Assignement 2

Sofia Davoli 813479

30/4/2020

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.*

B&B algorithm start from P_0, and split for x_1. if x_1=0 then z_{best}=9, the algorithm consider then the other split relative to x_2, x_3 but the solutions z_{best}=9 have no improvement or is not feasible. So in the left part B&B stop at P_2.

if x_1=1, then z=16.2 but the constraint are not satisfied, though other split are require for x_2 and x_3. In the split of x_2, if x_2=0, z_{best} have no improvement, if z_2=1, z_{best} change to 16, the algorithm descend in this direction until the constraint of integrality are satisfied.

B&B stop in P_{11} where z_{best}=14, x=(1,1,0,0) resulting in being the best solution for the problem.

Upper limit is obtain considering the integer part of the number in the previous split. (ex 17.8–>17) Lower limit is obtain considering the possible minimun value the function can obtain in that precise split, with the previously assigned variables. (ex considering the objective function $9x_1$ + $5x_2$ + $6x_{3} + $4x_4$ in split P_{12} assigned variables are (x_1=1,x_2=0), other variables can still vary from 1-0. The Lowes limit is equal to 9x1 +5x0+6x0+4x0= 9)

limit for the split of the B&B algorithm are:

-P_1: upper 16 lower 9

-P_2: upper 16 lower 0

-P_{12}: upper 16 lower 9

-P_{14}: upper 13 lower 9

-P_7:upper 16 lower 14

-P_9 upper 16 lower 14

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

solving ILP problem with 0 constraint and 4 variables.

```
model = make.lp(0,4) # 0 constraints, 4 variables
lp.control(model, sense="max") # original maximization problem
```

```r
set.objfn(model,obj=c(9,5,6,4)) # definition of the objective function using profit coef

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


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


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

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


set.bounds(model, lower= c(0,0,0,0))# non negativity
set.type(model, c(1:4), "binary")
```

Solution

```r
solve(model)
```

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

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

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

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

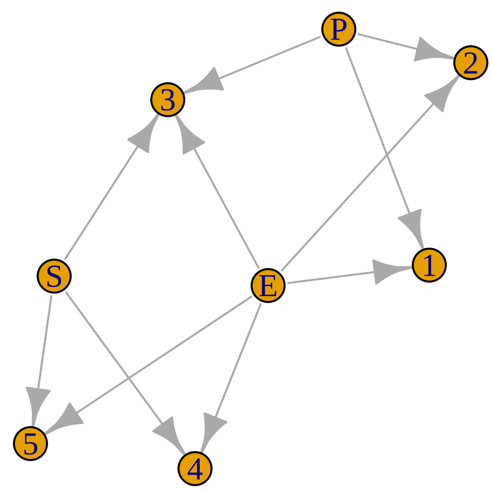
```
## [1] 14
```

The solution sorrespond to the one founded before.

Problem 2

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

```
el <- matrix( c("P","1",6.50, "E","1",7.50, "P","2",7,
"E","2",8, "P", "3", 8.25, "E","3", 7.25, "S", "3", 6.75, "E","4", 7.75, "S", "4", 7, "E","5", 7.50, "S
nc = 3, byrow = TRUE)
g <- graph_from_edgelist(el = el[,1:2], directed = T)
edge_attr(g, "weight") <- el[,3]
plot(g, layout=layout_with_graphopt)
```



The length of the vector is proportional to it's cost.

2 *Implement your model and solve it.*

```
edges= data.frame(index_i=c("P", "E", "P","E", "P","E", "S", "E", "S", "E", "S"),
 index_j =c(1,1,2,2,3,3,3,4,4,5,5),
 lb =c(0,0,0,0,0,0,0,0,0,0,0),
 ub =c(Inf,Inf,Inf,Inf,Inf,Inf,Inf,Inf,Inf,Inf,Inf),
 cost =c(6.50, 7.5, 7, 8, 8.25, 7.25, 6.75, 7.75, 7, 7.50, 6.75)) #variables DF
b = c(30000, 40000, 25000, 35000, 33000) #customer per region
model = make.lp(0,11) #possibles linkes
```

```r
lp.control(model, sense="min") #minimization problem
set.objfn(model,obj=edges$cost)

#constraint

set.bounds(model,lower=edges$lb,upper=edges$ub)

#link constraints
add.constraint(model,
 xt=c(1,1),
 type=">=",rhs=b[1],
 indices=c(1,2))

add.constraint(model,
 xt=c(1,1),
 type=">=",rhs=b[2],
 indices=c(3,4))

add.constraint(model,
 xt=c(1,1,1),
 type=">=",rhs=b[3],
 indices=c(5,6,7))

add.constraint(model,
 xt=c(1,1),
 type=">=",rhs=b[4],
 indices=c(8,9))

add.constraint(model,
 xt=c(1,1),
 type=">=",rhs=b[5],
 indices=c(10,11))

#capacity constraints
add.constraint(model,
 xt=c(1,1,1),
 type="<=",rhs=60000,
 indices=c(1,3,5))

add.constraint(model,
 xt=c(1,1,1,1,1),
 type="<=",rhs=70000,
 indices=c(2,4,6,8,10))

add.constraint(model,
 xt=c(1,1,1),
 type="<=",rhs=40000,
 indices=c(7,9,11))


set.type(model, c(1:11),type="integer") #model type
```

```r
solve(model)
```

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

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

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

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

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

3 *What is the optimal solution?*

```r
edges <- cbind(edges,y_val = get.variables(model))
edges[c("index_i", "index_j", "y_val")]
```

```
##    index_i index_j y_val
## 1        P       1 20000
## 2        E       1 10000
## 3        P       2 40000
## 4        E       2     0
## 5        P       3     0
## 6        E       3 25000
## 7        S       3     0
## 8        E       4     0
## 9        S       4 35000
## 10       E       5 28000
## 11       S       5  5000
```