

# Lezioni su AMPL & CPLEX

Corso di Ricerca Operativa · Prof. Gianpaolo Oriolo

Gianmaria Leo

Università di Roma “Tor Vergata”

5 Dicembre 2014

# AMPL: introduzione all'uso avanzato (III parte)

- Lezione 31/10/14: *insiemi ed espressioni*
- Lezione 21/11/14: *parametri e gestione dei dati*

Nuovi argomenti:

- 1 Opzioni
- 2 Visualizzazione
- 3 Aggiornamento
- 4 Scripting

L'istruzione `option` permette di interagire con le variabili di stato dell'ambiente AMPL

- Visualizzare il valore corrente delle variabili di stato:  
`option;`
- Visualizzare il valore corrente di una determinata variabile di stato:  
`option <option_name>;`
- Modificare il valore di una determinata variabile di stato:  
`option <option_name> <value>;`
- Resettare ciascuna opzione al valore di default:  
`reset options;`

## N.B.

L'istruzione `option` non controlla la correttezza del valore assegnato alla variabile di stato

# Principali variabili di stato e presolve

Opzione	Descrizione
<code>solver</code>	Solutore attivo
<code>&lt;solver_name&gt;_options</code>	Opzioni del solutore <code>solver_name</code>
<code>presolve</code>	Opzioni del preprocessing
<code>showstats</code>	Ottenere info dettagliate (modello, presolve)

Valori dell'opzione `presolve`:

0 Disattivato

1 Prima fase attiva:

- rafforzare i bound delle variabili
- rafforzare la formulazione con disuguaglianze valide
- eliminare vincoli e variabili implicate

$k$  Seconda fase attiva: Iterare  $k > 1$  volte la prima fase

# Visualizzare modello e istanza

L'istruzione **show** permette di visualizzare gli elementi (insiemi, parametri, variabili, vincoli, obiettivi) del modello

- Visualizzare i nomi di tutti gli elementi del modello:  
`show;`
- Visualizzare la dichiarazione di un elemento del modello:  
`show <elem_name>;`

L'istruzione **xref** permette di visualizzare le dipendenze di un determinato elemento del modello:

```
xref <elem_name>;
```

L'istruzione **expand** permette di visualizzare l'istanza

- Visualizzare il programma matematico:  
`expand;`
- Visualizzare un elemento del programma matematico (variabili, vincoli, obiettivo):  
`expand <elem_name>;`

# display

L'istruzione `display` permette di visualizzare il valore assunto da un determinato insieme (implicito) di oggetti o di una espressione

- Lista di elementi di insiemi/parametri:  
`display <obj_1>, <...>, <obj_k> ;`
- Funzione obiettivo:  
`display <obj_func>;`
- Insieme implicito dei valori di insiemi/parametri o espressioni:  
`display <implicit_set> <expr>;`

L'output desiderato può essere rediretto su in file:

```
display <...> > <file_name>.out; #write  
display <...> >> <file_name>.out; #append
```

# Opzioni per display

Opzione	Descrizione
<code>display_1col</code>	Numero massimo di elementi per riga (20)
<code>display_traspose</code>	Tabella trasposta
<code>display_width</code>	Numero massimo di caratteri del prompt <code>AMPL</code> (79)
<code>gutter_width</code>	Distanza tra le colonne di una tabella (3)
<code>omit_zero_cols</code>	Omettere le colonne contenenti valore nullo (0)
<code>omit_zero_rows</code>	Omettere le righe contenenti valore nullo (0)
<code>display_eps</code>	Valore minimo visualizzato (0)
<code>display_precision</code>	Precisione dell'arrotondamento (6)
<code>display_round</code>	Cifre decimali significative
<code>solution_precision</code>	Precisione nell'arrotondamento della soluzione (0)
<code>solution_round</code>	Cifre decimali significative della soluzione

# Info su variabili e vincoli

- Info su variabili

- lower bound e upper bound:

- `display <var>.lb, <var>.ub;`

- differenza tra il valore della variabile e il bound più vicino:

- `display <var>.slack;`

- costo ridotto:

- `display <var>.rc;`

- variabili eliminate dal presolve:

- `display {<condition_over_var> : <var>.status="pre"}`

- Info su vincoli

- valore ottimo della variabile duale associata:

- `display <constr>;`

- valore all'ottimo:

- `display <constr>.body;`

- lower bound e upper bound:

- `display <constr>.lb, <constr>.ub;`

- differenza tra il valore del vincolo e il bound più vicino

- `display <constr>.slack;`



# Sinonimi generici per variabili, vincoli e obiettivi

- Variabili:

- `_nvars`
- `_varname`
- `_var`, `_var.lb`, `_var.ub`, `_var.slack`, `_var.rc`

- Vincoli:

- `_ncons`
- `_conname`
- `_con`, `_con.lb`, `_con.ub`, `_con.slack`, `_con.body`

- Obiettivi:

- `_nobjs`
- `_objname`
- `_obj`

# Aggiornare il modello: modifiche permanenti

- Cancellare il modello:  
`reset;`
- Cancellare vincoli e funzioni obiettivo:  
`delete <obj_name>`  
`delete <constr> <implicit_set>`
- Cancellare vincoli e funzioni obiettivo dipendenti da un insieme o un parametro:  
`purge <set_name>`  
`purge <param_name>`

# Aggiornare il modello: modifiche temporanee

- Ignorare vincoli e funzioni obiettivo:

```
drop <constr_name>  
drop <constr> <implicit_set>
```

- Ripristinare vincoli e funzioni obiettivo:

```
restore <constr_name>  
restore <constr> <implicit_set>
```

- Bloccare/sbloccare variabili:

```
fix <var> := <value>;  
unfix <var>;
```

- Rilassare/ripristinare l'interezza delle variabili:

```
option relax_integrality 1;  
option relax_integrality 0;
```

- Cancellare i dati:  
`reset data <param_1>, <...>, <param_k>;`
- Aggiornare i dati (senza cancellarli fino a nuova assegnazione):  
`update data <param_1>, <...>, <param_k>;`
- Modificare singoli valori:  
`let <param> := <value>;`

# Strutture di controllo (file .run)

- Selezione **if-then-else**:

```
if <logical_condition1> then { ... }  
else if <logical_condition2> { ... }  
else { ... }
```

- Iterazione controllata **for**:

```
for {<index> in <implicit_set>}{ ... }
```

- Iterazione generica **repeat**:

```
repeat while <logical_condition> { ... };  
repeat { ... } while <logical_condition>;  
repeat until <logical_condition> { ... };  
repeat { ... } until <logical_condition>;
```

- Istruzioni **break** e **continue**:

```
repeat { ...  
  if <logical_condition_1> then break;  
  if <logical_condition_2> then continue;  
}
```