# Assignment - 6

Vamshee Deepak Goud Katta

11/22/2021

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.

From the above problem, we need to find the largest time taken from start to finish.

For that, let us consider the following:

Decision variable: Xij = 1 if the arc from node i to node j is chosen in the optimal (longest) path otherwise Xij = 0

Objective Function: To maximize the total time required from node 1 to node 9.

Max: Z = ∑(Aij)(Xij)

Where, Aij = time taken by arc to go from node i to node j.
Max: Z = 5X12 + 3X13 + 3X35 + 2X25 + 4X24 + 4X47 + 1X46 + 2X58 + 6X57 + 5X69 + 4X79 + 7X89

To get the longest path or largest total time through this network from start to finish below are the objective constraints.

For origin and destination nodes 1 and 9, outgoing arc is equal to 1;

Node 1: X12 + X13 = 1

Node 9: X69 + X79 + X89 = 1

For intermediate nodes, Arc in = arc out;

Node 2: X12 = X25 + X24

Node 3: X13 = X35

```
Node 4: X24 = X46 + X47

Node 5: X25 + X35 = X57 + X58

Node 6: X46 = X69

Node 7: X57 + X47 = X79

Node 8: X58 = X89
```

## Solving the above LP Problem in R

## Reading data

```
library(lpSolve)
library(lpSolveAPI)
Arc <- read.lp("vkatta_6.lp")
lp.control(Arc, 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"
```

## Solving the LP

```
solve(Arc)

## [1] 0

get.objective(Arc)

## [1] 17

get.variables(Arc)

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

```
Based on the above solution, the longest path or the largest time taken
through the network is through the path of X12, X25, X57 and X79 which is
through the nodes 1 -> 2 -> 5 -> 7 -> 9.

The total time corresponding to this path is 17.
```

2. **Selecting an Investment Portfolio:** An investment manager wants to determine an optimal 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.

A.

1.

To determine the expected number of shares to purchase for each of the stocks:

Let us first determine the expected returns on each of the stocks across the portfolio;

Expected return of stock= D1 /P0 + g

Expected return of stock for software companies

S1 = 2(1+0.05) / 40 + 0.05 = 10.25

S2= 1.50 (1+010) /50 + 0.10 = 13.3

S3= 3.50 (1+0.03)/ 80 + 0.03 = 7.51

Expected return of stock for hardware companies

H1=3. (1+0.04)/ 60 + 0.04 = 9.2

H2 =2 (1+0.07) /45 + 0.07 = 11.76

H3= 1 (1+0.15) /60 + 0.15 = 16.92

Expected return of stock for consulting companies

C1 = 1.8 (1+0.22)/ 30 + 0.22 = 29.32

C2 = 0 (1+0.25) /25 = 0

Return on the complete portfolio if the investment is shared equally among all the companies is

(S1 + S2 + S3 + H1 + H2 + H3 + C1 + C2)/8

= (10.25 + 13.3 + 7.51 + 9.2 + 11.76 + 16.92 + 29.32 + 0) / 8

= $12.28, average return on entire portfolio.

Now,

Maximum amount invested in one of the sectors = $2.5 million*40% = 1 million

Minimum investment in each stock = $100,000 = 0.1 million
Maximum return in portfolio is given by sum of returns on each stock multiplied by investment allocated to that particular stock

= 10.25*(.1/2.5) + 13.3 *(.3/2.5) + 7.51 *(.1/2.5) + 9.2* (.1/2.5) + 11.76 * (0.1/2.5) + 16.92 * (.8/2.5) +29.32 * (.9/2.5) + 0 * (.1/2.5) = 19.11 %

Optimal number of shares to buy for each of the stocks;

S1 = 100000/40 = 2500

S2= 100000/50 = 2000

S3= 300000/80 = 3750

H1= 100000/60 = 1666.67

H2 = 100000/45 = 2222.22

H3= 800000/60 = 13333.33

## 1. Formulating and solving the LP problem with integer restriction

```
Port1 <- make.lp(0,8)
lp.control(Port1, 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(Port1, c(4,6.5,5.9,5.4,5.15,10,8.4,6.25))
set.type(Port1, c(1:8), type = "integer")

add.constraint(Port1, c(40,50,80,60,45,60,30,25),"<=",2500000,indices =
c(1:8))
add.constraint(Port1, 1000,">=",0,indices = 1)
add.constraint(Port1, 1000,">=",0,indices = 2)
add.constraint(Port1, 1000,">=",0,indices = 3)
add.constraint(Port1, 1000,">=",0,indices = 4)
add.constraint(Port1, 1000,">=",0,indices = 5)
add.constraint(Port1, 1000,">=",0,indices = 6)
add.constraint(Port1, 1000,">=",0,indices = 7)
add.constraint(Port1, 1000,">=",0,indices = 8)
add.constraint(Port1, 40,">=",100000,indices = 1)
add.constraint(Port1, 50,">=",100000,indices = 2)
add.constraint(Port1, 80,">=",100000,indices = 3)
add.constraint(Port1, 60,">=",100000,indices = 4)
add.constraint(Port1, 45,">=",100000,indices = 5)
add.constraint(Port1, 60,">=",100000,indices = 6)
add.constraint(Port1, 30,">=",100000,indices = 7)
add.constraint(Port1, 25,">=",100000,indices = 8)
add.constraint(Port1, c(40,50,80),"<=",1000000,indices = c(1,2,3))
add.constraint(Port1, c(60,45,60),"<=",1000000,indices = c(4,5,6))
add.constraint(Port1, c(30,25),"<=",1000000,indices = c(7,8))

solve(Port1)
```

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

```
get.objective(Port1)
```

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

```
get.variables(Port1)
```

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

```
get.constraints(Port1)
```

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

## 2. Formulating and solving LP problem without integer restriction

```
Port2<-make.lp(0,8)
lp.control(Port2,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(Port2,c(4,6.5,5.9,5.4,5.15,10,8.4,6.25))

add.constraint(Port2, c(40,50,80,60,45,60,30,25),"<=",2500000,indices =
c(1:8))
add.constraint(Port2, 1000,">=",0,indices = 1)
add.constraint(Port2, 1000,">=",0,indices = 2)
add.constraint(Port2, 1000,">=",0,indices = 3)
add.constraint(Port2, 1000,">=",0,indices = 4)
add.constraint(Port2, 1000,">=",0,indices = 5)
add.constraint(Port2, 1000,">=",0,indices = 6)
add.constraint(Port2, 1000,">=",0,indices = 7)
add.constraint(Port2, 1000,">=",0,indices = 8)
add.constraint(Port2, 40,">=",100000,indices = 1)
add.constraint(Port2, 50,">=",100000,indices = 2)
add.constraint(Port2, 80,">=",100000,indices = 3)
add.constraint(Port2, 60,">=",100000,indices = 4)
add.constraint(Port2, 45,">=",100000,indices = 5)
add.constraint(Port2, 60,">=",100000,indices = 6)
add.constraint(Port2, 30,">=",100000,indices = 7)
add.constraint(Port2, 25,">=",100000,indices = 8)
add.constraint(Port2, c(40,50,80),"<=",1000000,indices = c(1,2,3))
add.constraint(Port2, c(60,45,60),"<=",1000000,indices = c(4,5,6))
add.constraint(Port2, c(30,25),"<=",1000000,indices = c(7,8))

solve(Port2)
```

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

```
get.objective(Port2)

## [1] 487152.8

get.variables(Port2)

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

get.constraints(Port2)

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



The change in percentage of optimal objective function with change in
restriction is 487145.2/487152.8 *100 = 0.0016%
```