <var_1>, <var_2>, <...>;  
ampl: quit;
```

Formulazione di un semplice problema di produzione:

$$\max \quad 25x + 30y$$

s.t.

$$\frac{1}{200}x + \frac{1}{140}y \leq 40$$

$$0 \leq x \leq 6000$$

$$0 \leq y \leq 4000$$

## Nella puntata precedente... (3/3)

- File: example1.txt:

```
option solver 'minos/minos';  
var x;  
var y;  
maximize profitto: 25*x + 30*y;  
subject to tempo: (1/200)*x + (1/140)*y <= 40;  
subject to limite_barre: 0 <= x <= 6000;  
subject to limite_bobine: 0 <= y <= 4000;  
solve;  
display x, y;  
display profitto;  
quit;
```

- Comando:

```
./bin/ampl < examples/example1.txt > logs/ampl_ex1.log
```

# Le applicazioni reali

- Nel caso di applicazioni reali, i modelli hanno generalmente grandi dimensioni (variabili, vincoli, parametri)
- Usare AMPL nel modo appena visto è impensabile perché:
  - il modello è illeggibile
  - risolvere più istanze associate a uno stesso modello non è agevole
  - il codice per generare i comandi AMPL è difficile da gestire
  - una semplice esperienza computazionale richiede uno sforzo non trascurabile

## Soluzione

### La metodologia AMPL

## ① Definire il modello:

- Dichiarare gli elementi del modello
- Definire le relazioni che gli elementi del modello devono soddisfare
- Definire le variabili
- Riportare funzione obiettivo e vincoli

## ② Definire i dati:

- Assegnare un valore a ciascun elemento definito nel modello
- Strutturare i dati rispetto alla loro definizione

## ③ Gestire l'esperienza computazionale:

- Definire il/i solutore/i da utilizzare
- Definire il tuning del/dei solutore/i
- Caricare una o più istanze da risolvere (modello + dati)
- Lanciare l'ottimizzazione
- Definire l'output desiderato

# Esempio: un modello di PL

- Il seguente modello di programmazione lineare ha  $n$  variabili e  $m$  vincoli:

$$\begin{aligned} \min \quad & \sum_{i=1}^n c_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^n a_{ij} x_i = b_j & \forall j = 1, \dots, m \\ & x_i \geq 0 & \forall i = 1, \dots, n \end{aligned}$$

- La dimensione del problema dipende da  $n$  e  $m$



# Passo 1: Definire il modello

- AMPL permette di definire un modello in modo indipendente da dimensioni e dati del problema
- Questo è possibile mediante l'uso dei files `.mod`
- Un singolo file `.mod` può essere utilizzato per trattare istanze diverse dello stesso modello

# Struttura del file .mod

Il file .mod è un file di testo che può essere strutturato nelle seguenti parti:

- Dichiarazione degli insiemi: `set`
- Dichiarazione dei parametri: `param`
- Dichiarazione delle variabili: `var`
- Verifica di condizioni su insiemi, parametri e variabili: `check`
- Definizione della funzione obiettivo:  
`minimize <objFunc_label>: <objFunc_expr>`
- Definizione di vincoli indicizzati:  
`subject to <constr_label> <elem_in_sets>: <constr_expr>`

## Esempio di file .mod (lp\_1.mod)

```
set P;  
set V;  
  
param c{P};  
param a{P,V};  
param b{V};  
  
var x{P};  
  
minimize obj: sum{i in P} c[i] * x[i];  
  
subject to vin1 {j in V}:  
    sum{i in P} a[i,j] * x[i] = b[j];  
  
subject to vin2 {i in P}: x[i] >= 0;
```

# Esempio: un'istanza del modello di PL

- Consideriamo la seguente istanza con  $n = 3$ ,  $m = 2$ :

$$\begin{aligned}\min \quad & x_1 + 2x_2 - 3x_3 \\ & 2x_1 - 2x_2 + 5x_3 = 0 \\ & -2x_1 + 6x_2 + 3x_3 = 1 \\ & x_1, x_2, x_3 \geq 0\end{aligned}$$

## Passo 2: Definire i dati

- AMPL permette di definire i dati del problema esternamente al modello
- Questo è possibile grazie all'ausilio dei files `.dat`
- I dati contenuti nel file `.dat` devono essere coerenti con il contenuto del file `.mod` associato
- Un'opportuna coppia (`.mod`, `.dat`) specifica un'istanza del modello

# Struttura del file .dat

Il file .dat è un file di testo strutturato nelle seguenti parti:

- Assegnamento di elementi agli insiemi: operatore :=
- Assegnamento di valori ai parametri: operatori :, :=

Logica dell'assegnamento:

```
set <set> := <el1> <el2> <...>
```

```
param <par>:      <c1>      <c2>  <...> :=  
    <r1>  <val:r1,c1>  <val:r1,c2>  <...>  
    <r2>  <val:r2,c1>  <val:r1,c2>  <...>  
    <...>
```

## Esempio di file .dat (lp\_1.dat)

```
set P := elem1 elem2 elem3;
```

```
set V := req1 req2;
```

```
param a:  req1  req2:=
```

```
elem1      2    -2
```

```
elem2     -2     6
```

```
elem3      5     3;
```

```
param:  b:=
```

```
req1      0
```

```
req2      1;
```

```
param:  c:=
```

```
elem1      1
```

```
elem2      2
```

```
elem3     -3;
```

## Passo 3: Esperienza computazionale

- Abbiamo visto che la risoluzione di un problema con AMPL richiede alcuni comandi (`model`, `data`, `option`, `solve`, `display`, ...)
- L'esperienza computazionale potrebbe richiedere la risoluzione di diversi problemi e/o l'uso di diversi solutori
- In generale, potremmo essere interessati a definire un particolare algoritmo/procedura/schema basato sulla risoluzione di una sequenza di problemi di programmazione matematica
- AMPL permette di gestire questa fase mediante l'ausilio dei files `.run`



# Nozioni sul file .run

- Il file `.run` è uno script che permette di eseguire batch di comandi AMPL
- Per eseguirlo, basta aprire un terminale e inviare il comando:  
`ampl <run_file_name>.run`
- In generale, esso contiene
  - caricamento del modello: `model <model_file_name>.mod`
  - caricamento dei dati: `data <data_file_name>.dat`
  - definizione del solutore: `option solver <path2solver>`
  - comando di ottimizzazione: `solve`
  - visualizzazione dei risultati: `display <expr>`

## Esempio di file .run (lp\_1.run)

```
model lp_1.mod;  
data lp_1.dat;  
  
option solver minos;  
  
solve;  
  
display x;
```

## Esercizio 1

Installare AMPL e risolvere il problema di programmazione lineare riportato nella slide 12.

## Esercizio 2

Risolvere con AMPL+MINOS la formulazione del problema di miscelazione dell'alluminio (vedi slides del 3/10/14) e fornire una soluzione ottima. Per risolvere il problema è richiesto l'utilizzo di opportuni files `.mod`, `.dat` e `.run`.