Problem 1

Alessandro Pontini

4/29/2020

```
library(lpSolveAPI)
```

Prendo il problema ILP ed inizio a risolverlo.

$$
\begin{aligned}
\max \quad & 9x_1 + 5x_2 + 6x_3 + 4x_4 \\
\text{s.t.} \quad & \\
& 6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10, \\
& x_3 + x_4 \leq 1, \\
& -x_1 + x_3 \leq 0, \\
& -x_2 + x_4 \leq 0 \\
& x_1, x_2, x_3, x_4 \in \{0, 1\}
\end{aligned}
$$

Nodo 1

```
# creo il modello
model = make.lp(0,4, verbose = 'full')

# imposto funzione di massimizzazione
lp.control(model, sense='max')

# metto i coefficienti
set.objfn(model, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
#set.type(model, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(model,
               xt=c(6,3,5,2),
               type="<=",rhs=10,
               indices=c(1:4))

# costraint 2
add.constraint(model,
               xt=c(1,1),
               type="<=",rhs=1,
               indices=c(3:4))

# costraint 3
add.constraint(model,
```

```
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(1,3))

# costraint 4
add.constraint(model,
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(2,4))
# metto bounds da 0
set.bounds(model,lower=c(0,0,0,0), upper = c(1,1,1,1))
# usiamo dopo
#set.type(lp.model, c(1:4), "binary")
solve(model)
```

```
get.objective(model)
```

```
## [1] 16.5
```

```
get.variables(model)
```

```
## [1] 0.8333333 1.0000000 0.0000000 1.0000000
```

Prendiamo il nodo 1 e ci restituisce una soluzione co Z=16.5 e l'unico valore che non possiamo accettare per il problema di ILP che potrebbe essere dedotto in BLP essendo binario. Pertanto con un valore di 5/6 per x_1 dobbiamo creare due nodi ulteriori 2 e 3 che abbiano come valore rispettivamente $x_1 = 0$ e $x_1 = 1$.

Nodo 2

```
model2 = make.lp(0,4, verbose = 'full')

# imposto funzione di massimizzazione
lp.control(model2, sense='max')

# metto i coefficienti
set.objfn(model2, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
#set.type(model, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(model2,
                xt=c(6,3,5,2),
                type="<=",rhs=10,
                indices=c(1:4))

# costraint 2
add.constraint(model2,
                xt=c(1,1),
                type="<=",rhs=1,
                indices=c(3:4))

# costraint 3
add.constraint(model2,
                xt=c(-1,1),
```

```
                type="<=",rhs=0,
                indices=c(1,3))

# costraint 4
add.constraint(model2,
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(2,4))
# metto bounds da 0
set.bounds(model2,lower=c(0,0,0,0), upper = c(0,1,1,1))
solve(model2)
```

```
get.objective(model2)
```

```
## [1] 9
```

```
get.variables(model2)
```

```
## [1] 0 1 0 1
```

Otteniziamo una $Z = 9$ che è un risultato accettabile per il problema di ILP poichè tutti i valori x_1, x_2, x_3, x_4 assumono valori di interi che sono $0, 1, 0, 1$. Qui il problema se consideriamo l'albero completo decisionale dovremmo portare avanti la decisione sul x_2, x_3 che possono diventare 0. Avremmo il nodo 2_1 e il nodo 2_2 che avrebbero avuto $x_2 = 1/0$ i quali a loro volta avrebbero avuto una soluzione identica tra nodo 2 e nodo 2_1, poichè identiche, mentre nell'altra si avrebbe avuto $Z = 0$ con tutti i valori a 0. Iterando ancora avremmo ottenuto soluzioni infeasible. La soluzione con $Z = 0$ la consideriamo comunque già superata per un problema di massimizzazione poichè abbiamo $Z = 9$ per il nodo 2. Controlliamo ora il nodo 3 come si comporta.

Nodo 3

```
model3 = make.lp(0,4, verbose = 'full')

# imposto funzione di massimizzazione
lp.control(model3, sense='max')

# metto i coefficienti
set.objfn(model3, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
#set.type(model, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(model3,
                xt=c(6,3,5,2),
                type="<=",rhs=10,
                indices=c(1:4))

# costraint 2
add.constraint(model3,
                xt=c(1,1),
                type="<=",rhs=1,
                indices=c(3:4))

# costraint 3
add.constraint(model3,
```

```
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(1,3))

# costraint 4
add.constraint(model3,
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(2,4))
# metto bounds da 0
set.bounds(model3,lower=c(1,0,0,0), upper = c(1,1,1,1))
solve(model3)
```

```
get.objective(model3)
```

```
## [1] 16.2
```

```
get.variables(model3)
```

```
## [1] 1.0 0.8 0.0 0.8
```

Al contrario della soluzione 2, abbiamo che x_1, x_2, x_3, x_4 assumono valori per $1, 0.8, 0, 0.8$. Con uno $Z = 16.2$ decidiamo di creare i nodi 4 e nodi 5 che vedano al variare di x_2 in base a $x_2 = 0$ e $x_2 = 1$. come varia la soluzione Z.

Nodo 4

```
model4 = make.lp(0,4, verbose = 'full')

# imposto funzione di massimizzazione
lp.control(model4, sense='max')

# metto i coefficienti
set.objfn(model4, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
#set.type(model, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(model4,
                xt=c(6,3,5,2),
                type="<=",rhs=10,
                indices=c(1:4))

# costraint 2
add.constraint(model4,
                xt=c(1,1),
                type="<=",rhs=1,
                indices=c(3:4))

# costraint 3
add.constraint(model4,
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(1,3))
```

```
# costraint 4
add.constraint(model4,
               xt=c(-1,1),
               type="<=",rhs=0,
               indices=c(2,4))
# metto bounds da 0
set.bounds(model4,lower=c(1,0,0,0), upper = c(1,0,1,1))
solve(model4)
```

```
get.objective(model4)
```

```
## [1] 13.8
```

```
get.variables(model4)
```

```
## [1] 1.0 0.0 0.8 0.0
```

Ottengo un nodo che mi restituisce per x_1, x_2, x_3, x_4 le soluzioni $1, 0, 0.8, 0$ con uno $Z = 13.8$. Questa soluzione non soddisfa l'algoritmo che continuerà ad iterare per questa volta $x_3 = 0/1$.

Nodo 5

```
model5 = make.lp(0,4, verbose = 'full')

# imposto funzione di massimizzazione
lp.control(model5, sense='max')

# metto i coefficienti
set.objfn(model5, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
#set.type(model, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(model5,
               xt=c(6,3,5,2),
               type="<=",rhs=10,
               indices=c(1:4))

# costraint 2
add.constraint(model5,
               xt=c(1,1),
               type="<=",rhs=1,
               indices=c(3:4))

# costraint 3
add.constraint(model5,
               xt=c(-1,1),
               type="<=",rhs=0,
               indices=c(1,3))

# costraint 4
add.constraint(model5,
               xt=c(-1,1),
```

```
                type="<=",rhs=0,
                indices=c(2,4))
# metto bounds da 0
set.bounds(model5,lower=c(1,1,0,0), upper = c(1,1,1,1))
solve(model5)
```

```
get.objective(model5)
```

```
## [1] 16
```

```
get.variables(model5)
```

```
## [1] 1.0 1.0 0.0 0.5
```

In questo caso otteniamo sostanzialmente una soluzione di $Z = 16$ con un valore delle variabili per $1, 1, 0, 0.5$ questo ci porta a considera perciò questa soluzione non ancora accettabile comunque la più appetibile. Nonostante tutto continuamo ora con il Nodo 4 e poi Nodo 5. Dove controlleremo $x_3 = 0/1$ per entrambi.

Nodo 6 e 7

```
model6 = make.lp(0,4, verbose = 'full')

# imposto funzione di massimizzazione
lp.control(model6, sense='max')

# metto i coefficienti
set.objfn(model6, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
#set.type(model, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(model6,
                xt=c(6,3,5,2),
                type="<=",rhs=10,
                indices=c(1:4))

# costraint 2
add.constraint(model6,
                xt=c(1,1),
                type="<=",rhs=1,
                indices=c(3:4))

# costraint 3
add.constraint(model6,
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(1,3))

# costraint 4
add.constraint(model6,
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(2,4))
# metto bounds da 0
```

```r
set.bounds(model6,lower=c(1,1,1,0), upper = c(1,1,1,1))
solve(model6)
```

```r
get.objective(model6)
```

```
## [1] -1e+30
```

```r
get.variables(model6)
```

```
## [1]  0.000000e+00 6.943237e-310 6.943237e-310 6.943238e-310
```

```r
model7 = make.lp(0,4, verbose = 'full')

# imposto funzione di massimizzazione
lp.control(model7, sense='max')

# metto i coefficienti
set.objfn(model7, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
#set.type(model, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(model7,
               xt=c(6,3,5,2),
               type="<=",rhs=10,
               indices=c(1:4))

# costraint 2
add.constraint(model7,
               xt=c(1,1),
               type="<=",rhs=1,
               indices=c(3:4))

# costraint 3
add.constraint(model7,
               xt=c(-1,1),
               type="<=",rhs=0,
               indices=c(1,3))

# costraint 4
add.constraint(model7,
               xt=c(-1,1),
               type="<=",rhs=0,
               indices=c(2,4))
# metto bounds da 0
set.bounds(model7,lower=c(1,1,0,0), upper = c(1,1,0,1))
solve(model7)
```

```r
get.objective(model7)
```

```
## [1] 16
```

```r
get.variables(model7)
```

```
## [1] 1.0 1.0 0.0 0.5
```

Otteniamo che il Nodo 7 è identico al nodo 4 mentre il nodo 6 ci restituisce una soluzione infeasible come avevamo descritto precedentemente. Pertanto l'unica soluzione per cui itereremo ancora è la 7. Creeremo nodi 10 e 11 per il nodo 7.

Nodo 8 e 9

```
model8 = make.lp(0,4, verbose = 'full')

# imposto funzione di massimizzazione
lp.control(model8, sense='max')

# metto i coefficienti
set.objfn(model8, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
#set.type(model, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(model8,
               xt=c(6,3,5,2),
               type="<=",rhs=10,
               indices=c(1:4))

# costraint 2
add.constraint(model8,
               xt=c(1,1),
               type="<=",rhs=1,
               indices=c(3:4))

# costraint 3
add.constraint(model8,
               xt=c(-1,1),
               type="<=",rhs=0,
               indices=c(1,3))

# costraint 4
add.constraint(model8,
               xt=c(-1,1),
               type="<=",rhs=0,
               indices=c(2,4))
# metto bounds da 0
set.bounds(model8,lower=c(1,0,1,0), upper = c(1,0,1,1))
solve(model8)
```

```
get.objective(model8)
```

```
## [1] -1e+30
```

```
get.variables(model8)
```

```
## [1]  0.000000e+00 6.943237e-310 6.943237e-310 6.943238e-310
```

```
model9 = make.lp(0,4, verbose = 'full')

# imposto funzione di massimizzazione
```

```
lp.control(model9, sense='max')

# metto i coefficienti
set.objfn(model9, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
#set.type(model, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(model9,
               xt=c(6,3,5,2),
               type="<=",rhs=10,
               indices=c(1:4))

# costraint 2
add.constraint(model9,
               xt=c(1,1),
               type="<=",rhs=1,
               indices=c(3:4))

# costraint 3
add.constraint(model9,
               xt=c(-1,1),
               type="<=",rhs=0,
               indices=c(1,3))

# costraint 4
add.constraint(model9,
               xt=c(-1,1),
               type="<=",rhs=0,
               indices=c(2,4))
# metto bounds da 0
set.bounds(model9,lower=c(1,0,0,0), upper = c(1,0,0,1))
solve(model9)
```

```
get.objective(model9)
```

```
## [1] 9
```

```
get.variables(model9)
```

```
## [1] 1 0 0 0
```

Otteniamo per il Nodo 8 una soluzione Infeasible mentre per il nodo 9 otteniamo la soluzione di $Z = 9$ con le variabili che prendono i valori di $1, 0, 0, 0$ che r4isulta essere equivalente alla soluzione del Nodo 2 ma con diversi valori nelle x. Come il nodo 2 se continuassimo ad iterare otterremmo la soluzione Nodo 9_1 con $x_4 = 0$ mentre una soluzione infeasible per $x_4 = 1$. Continuiamo con i nodi 10 e 11.

Nodi 10 e 11

```
model10 = make.lp(0,4, verbose = 'full')

# imposto funzione di massimizzazione
lp.control(model10, sense='max')
```

```r
# metto i coefficienti
set.objfn(model10, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
#set.type(model, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(model10,
               xt=c(6,3,5,2),
               type="<=",rhs=10,
               indices=c(1:4))

# costraint 2
add.constraint(model10,
               xt=c(1,1),
               type="<=",rhs=1,
               indices=c(3:4))

# costraint 3
add.constraint(model10,
               xt=c(-1,1),
               type="<=",rhs=0,
               indices=c(1,3))

# costraint 4
add.constraint(model10,
               xt=c(-1,1),
               type="<=",rhs=0,
               indices=c(2,4))
# metto bounds da 0
set.bounds(model10,lower=c(1,1,0,1), upper = c(1,1,0,1))
solve(model10)
```

```r
get.objective(model10)
```

```
## [1] -1e+30
```

```r
get.variables(model10)
```

```
## [1] 1.251613e-308 6.326388e-310 1.379807e-309 7.068368e-311
```

```r
model11 = make.lp(0,4, verbose = 'full')

# imposto funzione di massimizzazione
lp.control(model11, sense='max')

# metto i coefficienti
set.objfn(model11, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
#set.type(model, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(model11,
               xt=c(6,3,5,2),
```

```
                type="<=",rhs=10,
                indices=c(1:4))

# costraint 2
add.constraint(model11,
                xt=c(1,1),
                type="<=",rhs=1,
                indices=c(3:4))

# costraint 3
add.constraint(model11,
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(1,3))

# costraint 4
add.constraint(model11,
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(2,4))
# metto bounds da 0
set.bounds(model11,lower=c(1,1,0,0), upper = c(1,1,0,0))
solve(model11)
```

```
get.objective(model11)
```

```
## [1] 14
```

```
get.variables(model11)
```

```
## [1] 1 1 0 0
```

Otteniamo infine per i modelli 10 e 11 i seguenti risultati. Per il modello 10 otteniamo una soluzione infeasible mentre nel modello 11 otteniamo la soluzione al nostro problema ILP (BLP) per cui si ha esattamente la massimizzazione del problema con $Z = 14$ e i valori di x_1, x_2, x_3, x_4 uguali a $1, 1, 0, 0$. Questa soluzione risulta maggiore delle soluzioni dei nodi 2,9. Per la risoluzione del problema potevamo usare direttamente questo codice, che mette come tipo di variabili x_1, x_2, x_3, x_4 binarie e ci restituisce il valore finale.

Soluzione Finale

```
# creo il modello
modelx = make.lp(0,4)

# imposto funzione di massimizzazione
lp.control(modelx, sense='max')

# metto i coefficienti
set.objfn(modelx, obj=c(9,5,6,4))

# imposto problema  binario per x1,x2,x3,x4
set.type(modelx, c(1:4), "binary")

# aggiungo costraint 1
add.constraint(modelx,
                xt=c(6,3,5,2),
```

```
                type="<=",rhs=10,
                indices=c(1:4))

# costraint 2
add.constraint(modelx,
                xt=c(1,1),
                type="<=",rhs=1,
                indices=c(3:4))

# costraint 3
add.constraint(modelx,
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(1,3))

# costraint 4
add.constraint(modelx,
                xt=c(-1,1),
                type="<=",rhs=0,
                indices=c(2,4))

# metto bounds da 0
set.bounds(modelx,lower=c(0,0,0,0), upper = c(1,1,1,1))

solve(modelx)
get.objective(modelx)
get.variables(modelx)
```

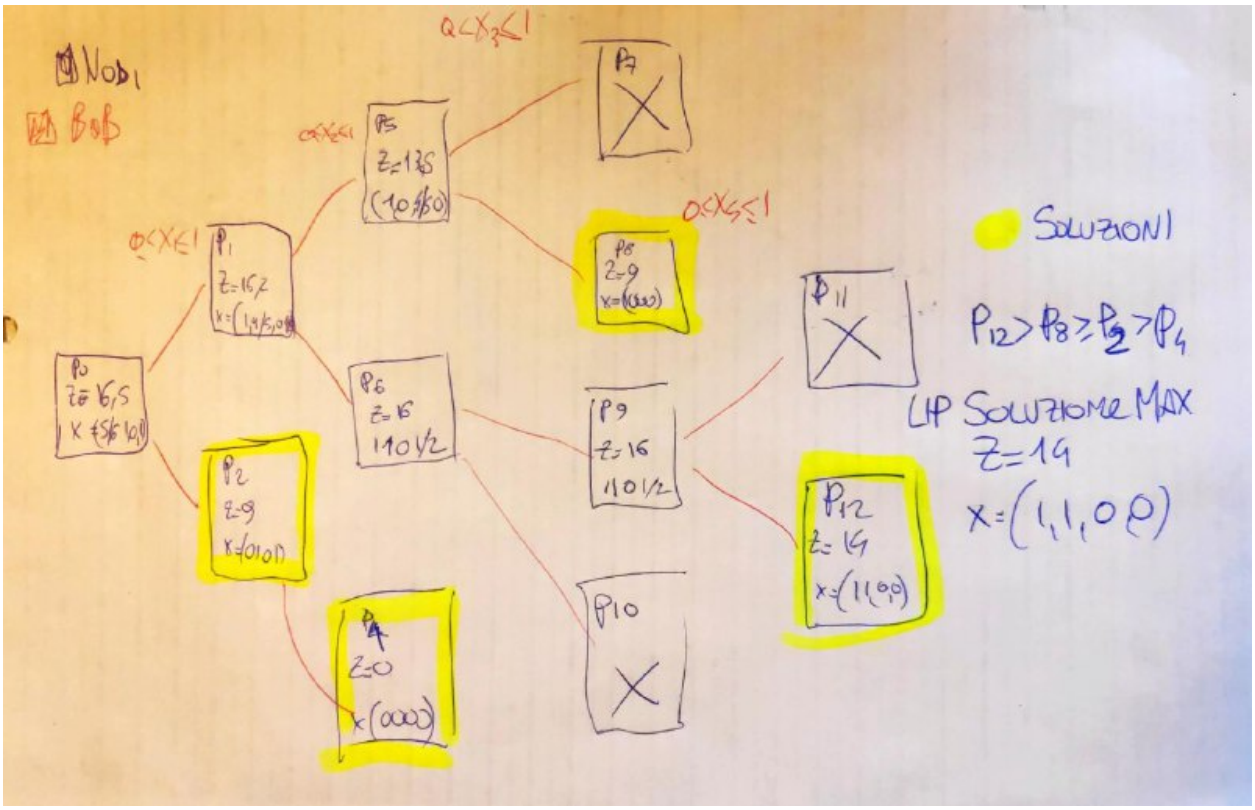
```
get.objective(modelx)
```

```
## [1] 14
```

```
get.variables(modelx)
```

```
## [1] 1 1 0 0
```

Aggiungo il mio ragionamento in foto.



Per finire mi calcolo gli intervalli del Branch and Bound per i nodi toccati per cui ottengo:

$$Nodo0 : UB : x' = (5/6, 1, 0, 1), Z = 16.5; LB : x' = (0, 1, 0, 1), Z = 9$$
$$Nodo1 : UB : x' = (1, 4/5, 0, 4/5), Z = 16.2; LB : x' = (1, 0, 0, 0), Z = 9$$
$$Nodo5 : UB : x' = (1, 0, 4/5, 0), Z = 13.8; LB : x' = (1, 0, 0, 0), Z = 9$$
$$Nodo6e9 : UB : x' = (1, 1, 0, 1/2), Z = 16; LB : x' = (1, 1, 0, 0), Z = 14$$