Integer Programming Exercise

Maddukuri Janakisrija

#This rmd file contains the code for assignment 6. #The purpose of this assignment is to explore the integer programming formulations and solutions.

```
library(lpSolveAPI)
```

By definition, integer programming is Mixed integer programming refers to a model in which just part of the variables must have integer values (MIP). Binary integer programming refers to IP issues using just binary variables (BIP).

1. Consider the following activity-on-arc project network, where the 12 arcs (arrows) represent the 12 activities (tasks) that must be performed to complete the project and the network displays the order in which the activities need to be performed. The number next to each arc (arrow) is the time required for the corresponding activity. Consider the problem of finding the longest path (the largest total time) through this network from start (node 1) to finish (node 9), since the longest path is the critical path.

The longest path is the critical path and objective function is given by

Amax = 3B13 + 5B12 + 3B35 + 2B25 + 2B58 + 4B57 + 4B47 + 1B46 + 7B89 + 4B79 + 5B69

where Bij(i = starting node, j= ending node)

starting node: 3B13 + 5B12 =1

Intermediate nodes: 5B12 - 2B25 - 4B24 =0 3B13 - 3B35 =0 4B24 - 1B46 - 4B47 = 0 3B35 + 2B25 - 2B58 - 6B57 = 0 1B46 - 5B69 = 0 6B57 + 4B47 - 4B79 = 0 2B58 - 7B89 = 0

Ending node : 7B89 + 4B79 + 5B69 = 1 Where Bij are binary

R program:

The longest path is the critical path between the nodes (1-2-5-7-9)

```
lprec <- make.lp(nrow = 9, ncol = 12)

#where nrow is the number of nodes and ncol is the number of arcs

lp.control(lprec, 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"
```

```
# creating names for arcs and nodes
arc <- c("b12", "b13", "b24", "b25", "b35", "b46", "b47", "b57", "b58", "b69", "b79",
"b89")

node <- c("node1", "node2", "node3", "node4", "node5", "node6", "node7", "node8", "no
de9")

# rename the IP object
rownames(lprec) <- node
colnames(lprec) <- arc

# objective function
time <- c(5, 3, 4, 2, 3, 1, 4, 6, 2, 5, 4, 7)
set.objfn(lprec, 1* time)

# node 1 is the "starting node"
set.row(lprec, 1, c(1,1), indices = c(1,2))

# node 2:8 is the "intermediate node"
set.row(lprec, 2, c(1,-1,-1), indices = c(1,3,4))
set.row(lprec, 3, c(1,-1), indices = c(2,5))
set.row(lprec, 4, c(1,-1,-1), indices = c(3,6,7))
set.row(lprec, 5, c(1,1,-1,-1), indices = c(4,5,8,9))
set.row(lprec, 6, c(1, -1),indices = c(6,10))
set.row(lprec, 7, c(1,1,-1), indices = c(7,8,11))
set.row(lprec, 8, c(1,-1), indices = c(9,12))

# node 9  is the "finish node"
set.row(lprec, 9, c(1,1,1), indices = c(10,11,12))

# set the constraints type
set.constr.type(lprec, rep("="), 9)

# set constraint to the RHS
rhs <- c(1, rep(0, 7), 1)
set.rhs(lprec, rhs)

# set all variables type to be binary
set.type(lprec, 1:12, "binary")
write.lp(lprec, "netlp.lp", "lp")

# solve
solve(lprec)
```

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

```
#get objective value
get.objective(lprec)
```

```
## [1] 17
```

```
#get values of decision variables
get.variables(lprec)
```

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

```
#get constraints to the rhs value
get.constraints(lprec)
```

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

```
cbind(arc, get.variables(lprec))
```

```
##       arc
##  [1,] "b12" "1"
##  [2,] "b13" "0"
##  [3,] "b24" "0"
##  [4,] "b25" "1"
##  [5,] "b35" "0"
##  [6,] "b46" "0"
##  [7,] "b47" "0"
##  [8,] "b57" "1"
##  [9,] "b58" "0"
## [10,] "b69" "0"
## [11,] "b79" "1"
## [12,] "b89" "0"
```

The critical path which is the maximum objective function is 17

2. Selecting an Investment Portfolio An investment manager wants to determine an opti- mal portfolio for a wealthy client. The fund has $2.5 million to invest, and its objective is to maximize total dollar return from both growth and dividends over the course of the coming year. The client has researched eight high-tech companies and wants the portfolio to consist of shares in these firms only. Three of the firms (S1 – S3) are primarily software companies, three (H1–H3) are primarily hardware companies, and two

(C1–C2) are internet consulting companies. The client has stipulated that no more than 40 percent of the investment be allocated to any one of these three sectors. To assure diversification, at least $100,000 must be invested in each of the eight stocks. Moreover, the number of shares invested in any stock must be a multiple of 1000.

1. Determine the maximum return on the portfolio. What is the optimal number of shares to buy for each of the stocks? What is the corresponding dollar amount invested in each stock?

According to the returns of the problem can be given by

Returns = (Price per share) * (Growth rate of share) + (Dividend per share)

Hence the objective function is

Amax = 4Bs1 + 6.5Bs2 + 5.9Bs3 + 5.4Bh1 + 5.15Bh2 + 10Bh3 + 8.4Bc1 + 6.25Bc2

Constraints: Investment constraint:

40Bs1 + 50Bs2 + 80Bs3 + 60Bh1 + 45Bh2 + 60Bh3 + 30Bc1 + 25Bc2 <= 2500000

The number of shares invested in any stock must be a multiple of 1000

1000Bs1 >= 0; 1000Bs2 >= 0; 1000Bs3 >= 0 1000Bh1 >= 0; 1000Bh2 >= 0; 1000Bh3 >= 0 1000Bc1 >= 0; 1000Bc2 >= 0

Atleast $100,000 must be invested in each of the eight stocks

40Bs1 >= 100000; 50Bs2 >= 100000; 80Bs3 >= 100000; 60Bh1 >= 100000; 45Bh2 >= 100000; 60Bh3 >= 100000; 30Bc1 >= 100000; 25Bc2 >= 100000;

No more than 40 percent of the investment constraints

40Bs1 + 50Bs2 + 80Bs3 <= 1000000 60Bh1 + 45Bh2 + 60Bh3 <= 1000000 30Bc1 + 25Bc2 <= 1000000

where Bsj, Bhj, Bcj >= 0 are integers.

Using lpsolve with integer restriction we get the objective function, maximum returns as 487145.2

number of stocks are s1 = 2500, s2 = 6000, ss3 = 1250; h1 = 1667, h2 = 2223, h3 = 3332; c1 = 30000, c2 = 4000;

The amount invested in each stock s1 = 100000, s2 = 300000, s3 = 100000; h1 = 100020, h2 = 100035, h3 = 799920; c1 = 900000, c2 = 100000;

Using lpsolve with integer restriction

```
library(lpSolveAPI)
lprec <- make.lp(0,8)
lp.control(lprec, 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"
```

```
set.objfn(lprec, c(4,6.5, 5.9, 5.4, 5.15, 10, 8.4, 6.25))
set.type(lprec, c(1:8), type = "integer")

add.constraint(lprec, c(40,50,80,60,45,60,30,25), "<=", 2500000, indices = c(1:8))

add.constraint(lprec,1000,">=",0,indices = 1)
add.constraint(lprec,1000,">=",0,indices = 2)
add.constraint(lprec,1000,">=",0,indices = 3)
add.constraint(lprec,1000,">=",0,indices = 4)
add.constraint(lprec,1000,">=",0,indices = 5)
add.constraint(lprec,1000,">=",0,indices = 6)
add.constraint(lprec,1000,">=",0,indices = 7)
add.constraint(lprec,1000,">=",0,indices = 8)
add.constraint(lprec,40,">=",100000,indices = 1)
add.constraint(lprec,50,">=",100000,indices = 2)
add.constraint(lprec,80,">=",100000,indices = 3)
add.constraint(lprec,60,">=",100000,indices = 4)
add.constraint(lprec,45,">=",100000,indices = 5)
add.constraint(lprec,60,">=",100000,indices = 6)
add.constraint(lprec,30,">=",100000,indices = 7)
add.constraint(lprec,25,">=",100000,indices = 8)
add.constraint(lprec,c(40,50,80),"<=",500000,indices = c(1,2,3))
add.constraint(lprec,c(60,45,60),"<=",1000000,indices = c(4,5,6))
add.constraint(lprec,c(30,25),"<=",1000000,indices = c(7,8))
solve(lprec)
```

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

```
#get objective value
get.objective(lprec)
```

```
## [1] 487145.2
```

```
get.variables(lprec)
```

```
## [1]  2500  6000  1250  1667  2223 13332 30000  4000
```

```
#get constraints to the rhs value
get.constraints(lprec)
```

```
##  [1]  2499975  2500000  6000000  1250000  1667000  2223000 13332000 30000000
##  [9]  4000000   100000   300000   100000   100020   100035   799920   900000
## [17]   100000   500000   999975  1000000
```

2)Compare the solution in which there is no integer restriction on the number of shares invested. By how much (in percentage terms) do the integer restrictions alter the value of the optimal objective function? By how much (in percentage terms) do they alter the optimal investment quantities?

Using lpsolve without integer restriction we get the objective function, maximum returns = 487152.8

number of stocks: S1= 2500.0, S2= 6000.0, S3= 1250.0; H1= 1667.667, H2= 2222.222, H3= 13333.333; C1= 30000.0, C2= 4000.0;

The amount invested in each stock S1= 100000, S2= 300000, S3= 100000; H1= 100000, H2= 100000, H3= 800000; C1= 900000, C2= 100000;

the integer restrictions alter the value of the optimal objective function by the percentage 0.00156

Using lpsolve without integer restriction

```
library(lpSolveAPI)
lprec<-make.lp(0,8)
lp.control(lprec,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"
```

```
set.objfn(lprec,c(4,6.5,5.9,5.4,5.15,10,8.4,6.25))
add.constraint(lprec,c(40,50,80,60,45,60,30,25),"<=",2500000,indices = c(1:8))
add.constraint(lprec,1000,">=",0,indices = 1)
add.constraint(lprec,1000,">=",0,indices = 2)
add.constraint(lprec,1000,">=",0,indices = 3)
add.constraint(lprec,1000,">=",0,indices = 4)
add.constraint(lprec,1000,">=",0,indices = 5)
add.constraint(lprec,1000,">=",0,indices = 6)
add.constraint(lprec,1000,">=",0,indices = 7)
add.constraint(lprec,1000,">=",0,indices = 8)
add.constraint(lprec,40,">=",100000,indices = 1)
add.constraint(lprec,50,">=",100000,indices = 2)
add.constraint(lprec,80,">=",100000,indices = 3)
add.constraint(lprec,60,">=",100000,indices = 4)
add.constraint(lprec,45,">=",100000,indices = 5)
add.constraint(lprec,60,">=",100000,indices = 6)
add.constraint(lprec,30,">=",100000,indices = 7)
add.constraint(lprec,25,">=",100000,indices = 8)
add.constraint(lprec,c(40,50,80),"<=",1000000,indices = c(1,2,3))
add.constraint(lprec,c(60,45,60),"<=",1000000,indices = c(4,5,6))
add.constraint(lprec,c(30,25),"<=",1000000,indices = c(7,8))
solve(lprec)
```

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

```
#get objective value
get.objective(lprec)
```

```
## [1] 487152.8
```

```
#get values of decision variables
get.variables(lprec)
```

```
## [1]  2500.000  6000.000  1250.000  1666.667  2222.222 13333.333 30000.000
## [8]  4000.000
```

```
#get constraints to the rhs value
get.constraints(lprec)
```

```
##  [1]  2500000  2500000  6000000  1250000  1666667  2222222 13333333 30000000
##  [9]  4000000   100000   300000   100000   100000   100000   800000   900000
## [17]   100000   500000  1000000  1000000
```