Linear Programming - Assignment 1

0 - Il problema

Un'azienda di materiali da costruzione sta cercando un modo per massimizzare il profitto per il trasporto delle sue merci.

L'azienda ha a disposizione un treno con 4 vagoni.

Per rifornire i vagoni si possono scegliere tra 3 tipi di carico, ognuno con le sue specifiche.

Quanto di ogni tipo di carico dovrebbe essere caricato su quale vagone per massimizzare il profitto?

1 - Dati

TRAIN WAGON j	WEIGHT CAPACITY (TONNE) w_j	VOLUME CAPACITY (m^2) s_j
(wag) 1	10	5000
(wag) 2	8	4000
(wag) 3	12	8000
(wag) 4	6	2500

CARGO TYPE i	AVAILABLE (TONNE) a_i	VOLUME (m^2/t) v_i	PROFIT (PER TONNE) p_i
(cg) 1	20	500	3500
(cg) 2	10	300	2500
(cg) 3	18	400	2000

2 - Variabile di decisione

```
library(lpSolveAPI)
model = make.lp(0,12)
lp.control(model, sense="max")
```

```
## $anti.degen
## [1] "fixedvars" "stalling"
##
## $basis.crash
## [1] "none"
##
## $bb.depthlimit
## [1] -50
##
## $bb.floorfirst
## [1] "automatic"
##
## $bb.rule
## [1] "pseudononint" "greedy"       "dynamic"      "rcostfixing"
##
## $break.at.first
## [1] FALSE
##
## $break.at.value
## [1] 1e+30
##
## $epsilon
##       epsb       epsd      epsel     epsint epsperturb   epspivot
##      1e-10      1e-09      1e-12      1e-07      1e-05      2e-07
##
## $improve
## [1] "dualfeas" "thetagap"
##
## $infinite
## [1] 1e+30
##
## $maxpivot
## [1] 250
##
## $mip.gap
## absolute relative
##    1e-11    1e-11
##
## $negrange
## [1] -1e+06
##
## $obj.in.basis
## [1] TRUE
##
## $pivoting
## [1] "devex"    "adaptive"
##
## $presolve
## [1] "none"
##
## $scalelimit
## [1] 5
##
## $scaling
## [1] "geometric"   "equilibrate" "integers"
##
## $sense
## [1] "maximize"
##
## $simplextype
## [1] "dual"   "primal"
##
## $timeout
## [1] 0
##
## $verbose
## [1] "neutral"
```

Definiamo le variabili di decisioni: ogni variabile x_{ij} rappresenta l'i-esimo carico sul j-esimo vagone $.i \in (1,2,3)$ e $j \in (1,2,3,4)$, le variabili definite sono 12.

3 - Funzione obiettivo

```
set.objfn(model,obj=c(3500,3500,3500,3500,2500,2500,2500,2500,2000,2000,2000,2000))
```

Definiamo la funzione obiettivo che andrà massimizzata :

$max \sum_{i=1}^{3} \sum_{j=1}^{4} p_i x_{ij}$ con $p \in (3500, 2500, 2000)$ che rappresenta il profitto per ogni tonnellata di carico.

Moltiplichiamo ogni variabile per il profitto e calcoliamo la sommatoria di questa quantità per tutte le variabili considerate, massimizzando il profitto.

4 - I vincoli

I primi quattro vincoli si riferiscono al volume massimo di ogni vagone e v_i rappresenta il volume del i-esimo carico .

$\sum_{i=1}^{3} x_{i1} v_i \leq 5000$ $\sum_{i=1}^{3} x_{i2} v_i \leq 4000$ $\sum_{i=1}^{3} x_{i3} v_i \leq 8000$ $\sum_{i=1}^{3} x_{i4} v_i \leq 2500$

```
add.constraint(model,
               xt=c(500,300,400),
               type="<=",rhs=5000,
               indices=c(1,5,9))
add.constraint(model,
               xt=c(500,300,400),
               type="<=",rhs=4000,
               indices=c(2,6,10))
add.constraint(model,
               xt=c(500,300,400),
               type="<=",rhs=8000,
               indices=c(3,7,11))
add.constraint(model,
               xt=c(500,300,400),
               type="<=",rhs=2500,
               indices=c(4,8,12))
```

Definiamo altri quattro bvincoli che si riferiscono al peso massimo in tonnelate per ogni vagone.

$\sum_{i=1}^{3} x_{i1} \leq 10$ $\sum_{i=1}^{3} x_{i2} \leq 8$ $\sum_{i=1}^{3} x_{i3} \leq 12$ $\sum_{i=1}^{3} x_{i4} \leq 6$

```
add.constraint(model,
               xt=c(1,1,1),
               type="<=",rhs=10,
               indices=c(1,5,9))
add.constraint(model,
               xt=c(1,1,1),
               type="<=",rhs=8,
               indices=c(2,6,10))
add.constraint(model,
               xt=c(1,1,1),
               type="<=",rhs=12,
               indices=c(3,7,11))
add.constraint(model,
               xt=c(1,1,1),
               type="<=",rhs=6,
               indices=c(4,8,12))
```

Aggiungiamo altri tre vincoli che si riferiscono alla quantità disponibile di ogni carico:

$\sum_{j=1}^{4} x_{1j} \leq 20$ $\sum_{j=1}^{4} x_{2j} \leq 10$ $\sum_{j=1}^{4} x_{3j} \leq 18$

```
add.constraint(model,
               xt=c(1,1,1,1),
               type="<=",rhs=20,
               indices=c(1:4))
add.constraint(model,
               xt=c(1,1,1,1),
               type="<=",rhs=10,
               indices=c(5:8))
add.constraint(model,
               xt=c(1,1,1,1),
               type="<=",rhs=18,
               indices=c(9:12))
```

Quindi come ultimo vincolo settiamo l' il limite inferiore per le variabili di decisioni, sapendo che le quantità non possono essere minori di 0.

```
set.bounds(model,lower=c(0,0,0,0,0,0,0,0,0,0,0,0))
```

5 - Soluzione del problema

Quindi analizziamo le soluzioni del modello.

```
model
```

```
## Model name:
##   a linear program with 12 decision variables and 11 constraints
```

```
solve(model)
```

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

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

```
##  [1] 10  8  2  0  0  0 10  0  0  0  0  6
```

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

```
## [1] 107000
```

Il valore della funzione obiettivo è 107000, quest'ultimo è ricavato dalle combionazioni tra carichi e vagoni, più precisamente :

- 10,8,2 sono rispettivamente le tonnellate del primo carico, caricate sui primi tre vagoni.
- 10 tonnellate del secondo carico sul terzo vagone.
- 6 tonnellate del terzo carico sul 4 vagone.

6 - Analisi sensitiva

L'analisi della sensitività ci aiuta a rispondere a domande su quanto sia sensibile la soluzione ottimale alle variazioni dei coefficienti in un modello.

```
require(dplyr)
```

```
## Caricamento del pacchetto richiesto: dplyr
```

```
##
## Caricamento pacchetto: 'dplyr'
```

```
## I seguenti oggetti sono mascherati da 'package:stats':
##
##     filter, lag
```

```
## I seguenti oggetti sono mascherati da 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
require(tidyr)
```

```
## Caricamento del pacchetto richiesto: tidyr
```

```
printSensitivityObj <- function(model){
  options(scipen=999)
  arg.obj = get.sensitivity.obj(model)

  numRows <- length(arg.obj$objfrom)
  symb <- c()
  for (i in c(1:numRows)) symb[i] <- paste("C", i, sep = "" )

  obj <- data.frame(Objs = symb, arg.obj)

  obj<-
    obj %>%
    mutate(objfrom=replace(objfrom, objfrom < -1.0e4, "-inf")) %>%
    mutate(objtill=replace(objtill, objtill > 1.0e4, "inf")) %>%
    unite(col = "Sensitivity",
          objfrom, Objs, objtill ,
          sep = " <= ", remove = FALSE) %>%
    select(c("Objs","Sensitivity"))
  print(obj)
}
printSensitivityObj(model)
```

```
##    Objs                   Sensitivity
## 1    C1 3500 <= C1 <= 4833.33333333333
## 2    C2 3500 <= C2 <= 4833.33333333333
## 3    C3             3500 <= C3 <= 3500
## 4    C4             -inf <= C4 <= 3500
## 5    C5             -inf <= C5 <= 2500
## 6    C6             -inf <= C6 <= 2500
## 7    C7             2500 <= C7 <= 2500
## 8    C8             -inf <= C8 <= 2500
## 9    C9             -inf <= C9 <= 2000
## 10  C10            -inf <= C10 <= 2000
## 11  C11            -inf <= C11 <= 2000
## 12  C12            2000 <= C12 <= 2500
```

La prima analisi sensitiva viene svolta sulle 12 variabili decisioni con i corrispondenti coefficenti e quindi sulla funzione obiettivo.

Dai risultati ottenuti osserviamo la variazione dei coefficenti che permettono alla soluzione ottimale di rimanere costante. I coefficenti vengono rappresentati dalle lettere C per esempio C1 rappresenta il profitto per ogni tonnelata del primo carico sul primo vagone e così via.

Notiamo come molti coefficenti possono solamente decrescere o altri che rimangono costanti.

```
get.dual.solution(lprec = model)
```

```
##  [1]    1    0    0    0    0 2000 2000 2000 2000 1500  500    0    0    0    0
## [16]    0    0    0    0    0    0    0    0    0
```

Calcoliamo il 'prezzo ombra' che indica di quanto cambia la funzione obiettivo al cambiare di un'unità della variabile decisionale. Si osserva che alcuni vincoli hanno 'prezzi ombra' diversi da zero e che i vincoli attivi corrispondono a quest'ultimi.

In totale quindi osserviamo 6 vincoli attivi.

```
printSensitivityRHS <- function(model){
  options(scipen = 999)
  arg.rhs <- get.sensitivity.rhs(model)
  numRows <- length(arg.rhs$duals)
  symb <- c()
  for (i in c(1:numRows)) symb[i] <- paste("B", i, sep = "")

  rhs <- data.frame(rhs = symb, arg.rhs)

  rhs <- rhs %>%
    mutate(dualsfrom=replace(dualsfrom, dualsfrom < -1.0e4, "-inf")) %>%
    mutate(dualstill=replace(dualstill, dualstill > 1.0e4, "inf")) %>%
    unite(col = "Sensitivity",
          dualsfrom,
          rhs,
          dualstill ,
          sep = " <= ", remove = FALSE) %>%
    select(c("rhs","Sensitivity"))
  colnames(rhs)[1] <- c('Rhs')
  print(rhs)
}
printSensitivityRHS(model)
```

```
##    Rhs        Sensitivity
## 1   B1 3000 <= B1 <= 5000
## 2   B2 2400 <= B2 <= 4000
## 3   B3  -inf <= B3 <= inf
## 4   B4  -inf <= B4 <= inf
## 5   B5     10 <= B5 <= 10
## 6   B6       8 <= B6 <= 8
## 7   B7      6 <= B7 <= 12
## 8   B8    0 <= B8 <= 6.25
## 9   B9     20 <= B9 <= 26
## 10 B10    10 <= B10 <= 16
## 11 B11 -inf <= B11 <= inf
## 12 B12 -inf <= B12 <= inf
## 13 B13 -inf <= B13 <= inf
## 14 B14 -inf <= B14 <= inf
## 15 B15    -10 <= B15 <= 0
## 16 B16 -inf <= B16 <= inf
## 17 B17 -inf <= B17 <= inf
## 18 B18 -inf <= B18 <= inf
## 19 B19 -inf <= B19 <= inf
## 20 B20      0 <= B20 <= 0
## 21 B21      0 <= B21 <= 0
## 22 B22      0 <= B22 <= 6
## 23 B23 -inf <= B23 <= inf
```

Infine effettuiamo l'analisi sensitiva sui rhd, dove è possibile osservare i valori di incremento e decremento dei termini noti dei vincoli, le B rappresentano i valori associati ai vincoli (B1 è associato al primo vincolo, ecc). I termini noti che dobbiamo considerare sono da B1 a B11, perchè quest'ultimo corrisponde all'ultimo vincolo del nostro modello. Da questa analisi ricaviamo diverse informazioni per esempio B9, il cui valore è 20, non può decrescere ma può aumentare di 6 affinchè il prezzoe sia di 1500.

Questions about LP

1. Can an LP model have more than one optimal solution. Is it possible for an LP model to have exactly two optimal solutions? Why or why not?

Sapendo che quando la retta della funzione obiettivo passa per un vertice la soluzione ottimale è unica, se la retta passa per due vertici le soluzioni sono multiple, ma non possono essere solo due perchè il anche il segmento che conginge i due vertici ha altre soluzioni, quindi non è possibile avere esattamente due soluzioni ottime.

2 - Are the following objective functions for an LP model equivalent? That is, if they are both used, one at a time, to solve a problem with exactly the same constraints, will the optimal values for x_1, x_2, x_3 be the same in both cases? Why or why not?

$$max 2x_1 + 3x_2 - x_3$$

$$min - 2x_1 - 3x_2 + x3$$

Se utilizziamo gli stessi valori x_1, x_2, x_3 nelle due equazioni (per esempio se sostituisco tutte e tre le x con il valore 1) , possiamo osservare che le due equauzioni sono equivalenti e l'unica differenza è il segno, infatti portano alla stessa soluzione ottima, i vincoli sono gli stessi e disegnano la stessa regione ammissibile.

3-Which of the following constraints are not linear or cannot be included as a constraint in a linear programming problem?

- $2x_1 + x2 - 3x_3 \geq 50$ E' lineare
- $2x_1 + \sqrt{x}_2 \geq 60$ Non è lineare perchè se provo a linearizzarla ottengo una dissequazione di secondo grado.
- $4x_1 -_{1/2} x_2 = 75$ E' lineare perchè è possibile moltiplicare ogni membro dell'equazione per 2 ed ottenere una funzione lineare.
- $\frac{3x_1 + 2x_2 x_1 - 3x_3}{x_1 + x_2 + x_3} \leq 0.9$ Non è lineare.
- $3x_1^2 + 7x_2 \leq 45$ Non è lineare perchè è una dissequazione di secondo grado e non ha soluzioni uniche.

Università degli Studi di Milano-Bicocca

Corso di Laurea in Data Science

Simone Farallo 889719