Assignment 2 - Integer Linear Programming

Giulia Mura

2/5/2020

Problem 1.

1. Determine the nodes that will be visited by the BB algorithm and for each of them get the upper and lower limit deduced by the algorithm in the execution.

I nodi che saranno visitati dall'algoritmo Branch-And-Bound sono: P_0, P_1, P_2, P_7, P_9, P_{11}, P_{12}.
Per il nodo P_0 abbiamo UB=16.5 e LB=9, questo significa che la soluzione ottimale intera si troverà tra questi due valori.
Per il nodo P_1 abbiamo UB=16.2 e LB=12.2 (LB ottenuto arrotondando per difetto la prima variabile con la parte frazionale più alta, quindi x_2.).
Per il nodo P_2 abbiamo UB=9 e LB=9 (nodo sul quale poi si decide di non procedere in quanto ha un upper bound inferiore all'upper e al lower bound del nodo P_1).
Per il nodo P_7 abbiamo UB=16 e LB=14.
Per il nodo P_9 abbiamo UB=16 e LB=14.
Per il nodo P_{11} abbiamo UB=14 e LB=14.
Per il nodo P_{12} abbiamo UB=13.8 e LB=9 (nodo sul quale non si prosegue in quanto ha un upper bound inferiore al lower bound del nodo P_{11}).
Infatti, secondo l'albero rappresentato la soluzione ottimale intera è pari a 14 e si trova con il nodo P_{11} con le variabili pari a $x_1 = 1; x_2 = 1; x_3 = 0; x_4 = 0$.

2. Solve the problem with an ILP solver and check the value of the objective function matches the one found at point 1.

$max : 9x_1 + 5x_2 + 6x_3 + 4x_4$
$s.t.$
$6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10$
$x_3 + x_4 \leq 1$
$-x_1 + x_3 \leq 0$
$-x_2 + x_4 \leq 0$
$x_1, x_2, x_3, x_4 \in \{0, 1\}$

Creo il modello

```
model = make.lp(0,4, verbose = "full")
name.lp(model, 'Max')
lp.control(model, sense = "max")
```

Creo la funzione obiettivo

```
set.objfn(model, obj = c(9, 5, 6, 4))
```

Inserisco le constraints

```
add.constraint(model,
               xt = c(6,3,5,2),
               type = "<=", rhs = 10,
               indices = 1:4)
add.constraint(model,
               xt = c(1,1),
               type = "<=", rhs = 1,
               indices = 3:4)
add.constraint(model,
               xt = c(-1,1),
               type = "<=", rhs = 0,
               indices = c(1,3))
add.constraint(model,
               xt = c(-1,1),
               type = "<=", rhs = 0,
               indices = c(2,4))
set.bounds(model,lower=c(0,0,0,0))
set.type(model, c(1:4), "binary")
```

Risolvo il modello

```
solve(model)
```

```
##
## Model name:  'Max' - run #1
## Objective:   Maximize(R0)
##
## SUBMITTED
## Model size:        4 constraints,       4 variables,           10 non-zeros.
## Sets:                                   0 GUB,                  0 SOS.
##
##
## CONSTRAINT CLASSES
## General BIN        3
## Set cover          1
##
## Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
## The primal and dual simplex pricing strategy set to 'Devex'.
##
## Optimal solution with dual simplex at iter            2.
##
## Relaxed solution                16.5 after          2 iter is B&B base.
##
## Feasible solution                 14 after          7 iter,         6 nodes (gap 11.8)
## get_ptr_sensitivity_objex: Sensitivity unknown
##
## Primal objective:
##
##   Column name                        Value   Objective         Min         Max
##   --------------------------------------------------------------------------
##   C1                                     9           9           0           0
##   C2                                     5           5           0           0
```

```
##   C3                               6          0          0          0
##   C4                               4          0          0          0
##
## get_ptr_sensitivity_rhs: Sensitivity unknown
## Primal variables:
##
##   Column name                    Value      Slack        Min        Max
##   --------------------------------------------------------------------
##   C1                                1          0          0          0
##   C2                                1          0          0          0
##   C3                                0          0          0          0
##   C4                                0          0          0          0
##
## Dual variables:
##
##   Row name                       Value      Slack        Min        Max
##   --------------------------------------------------------------------
##   R1                                0          9          0          0
##   R2                                0          0          0          0
##   R3                                0         -1          0          0
##   R4                                0         -1          0          0
##
##
## Optimal solution                  14 after         10 iter,         8 nodes (gap 11.8).
##
## Excellent numeric accuracy ||*|| = 0
##
##  MEMO: lp_solve version 5.5.2.0 for 64 bit OS, with 64 bit LPSREAL variables.
##       In the total iteration count 10, 2 (20.0) were bound flips.
##       There were 4 refactorizations, 0 triggered by time and 0 by density.
##        ... on average 2.0 major pivots per refactorization.
##       The largest [LUSOL v2.2.1.0] fact(B) had 10 NZ entries, 1.0x largest basis.
##       The maximum B&B level was 5, 0.6x MIP order, 5 at the optimal solution.
##       The constraint matrix inf-norm is 6, with a dynamic range of 6.
##       Time to load data was 0.037 seconds, presolve used 0.000 seconds,
##        ... 0.001 seconds in simplex solver, in total 0.038 seconds.
```

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

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

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

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

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

Problem 2.

SunNet is a residential Internet Service Provider (ISP) in the central Florida area. Presently, the company operates one centralized facility that all of its clients call into for Internet access.
To improve service, the company is planning to open three satellite offices in the cities of Pine Hills, Eustis, and Sanford. The company has identified five different regions to be serviced by these three offices. The following table summarizes the number of customers in each region, the service capacity at each office, and the monthly average cost per customer for providing the service to each region from each office. Table entries

of “n.a.” indicate infeasible region-to-service center combinations.
SunNet would like to determine how many customers from each region to assign to each service center to minimize the total cost.

1. Draw a network flow model to represent this problem.

```
knitr::include_graphics("/Users/giulia/NetFlow.png")
```

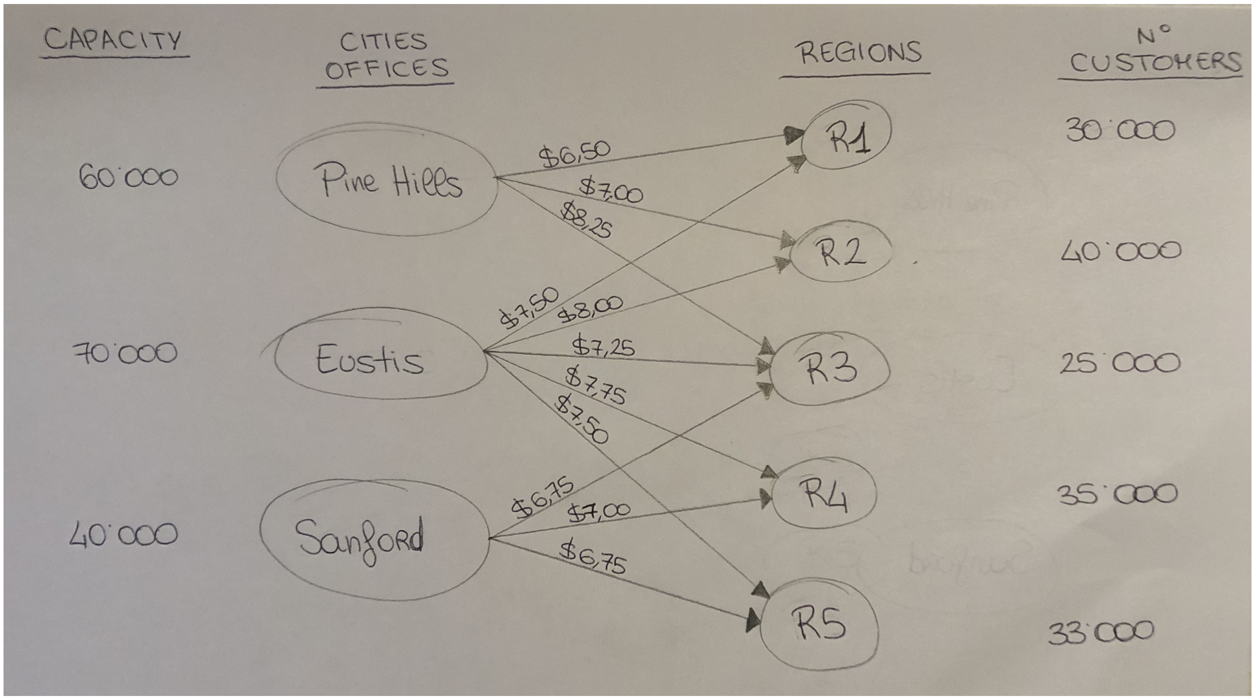

Figure 1: Network Flow

2. Implement your model and solve it.

The decision variables

x_1 = service from Pine Hills to region 1
x_2 = service from Pine Hills to region 2
x_3 = service from Pine Hills to region 3
x_4 = service from Eustis to region 1
x_5 = service from Eustis to region 2
x_6 = service from Eustis to region 3
x_7 = service from Eustis to region 4
x_8 = service from Eustis to region 5
x_9 = service from Sanford to region 3
x_{10} = service from Sanford to region 4
x_{11} = service from Sanford to region 5

The objective function

$min: 6.50x_1 + 7x_2 + 8.25x_3 + 7.50x_4 + 8x_5 + 7.25x_6 + 7.75x_7 + 7.50x_8 + 6.75x_9 + 7x_{10} + 6.75x_{11}$

The constraints

Costo massimo PineHills: $x_1 + x_2 + x_3 \leq\ 60000$
Costo massimo Eustis: $x_4 + x_5 + x_6 + x_7 + x_8 \leq\ 70000$
Costo massimo Sanford: $x_9 + x_{10} + x_{11} \leq\ 40000$

Consumatori per la regione 1: $x_1 + x_4 = 30000$
Consumatori per la regione 2: $x_2 + x_5 = 40000$
Consumatori per la regione 3: $x_3 + x_6 + x_9 = 25000$
Consumatori per la regione 4: $x_7 + x_{10} = 35000$
Consumatori per la regione 5: $x_8 + x_{11} = 33000$

Nonnegativity: $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11} \geq\ 0$

```
edgelist <- data.frame(from = c("PineHills","PineHills","PineHills","Eustis","Eustis","Eustis","Eustis"
                       to = c("R1","R2","R3","R1","R2","R3","R4","R5","R3","R4","R5"),
                       cost = c(6.50, 7 , 8.25, 7.50, 8, 7.25, 7.75, 7.50, 6.75, 7, 6.75))

g <- graph_from_edgelist(as.matrix(edgelist[,c('from','to')]))

E(g)$cost <- edgelist$cost

edgelist
```

```
##         from to cost
## 1  PineHills R1 6.50
## 2  PineHills R2 7.00
## 3  PineHills R3 8.25
## 4     Eustis R1 7.50
## 5     Eustis R2 8.00
## 6     Eustis R3 7.25
## 7     Eustis R4 7.75
## 8     Eustis R5 7.50
## 9    Sanford R3 6.75
## 10   Sanford R4 7.00
## 11   Sanford R5 6.75
```

Creo il modello

```
modello = make.lp(0,11)
lp.control(modello, sense = "min")
set.objfn(modello,obj = edgelist$cost)
```

Inserisco le constraints

```
add.constraint(modello,
               xt=c(1,1,1),
               type = "<=", rhs = 60000,
               indices= 1:3)

add.constraint(modello,
               xt=c(1,1,1,1,1),
               type = "<=", rhs = 70000,
               indices = 4:8)

add.constraint(modello,
               xt=c(1,1,1),
```

```r
               type = "<=", rhs = 40000,
               indices = 9:11)

add.constraint(modello,
               xt=c(1,1),
               type = "=", rhs = 30000,
               indices = c(1,4))

add.constraint(modello,
               xt=c(1,1),
               type = "=", rhs = 40000,
               indices = c(2,5))

add.constraint(modello,
               xt=c(1,1,1),
               type = "=", rhs = 25000,
               indices = c(3,6,9))

add.constraint(modello,
               xt=c(1,1),
               type = "=", rhs = 35000,
               indices = c(7,10))

add.constraint(modello,
               xt=c(1,1),
               type = "=", rhs = 33000,
               indices = c(8,11))

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

Risolvo il modello.

```r
solve(modello)
```

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

```r
get.objective(modello)
```

```
## [1] 1155000
```

```r
get.variables(modello)
```

```
##  [1] 20000 40000     0 10000     0 25000     0 28000     0 35000  5000
```

3. What is the optimal solution?

La soluzione ottimale risulta avere le seguenti variabili: $x_1 = 20000; x_2 = 40000; x_3 = 0; x_4 = 10000; x_5 = 0; x_6 = 25000; x_7 = 0; x_8 = 28000; x_9 = 0; x_{10} = 35000; x_{11} = 5000$ e il risultato è pari a 1.155.000.