"Sapienza" Università di Roma - Dipartimento di Ingegneria Informatica, Automatica e Gestionale

Uso del linguaggio di modellazione AMPL

Renato Bruni

bruni@dis.uniroma1.it

www.dis.uniroma1.it/~bruni

Linguaggio di Modellazione

- Serve ad esprimere un **problema di ottimizzazione** in una forma che sia comprensibile dal **solutore**
- Più facile che implementare tutto in un linguaggio più a basso livello tipo c++ o java
- Ne esistono vari, vedremo **AMPL** che è uno dei più diffusi
- **AMPL** può chiamare vari solutori, ad esempio **Cplex**, che implementa simplesso, branch-and-bound, branch-and-cut e altro.
- È a pagamento, ma esiste la versione **student** limitata ma **gratuita**
- Può essere cercata su Internet o scaricata da

  http://www.dis.uniroma1.it/~bruni/files/AMPLcml.zip
- È un linguaggio algebrico, cioè contiene primitive per esprimere la notazione matematica normalmente utilizzata nello scrivere i modelli

Esprimere un modello in AMPL

- Come in altri linguaggi di programmazione, tutto viene espresso attraverso normali **file di testo**
- Si usa un **file di modello** (*.mod) che contiene la struttura logica del modello ma non i dati (indipendenza dai dati)
- I dati possono essere in un **file di dati** (*.dat) o letti in altri modi
- Al livello base, per esprimere un modello si possono usare **insiemi** (set, insiemi su cui sono organizzati i dati), **parametri** (param, per esprimere i dati), **variabili** (var)
- Occorre dichiarare ciò che serve (insiemi, parametri, variabili, …), e poi usarle il tutto per scrivere la funzione obiettivo e i vincoli nel file .mod
- Per una guida introduttiva alla sintassi vedere
  http://www.dis.uniroma1.it/~bruni/files/AMPLsyntax.pdf

Primo Esempio Introduttivo

- Una azienda produce due prodotti: **standard** e **deluxe**
- Per farlo usa tre risorse: **materia prima**, **levigatura**, **pulitura**, secondo questa tabella

	1kg standard	1kg deluxe
Materia prima	4Kg	4Kg
Levigatura	4 ore	2 ore
Pulitura	2 ore	5 ore
Prezzo vendita	10EUR/Kg	15EUR/Kg

- Le risorse sono limitate: per ogni settimana si può avere al più 75 Kg di materia prima, usare levigatura per al più 80 ore e pulitura per al più 60 ore
- Vogliamo pianificare la produzione in modo da massimizzare i ricavi

Formulazione

- Scelta variabili:

  x_1 = **quantità** in Kg di standard da produrre settimanalmente

  x_2 = **quantità** in Kg di deluxe da produrre settimanalmente

- Il modello è banalmente il seguente **PL**:

$$\max \; c^T x$$

$$Ax \leq b$$

$$x_i \geq 0 \quad i \in \{1,2\}$$

Dove $c = (10, 15)^T$ $b = (75, 80, 60)^T$ $A = \begin{pmatrix} 4 & 4 \\ 4 & 2 \\ 2 & 5 \end{pmatrix}$

File .mod

```
set PROD;      # insieme dei prodotti
set RISORSE;  # insieme delle risorse
param b{RISORSE};
param a{RISORSE,PROD};
param c{PROD};
var x{j in PROD} >=0;
maximize profit: sum{j in PROD} c[j]*x[j];
s.t. vincolo{i in RISORSE}: sum{j in PROD} a[i,j]*x[j] <= b[i];
```

File .dat

```
set PROD:=STANDARD, DELUXE;
set RISORSE:=MATERIA, LEVIGATURA, PULITURA;

param:          b:=
MATERIA         75
LEVIGATURA      80
PULITURA        60;

param a:  STANDARD   DELUXE:=
MATERIA          4           4
LEVIGATURA       4           2
PULITURA         2           5;

param:          c:=
STANDARD        10
DELUXE          15;
```

Esecuzione

Eseguire ampl.exe , (fornisce una finestra a riga di comando)

Dare i comandi (notare il punto e virgola alla fine di ogni comando):

option solver cplex; (per scegliere il solutore)

model (path)pian.mod; (per scegliere il file di modello)

data (path)pian.dat; (per scegliere il file di dati)

solve; (per risolvere)

Se i file di modello e di dati si trovano nella stessa cartella non serve il path, altrimenti si

Risultati

In caso di errori, resettare usando il comando reset;

Per correggere errori, cominciare dalla prima segnalazione di errore (viene indicata la riga) risolvere il problema e ripetere il processo.

Da notare che esistono anche errori di cui AMPL non può accorgersi (ad esempio errati valori numerici dei parametri, etc.).

Per vedere infine la soluzione, o ogni altro oggetto di interesse, usare il comando display. Per esempio:

display x; (per vedere il vettore x)

Se tutto è corretto, la soluzione del modello in esame è:

```
x [*] :=
 DELUXE   7.5
 STANDARD  11.25
```

Secondo Esempio Introduttivo

- Consideriamo il seguente modello, con variabili binarie x_{ij} che rappresentano ad esempio un collegamento tra i e j

$$\max \sum_i \sum_j x_{ij}$$

$$\sum_i x_{ij} \le b_1 \quad \forall j \quad \text{(numero di collegamenti che partono da ogni oggetto)}$$

$$\sum_j x_{ij} \le b_2 \quad \forall i \quad \text{(numero di collegamenti che arrivano ad ogni oggetto)}$$

$$\sum_i \sum_j c_{ij} x_{ij} \le b_3 \quad \text{(costo complessivo inferiore a un budget)}$$

$$x_{ij} \in \{0,1\}$$

I valori numerici delle costanti siano $c = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \\ 3 & 1 & 4 & 2 \\ 4 & 3 & 2 & 1 \end{pmatrix}$ e $b = \begin{pmatrix} 4 \\ 3 \\ 10 \end{pmatrix}$

- Vediamo come rappresentarlo in AMPL per risolverlo numericamente

File .mod versione 1

```
param n;            # numero di oggetti
set O := 1..n;      # oggetti
param c{O,O} ;      # costi
param b{1..3};      # termini noti
var x{O,O} >= 0, binary; # collegamenti attivabili

maximize obj: sum{i in O,j in O} x[i,j];
s.t. da{j in O}: sum{i in O} x[i,j] <= b[1];
s.t. ad{i in O}: sum{j in O} x[i,j] <= b[2];
s.t. cost: sum{i in O,j in O} c[i,j]*x[i,j] <= b[3];
```

File .mod versione 2

- Se invece non è possibile avere collegamento tra un oggetto e se stesso

```
param n;            # numero di oggetti
set O := 1..n;      # oggetti
param c{O,O} ;      # costi
param b{1..3};      # termini noti
var x{i in O, j in O: i<>j} >= 0, binary; # collegamenti attivabili

maximize obj: sum{i in O,j in O: i<>j} x[i,j];
s.t. da{j in O}: sum{i in O: i<>j} x[i,j] <= b[1];
s.t. ad{i in O}: sum{j in O: i<>j} x[i,j] <= b[2];
s.t. cost: sum{i in O,j in O: i<>j} c[i,j]*x[i,j] <= b[3];
```

File .mod versione 3

- Se infine i collegamenti ij e ji sono mutuamente esclusivi

```
param n;            # numero di oggetti
set O := 1..n;      # oggetti
param c{O,O} ;      # costi
param b{1..3};      # termini noti
var x{i in O, j in O: i<>j} >= 0, binary; # collegamenti attivabili

maximize obj: sum{i in O,j in O: i<>j} x[i,j];
s.t. da{j in O}: sum{i in O: i<>j} x[i,j] <= b[1];
s.t. ad{i in O}: sum{j in O: i<>j} x[i,j] <= b[2];
s.t. cost: sum{i in O,j in O: i<>j} c[i,j]*x[i,j] <= b[3];
s.t. esclusione{i in O, j in O: i<j}: x[i,j] + x[j,i] <= 1;
```

File .dat per tutte le versioni

```
param n := 4;

param b :=
      1  4
      2  3
      3 10;

param c:  1  2  3  4 :=
      1     1  2  3  4
      2     2  4  1  3
      3     3  1  4  2
      4     4  3  2  1;
```

- Per eseguirlo seguire quanto detto per il primo esempio

Un Utilizzo più Avanzato

- Vediamo un come risolvere modelli di Localizzazione degli Impianti

Risolviamo usando:

- Formulazione Forte,
- Formulazione Debole
- Simplesso Dinamico con generazione di vincoli

pl.mod in AMPL

```
param m:=  6;   # numero di clienti
param n:=  5;    # numero di impianti

set I := 1..m;  # clienti
set J := 1..n;  # impianti

param c{I,J};   # costi di allaccio
param f{J};     # costi di attivazione

var y{I,J} >=0; # variabili di allaccio
var x{J} >=0;   # variabili di attivazione

minimize costo: sum{j in J} (f[j]*x[j] + sum{i in I} c[i,j]*y[i,j]);
s.t. servizio{i in I}: sum{j in J} y[i,j] = 1;

s.t. attivazione{i in I, j in J}: x[j] - y[i,j] >= 0;        # vincoli attivaz. f forte
#s.t. attivazione_debole{j in J}: m*x[j] = sum{i in I}y[i,j];  # vincoli attivaz.
                                                                f debole
```

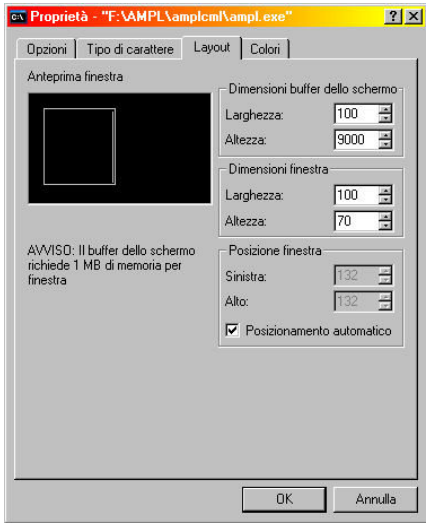
pl.dat in AMPL

```
param:  f :=
      1    4
      2    3
      3    1
      4    4
      5    7;

param c:   1    2    3    4    5 :=
        1  12  13    6    0    1
        2    8    4    9    1    2
        3    2    6    6    0    1
        4    3    5    2  10    8
        5    8    0    5  10    8
        6    2    0    3    4    1;
```

Adesso Risolviamo

- A questo punto **potremmo risolvere** scrivendo i comandi opportuni (**option solver cplex; …**) nella finestra di AMPL
- Ma possiamo anche scrivere tutti i comandi che vogliamo eseguire in un file **.run** (ad esempio pl.run) e poi lanciare quello.
- Per fare più prove è **più comodo**! E nel file **.run** possiamo anche usare comandi più avanzati
- Lo lanciamo scrivendo **include pl.run;**
- Conviene anche impostare le **proprietà** della finestra di AMPL in modo da avere molte **più righe**, ed applicarlo a tutte le finestre con lo stesso titolo



pl.run in AMPL

```
option solver cplex;      # sceglie di usare come solutore cplex
#option omit_zero_rows 1; # nel display non stampa le variabili che valgono 0
#option solver_msg 0;     # per non stampare i messaggi del solutore

model plant_location.mod; # legge il file di modello
data plant_location.dat; # legge i dati dal file di dati

solve;

printf"\n\Le variabili valgono:\n"; display x, y;
printf"Il costo totale di attivazione e': %f \n", sum{j in J} f[j]*x[j];
printf"Il costo totale di allaccio e': %f \n", sum{i in I, j in J} c[i,j]*y[i,j];
```

Soluzione Formulazione Forte

CPLEX 11.2.0: optimal solution; objective 11
9 dual simplex iterations (0 in phase I)

Le variabili valgono:
x [*] :=
1 0
2 1
3 1
4 1
5 0;

y [*,*]
: 1 2 3 4 5 :=
1 0 0 0 1 0
2 0 0 0 1 0
3 0 0 0 1 0
4 0 0 1 0 0
5 0 1 0 0 0
6 0 1 0 0 0;

Il costo totale di attivazione e': 8.000000
Il costo totale di allaccio e': 3.000000

Soluzione Formulazione Debole

CPLEX 11.2.0: optimal solution; objective 6.166666667
0 dual simplex iterations (0 in phase I)

Le variabili valgono:
x [*] :=
1 0
2 0.333333
3 0.166667
4 0.5
5 0;

y [*,*]
: 1 2 3 4 5 :=
1 0 0 0 1 0
2 0 0 0 1 0
3 0 0 0 1 0
4 0 0 1 0 0
5 0 1 0 0 0
6 0 1 0 0 0;

Il costo totale di attivazione e': 3.166667
Il costo totale di allaccio e': 3.000000

Per Fornire i Dati

- Per fornire dati, anziché il file **.dat** potremmo leggere dei file **.txt**

```
read{j in J} f[j] < costi_attivazione.txt;
read{i in I, j in J} c[i,j] < costi_allaccio.txt;
```

- Oppure potremmo **assegnarli** usando istruzioni

```
for{j in J}
{
let f[j] := 3+j;
}
```

- O ancora generarli in modo **random**

```
option randseed 0;    # per scegliere un seed basato sull'orologio di sistema.
param mean := 4;      # valore medio del numero random generato
param variance := 1.5;  # varianza del numero random generato
printf"\nf [*] :=\n";   # per stampare i valori generati
for{j in J}
{
let f[j] := Normal (mean, variance); # per generare il valore random
printf" %f\n", f[j];  # per stampare i valori generati
}
```

pld.mod (simplesso dinamico)

```
param m:=  6;   # numero di clienti
param n:=  5;    # numero di impianti

set I := 1..m;  # clienti
set J := 1..n;  # impianti

param c{I,J};   # costi di allaccio
param f{J};     # costi di attivazione

var y{I,J} >=0; # variabili di allaccio
var x{J} >=0;   # variabili di attivazione

minimize costo: sum{j in J} (f[j]*x[j] + sum{i in I} c[i,j]*y[i,j]);
s.t. servizio{i in I}: sum{j in J} y[i,j] = 1;
# non ci sono i vincoli di attivazione!
```

pld.run I (risoluzione)

```
option solver cplex;
model plant_location.mod;  # legge il file di modello
data plant_location.dat;   # legge i dati dal file di dati

param trovato;
let trovato:=0;

option cplex_auxfiles 'rc';   # per avere in cplex i nomi di var e vinc di AMPL
option cplex_options 'writeprob modellorisolto.lp'; # scrive il modello corrente

solve;

printf"\n\Le variabili valgono:\n"; display x, y;
```

[continua]…

pld.run II (generazione vincoli)

…*[continua]*

```
for{j in J, i in I}       # contolla se ci sono vincoli di attivazione violati
{
  if(x[j] < y[i,j]) then  # se ha trovato un vincolo violato
  {
    printf"\nEsiste il vincolo di attivazione violato: x[%d] >= y[%d,%d] ", j,i,j;
    printf"\nPer aggiungerlo scrivi\n nomevinc: x[%d] >= y[%d,%d];
                                                    commands go.run; \n", j,i,j;
    let trovato := 1;
    break;
  }
}
if(trovato == 0) then printf"\nTutti i vincoli di attivazione sono soddisfatti \n";
```

go.run (si ripete)

```
let trovato:=0;
option cplexamp_auxfiles 'rc';
option cplex_options 'writeprob modellorisolto.lp';
solve;
printf"\n\Le variabili valgono:\n"; display x, y;
for{j in J, i in I}
{
  if(x[j] < y[i,j]) then
  {
     printf"\nEsiste il vincolo di attivazione violato: x[%d] >= y[%d,%d] ", j,i,j;
     printf"\nPer aggiungerlo scrivi nomevincolo: x[%d] >= y[%d,%d];
                                                    commands go.run; \n", j,i,j;
                let trovato := 1;
                break;
        }
}
if(trovato == 0) then printf"\nTutti i vincoli di attivazione sono soddisfatti \n";
```

Modello Risolto 1

Minimize

obj: 12 y(1,1) + 13 y(1,2) + 6 y(1,3) + y(1,5) + 8 y(2,1) + 4 y(2,2)
+ 9 y(2,3) + y(2,4) + 2 y(2,5) + 2 y(3,1) + 6 y(3,2) + 6 y(3,3) + y(3,5)
+ 3 y(4,1) + 5 y(4,2) + 2 y(4,3) + 10 y(4,4) + 8 y(4,5) + 8 y(5,1)
+ 5 y(5,3) + 10 y(5,4) + 8 y(5,5) + 2 y(6,1) + 3 y(6,3) + 4 y(6,4)
+ y(6,5) + 4 x(1) + 3 x(2) + x(3) + 4 x(4) + 7 x(5)

Subject To

servizio(1): y(1,1) + y(1,2) + y(1,3) + y(1,4) + y(1,5) = 1
servizio(2): y(2,1) + y(2,2) + y(2,3) + y(2,4) + y(2,5) = 1
servizio(3): y(3,1) + y(3,2) + y(3,3) + y(3,4) + y(3,5) = 1
servizio(4): y(4,1) + y(4,2) + y(4,3) + y(4,4) + y(4,5) = 1
servizio(5): y(5,1) + y(5,2) + y(5,3) + y(5,4) + y(5,5) = 1
servizio(6): y(6,1) + y(6,2) + y(6,3) + y(6,4) + y(6,5) = 1

Soluzione 1

CPLEX 11.2.0: optimal solution; objective 3
0 dual simplex iterations (0 in phase I)

Le variabili valgono:
x [*] := 0 0 0 0 0 ;

y [*,*]

:	1	2	3	4	5	:=
1	0	0	0	1	0	
2	0	0	0	1	0	
3	0	0	0	1	0	
4	0	0	1	0	0	
5	0	1	0	0	0	
6	0	1	0	0	0;	

Esiste il vincolo di attivazione violato: x[2] >= y[5,2]
Per aggiungerlo scrivi nomevinc: x[2] >= y[5,2]; commands go.run;
ampl: uno: x[2] >= y[5,2]; commands go.run;

Modello Risolto 2

Minimize

obj: 12 y(1,1) + 13 y(1,2) + 6 y(1,3) + y(1,5) + 8 y(2,1) + 4 y(2,2)
+ 9 y(2,3) + y(2,4) + 2 y(2,5) + 2 y(3,1) + 6 y(3,2) + 6 y(3,3) + y(3,5)
+ 3 y(4,1) + 5 y(4,2) + 2 y(4,3) + 10 y(4,4) + 8 y(4,5) + 8 y(5,1)
+ 5 y(5,3) + 10 y(5,4) + 8 y(5,5) + 2 y(6,1) + 3 y(6,3) + 4 y(6,4)
+ y(6,5) + 4 x(1) + 3 x(2) + x(3) + 4 x(4) + 7 x(5)

Subject To

servizio(1): y(1,1) + y(1,2) + y(1,3) + y(1,4) + y(1,5) = 1
servizio(2): y(2,1) + y(2,2) + y(2,3) + y(2,4) + y(2,5) = 1
servizio(3): y(3,1) + y(3,2) + y(3,3) + y(3,4) + y(3,5) = 1
servizio(4): y(4,1) + y(4,2) + y(4,3) + y(4,4) + y(4,5) = 1
servizio(5): y(5,1) + y(5,2) + y(5,3) + y(5,4) + y(5,5) = 1
servizio(6): y(6,1) + y(6,2) + y(6,3) + y(6,4) + y(6,5) = 1
uno: - y(5,2) + x(2) >= 0

Soluzione 2

CPLEX 11.2.0: optimal solution; objective 6
1 dual simplex iterations (0 in phase I)

Le variabili valgono:
x [*] := 0 1 0 0 0 ;

y [*,*]

:	1	2	3	4	5	:=
1	0	0	0	1	0	
2	0	0	0	1	0	
3	0	0	0	1	0	
4	0	0	1	0	0	
5	0	1	0	0	0	
6	0	1	0	0	0;	

Esiste il vincolo di attivazione violato: x[3] >= y[4,3]
Per aggiungerlo scrivi nomevinc: x[3] >= y[4,3]; commands go.run;
ampl: due: x[3] >= y[4,3]; commands go.run;

Modello Risolto 3

Minimize

obj: 12 y(1,1) + 13 y(1,2) + 6 y(1,3) + y(1,5) + 8 y(2,1) + 4 y(2,2)
+ 9 y(2,3) + y(2,4) + 2 y(2,5) + 2 y(3,1) + 6 y(3,2) + 6 y(3,3) + y(3,5)
+ 3 y(4,1) + 5 y(4,2) + 2 y(4,3) + 10 y(4,4) + 8 y(4,5) + 8 y(5,1)
+ 5 y(5,3) + 10 y(5,4) + 8 y(5,5) + 2 y(6,1) + 3 y(6,3) + 4 y(6,4)
+ y(6,5) + 4 x(1) + 3 x(2) + x(3) + 4 x(4) + 7 x(5)

Subject To

servizio(1): y(1,1) + y(1,2) + y(1,3) + y(1,4) + y(1,5) = 1
servizio(2): y(2,1) + y(2,2) + y(2,3) + y(2,4) + y(2,5) = 1
servizio(3): y(3,1) + y(3,2) + y(3,3) + y(3,4) + y(3,5) = 1
servizio(4): y(4,1) + y(4,2) + y(4,3) + y(4,4) + y(4,5) = 1
servizio(5): y(5,1) + y(5,2) + y(5,3) + y(5,4) + y(5,5) = 1
servizio(6): y(6,1) + y(6,2) + y(6,3) + y(6,4) + y(6,5) = 1
uno: - y(5,2) + x(2) >= 0
due: - y(4,3) + x(3) >= 0

Soluzione 3

CPLEX 11.2.0: optimal solution; objective 7
1 dual simplex iterations (0 in phase I)

Le variabili valgono:
x [*] := 0 1 0 0 0 ;

y [*,*]
: 1 2 3 4 5 :=
1 0 0 0 1 0
2 0 0 0 1 0
3 0 0 0 1 0
4 1 0 0 0 0
5 0 1 0 0 0
6 0 1 0 0 0;

Esiste il vincolo di attivazione violato: x[1] >= y[4,1]
Per aggiungerlo scrivi nomevinc: x[1] >= y[4,1]; commands go.run;
ampl: tre: x[1] >= y[4,1]; commands go.run;

Modello Risolto 4

Minimize

obj: 12 y(1,1) + 13 y(1,2) + 6 y(1,3) + y(1,5) + 8 y(2,1) + 4 y(2,2)
+ 9 y(2,3) + y(2,4) + 2 y(2,5) + 2 y(3,1) + 6 y(3,2) + 6 y(3,3) + y(3,5)
+ 3 y(4,1) + 5 y(4,2) + 2 y(4,3) + 10 y(4,4) + 8 y(4,5) + 8 y(5,1)
+ 5 y(5,3) + 10 y(5,4) + 8 y(5,5) + 2 y(6,1) + 3 y(6,3) + 4 y(6,4)
+ y(6,5) + 4 x(1) + 3 x(2) + x(3) + 4 x(4) + 7 x(5)

Subject To

servizio(1): y(1,1) + y(1,2) + y(1,3) + y(1,4) + y(1,5) = 1
servizio(2): y(2,1) + y(2,2) + y(2,3) + y(2,4) + y(2,5) = 1
servizio(3): y(3,1) + y(3,2) + y(3,3) + y(3,4) + y(3,5) = 1
servizio(4): y(4,1) + y(4,2) + y(4,3) + y(4,4) + y(4,5) = 1
servizio(5): y(5,1) + y(5,2) + y(5,3) + y(5,4) + y(5,5) = 1
servizio(6): y(6,1) + y(6,2) + y(6,3) + y(6,4) + y(6,5) = 1
uno: - y(5,2) + x(2) >= 0
due: - y(4,3) + x(3) >= 0
tre: - y(4,1) + x(1) >= 0

Soluzione 4

CPLEX 11.2.0: optimal solution; objective 7
1 dual simplex iterations (0 in phase I)

Le variabili valgono:
x [*] := 0 1 1 0 0 ;

y [*,*]
: 1 2 3 4 5 :=
1 0 0 0 1 0
2 0 0 0 1 0
3 0 0 0 1 0
4 0 0 1 0 0
5 0 1 0 0 0
6 0 1 0 0 0;

Esiste il vincolo di attivazione violato: x[4] >= y[1,4]
Per aggiungerlo scrivi nomevinc: x[4] >= y[1,4]; commands go.run;
ampl: quattro: x[4] >= y[1,4]; commands go.run;

Modello Risolto 5

Minimize

obj: 12 y(1,1) + 13 y(1,2) + 6 y(1,3) + y(1,5) + 8 y(2,1) + 4 y(2,2)
+ 9 y(2,3) + y(2,4) + 2 y(2,5) + 2 y(3,1) + 6 y(3,2) + 6 y(3,3) + y(3,5)
+ 3 y(4,1) + 5 y(4,2) + 2 y(4,3) + 10 y(4,4) + 8 y(4,5) + 8 y(5,1)
+ 5 y(5,3) + 10 y(5,4) + 8 y(5,5) + 2 y(6,1) + 3 y(6,3) + 4 y(6,4)
+ y(6,5) + 4 x(1) + 3 x(2) + x(3) + 4 x(4) + 7 x(5)

Subject To

servizio(1): y(1,1) + y(1,2) + y(1,3) + y(1,4) + y(1,5) = 1
servizio(2): y(2,1) + y(2,2) + y(2,3) + y(2,4) + y(2,5) = 1
servizio(3): y(3,1) + y(3,2) + y(3,3) + y(3,4) + y(3,5) = 1
servizio(4): y(4,1) + y(4,2) + y(4,3) + y(4,4) + y(4,5) = 1
servizio(5): y(5,1) + y(5,2) + y(5,3) + y(5,4) + y(5,5) = 1
servizio(6): y(6,1) + y(6,2) + y(6,3) + y(6,4) + y(6,5) = 1
uno: - y(5,2) + x(2) >= 0
due: - y(4,3) + x(3) >= 0
tre: - y(4,1) + x(1) >= 0
quattro: - y(1,4) + x(4) >= 0

Soluzione 5

CPLEX 11.2.0: optimal solution; objective 8
1 dual simplex iterations (0 in phase I)

Le variabili valgono:
x [*] := 0 1 1 0 0 ;

y [*,*]
: 1 2 3 4 5 :=
1 0 0 0 0 1
2 0 0 0 1 0
3 0 0 0 1 0
4 0 0 1 0 0
5 0 1 0 0 0
6 0 1 0 0 0;

Esiste il vincolo di attivazione violato: x[4] >= y[2,4]
Per aggiungerlo scrivi nomevinc: x[4] >= y[2,4]; commands go.run;
ampl: cinque: x[4] >= y[2,4]; commands go.run;

Modello Risolto 6

Minimize

obj: 12 y(1,1) + 13 y(1,2) + 6 y(1,3) + y(1,5) + 8 y(2,1) + 4 y(2,2)
+ 9 y(2,3) + y(2,4) + 2 y(2,5) + 2 y(3,1) + 6 y(3,2) + 6 y(3,3) + y(3,5)
+ 3 y(4,1) + 5 y(4,2) + 2 y(4,3) + 10 y(4,4) + 8 y(4,5) + 8 y(5,1)
+ 5 y(5,3) + 10 y(5,4) + 8 y(5,5) + 2 y(6,1) + 3 y(6,3) + 4 y(6,4)
+ y(6,5) + 4 x(1) + 3 x(2) + x(3) + 4 x(4) + 7 x(5)

Subject To

servizio(1): y(1,1) + y(1,2) + y(1,3) + y(1,4) + y(1,5) = 1
servizio(2): y(2,1) + y(2,2) + y(2,3) + y(2,4) + y(2,5) = 1
servizio(3): y(3,1) + y(3,2) + y(3,3) + y(3,4) + y(3,5) = 1
servizio(4): y(4,1) + y(4,2) + y(4,3) + y(4,4) + y(4,5) = 1
servizio(5): y(5,1) + y(5,2) + y(5,3) + y(5,4) + y(5,5) = 1
servizio(6): y(6,1) + y(6,2) + y(6,3) + y(6,4) + y(6,5) = 1
uno: - y(5,2) + x(2) >= 0
due: - y(4,3) + x(3) >= 0
tre: - y(4,1) + x(1) >= 0
quattro: - y(1,4) + x(4) >= 0
cinque: - y(2,4) + x(4) >= 0

Soluzione 6

CPLEX 11.2.0: optimal solution; objective 9
0 dual simplex iterations (0 in phase I)

Le variabili valgono:
x [*] := 0 1 1 0 0 ;

y [*,*]
: 1 2 3 4 5 :=
1 0 0 0 0 1
2 0 0 0 0 1
3 0 0 0 1 0
4 0 0 1 0 0
5 0 1 0 0 0
6 0 1 0 0 0;

Esiste il vincolo di attivazione violato: x[4] >= y[3,4]
Per aggiungerlo scrivi nomevinc: x[4] >= y[3,4]; commands go.run;
ampl: sei: x[4] >= y[3,4]; commands go.run;

Modello Risolto 7

Minimize

obj: 12 y(1,1) + 13 y(1,2) + 6 y(1,3) + y(1,5) + 8 y(2,1) + 4 y(2,2)
+ 9 y(2,3) + y(2,4) + 2 y(2,5) + 2 y(3,1) + 6 y(3,2) + 6 y(3,3) + y(3,5)
+ 3 y(4,1) + 5 y(4,2) + 2 y(4,3) + 10 y(4,4) + 8 y(4,5) + 8 y(5,1)
+ 5 y(5,3) + 10 y(5,4) + 8 y(5,5) + 2 y(6,1) + 3 y(6,3) + 4 y(6,4)
+ y(6,5) + 4 x(1) + 3 x(2) + x(3) + 4 x(4) + 7 x(5)

Subject To

servizio(1): y(1,1) + y(1,2) + y(1,3) + y(1,4) + y(1,5) = 1
servizio(2): y(2,1) + y(2,2) + y(2,3) + y(2,4) + y(2,5) = 1
servizio(3): y(3,1) + y(3,2) + y(3,3) + y(3,4) + y(3,5) = 1
servizio(4): y(4,1) + y(4,2) + y(4,3) + y(4,4) + y(4,5) = 1
servizio(5): y(5,1) + y(5,2) + y(5,3) + y(5,4) + y(5,5) = 1
servizio(6): y(6,1) + y(6,2) + y(6,3) + y(6,4) + y(6,5) = 1
uno: - y(5,2) + x(2) >= 0
due: - y(4,3) + x(3) >= 0
tre: - y(4,1) + x(1) >= 0
quattro: - y(1,4) + x(4) >= 0
cinque: - y(2,4) + x(4) >= 0
sei: - y(3,4) + x(4) >= 0

Soluzione 7

CPLEX 11.2.0: optimal solution; objective 10
1 dual simplex iterations (0 in phase I)

Le variabili valgono:
x [*] := 0 1 1 0 0 ;

y [*,*]
: 1 2 3 4 5 :=
1 0 0 0 0 1
2 0 0 0 0 1
3 0 0 0 0 1
4 0 0 1 0 0
5 0 1 0 0 0
6 0 1 0 0 0;

Esiste il vincolo di attivazione violato: x[5] >= y[1,5]
Per aggiungerlo scrivi nomevinc: x[5] >= y[1,5]; commands go.run;
ampl: sette: x[5] >= y[1,5]; commands go.run;

Modello Risolto 8

Minimize
obj: 12 y(1,1) + 13 y(1,2) + 6 y(1,3) + y(1,5) + 8 y(2,1) + 4 y(2,2)
+ 9 y(2,3) + y(2,4) + 2 y(2,5) + 2 y(3,1) + 6 y(3,2) + 6 y(3,3) + y(3,5)
+ 3 y(4,1) + 5 y(4,2) + 2 y(4,3) + 10 y(4,4) + 8 y(4,5) + 8 y(5,1)
+ 5 y(5,3) + 10 y(5,4) + 8 y(5,5) + 2 y(6,1) + 3 y(6,3) + 4 y(6,4)
+ y(6,5) + 4 x(1) + 3 x(2) + x(3) + 4 x(4) + 7 x(5)
Subject To
servizio(1): y(1,1) + y(1,2) + y(1,3) + y(1,4) + y(1,5) = 1
servizio(2): y(2,1) + y(2,2) + y(2,3) + y(2,4) + y(2,5) = 1
servizio(3): y(3,1) + y(3,2) + y(3,3) + y(3,4) + y(3,5) = 1
servizio(4): y(4,1) + y(4,2) + y(4,3) + y(4,4) + y(4,5) = 1
servizio(5): y(5,1) + y(5,2) + y(5,3) + y(5,4) + y(5,5) = 1
servizio(6): y(6,1) + y(6,2) + y(6,3) + y(6,4) + y(6,5) = 1
uno: - y(5,2) + x(2) >= 0
due: - y(4,3) + x(3) >= 0
tre: - y(4,1) + x(1) >= 0
quattro: - y(1,4) + x(4) >= 0
cinque: - y(2,4) + x(4) >= 0
sei: - y(3,4) + x(4) >= 0
sette: - y(1,5) + x(5) >= 0

finora ho 7 vincoli di attivazione
dei 6x5=30 possibili

Soluzione 8 (fine)

CPLEX 11.2.0: optimal solution; objective 11
3 dual simplex iterations (0 in phase I)

Le variabili valgono:
x [*] := 0 1 1 1 0 ;

y [*,*]
: 1 2 3 4 5 :=
1 0 0 0 1 0
2 0 0 0 1 0
3 0 0 0 1 0
4 0 0 1 0 0
5 0 1 0 0 0
6 0 1 0 0 0;

Soluzione ottima trovata aggiungendo 7 vincoli di attivazione invece di 6x5=30

Per problemi più grandi fa la differenza tra risolvere o no

Tutti i vincoli di attivazione sono soddisfatti

.mod Automatico

```
param m:= 16; # numero di clienti
param n:= 16; # numero di impianti
set I := 1..m; # clienti
set J := 1..n; # impianti

param c{I,J}; # costi di allaccio
param f{J}; # costi di attivazione
read {i in I,j in J} c[i,j] < matrice.txt;
read {j in J} f[j] < vettore.txt;

param k, default 0; # numero di vincoli generati
param A{I,J}, default 0; # indica vincoli generati

var y{I,J} >=0; # variabili di allaccio
var x{J} >=0; # variabili di attivazione

minimize costo: sum{j in J} (f[j]*x[j] + sum{i in I} c[i,j]*y[i,j]);
s.t. servizio{i in I}: sum{j in J} y[i,j] = 1;
s.t. attivazione{i in I, j in J: A[i,j] > 0}: x[j] - y[i,j] >= 0;
# la matrice A indica quali vincoli sono stati generati
```

.run Automatico

```
option solver cplex;
model pld4.mod; # legge il file di modello
param trovato;    # flag per capire se ho trovato un vincolo di servizio violato
param epsilon := 0.001;  # tolleranza numerica

option cplex_auxfiles 'rc'; # per avere in cplex i nomi di var e vinc di AMPL
option cplex_options 'writeprob modellorisolto.lp'; # scrive il modello corrente
solve;
printf"\n\Le variabili valgono:\n"; display x, y;

repeat
{
		let trovato:=0;
		for{i in I, j in J} # controlla se ci sono vincoli di attivazione violati
		{
			if( x[j] < y[i,j] - epsilon ) then  # se ha trovato un vincolo violato
			{
				printf"\nTrovato vincolo di attivazione violato: x[%d] >= y[%d,%d] \n", j,i,j;
				let trovato:=1;
				let k := k + 1;  # contatore vincoli generati
				let A[i,j] := 1; # segna quali vincolo generare
				break;
			}
		}
		if (trovato == 1) then
		{
			option cplex_auxfiles 'rc'; # per avere in cplex i nomi di var e vinc di AMPL
			option cplex_options 'writeprob modellorisolto.lp'; # scrive il modello corrente
			solve;
			printf"\n\Le variabili valgono:\n"; display x, y;
		}
}
while (trovato == 1);
printf"\nTutti i vincoli di servizio sono soddisfatti dopo averne aggiunti %d su %d\n", k, m*n;
```