SCA1 - Assignment 2

Thanh Tran Viet-h12430817, Krishna Varun Varma Kalidindi-h12438206, Iryna Ustych-h12427988

16 October 2024

Problem 1

Question a

Sets:
I: the set of product types, $I = \{1, 2, 3\}$
J: the set of manufacturing departments, $J = \{A, B, C\}$

Indices:
i: product type, $i \in I$
j: manufacturing department, $j \in J$

Parameters:
p_i: the profit contribution per unit of product type i
C_j: the labour hour capacity of manufacturing department j
c_{ij}: the labour hour used for producing a unit of product type i at manufacturing department j

Decision variables:
x_i: the quantity of each product type i to be produced

Objective function:
Maximizing total profit contribution

$$Max \sum_{i \in I} p_i x_i$$

Constraints:
For each manufacturing department j, the total labour hour used must not exceed the labour hour capacity:

$$\sum_{i \in I} c_{ij} x_i \leq C_j, \forall j \in J$$

For each product type i, the produced quantity must be non-negative:

$$x_i \geq 0, \forall i \in I$$

Question b

```
# Clear enviroment
rm(list = ls())

# Import library
library("lpSolve")

# Decision varible coefficients
```

```r
c1 <- c(22, 25, 28)

# Constraint matrix
A1 <- matrix(c(1.5, 3, 2,
               2, 1, 2.5,
               0.25, 0.25, 0.25),
             nrow = 3,
             ncol = 3,
             byrow = TRUE)

# Constraint direction
dir1 <- c(rep("<=",3))

# Constraint RHS
b1 <- c(425, 375, 55)

# Model
model1 <- lp(direction = "max",
             objective.in = c1,
             const.mat = A1,
             const.dir = dir1,
             const.rhs = b1,
             all.int = TRUE)

# Print result
for (i1 in 1:length(model1$solution)) {
  print(paste("The quantity of product type",
              i1,
              "that should be produced is",
              model1$solution[i1]))
}
```

```
## [1] "The quantity of product type 1 that should be produced is 154"
## [1] "The quantity of product type 2 that should be produced is 64"
## [1] "The quantity of product type 3 that should be produced is 1"
```

```r
print(paste("The projected total profit contribution is",
            model1$objval))
```

```
## [1] "The projected total profit contribution is 5016"
```

Question c

s_i: the setup cost of product type i

When the setup costs associate, the total profit contribution decreases

```r
# Setup cost
setup_cost1 <- c(400, 525, 550)

actual_setup_cost1 <- c()

for (i1 in 1:length(model1$solution)) {
  if (model1$solution[i1] >= 0) {actual_setup_cost1[i1] <- setup_cost1[i1]}
  else {actual_setup_cost1[i1] <- 0}
}
```

```r
# Calculate revised total profit
revised_total_profit1 <- model1$objval - sum(actual_setup_cost1)

# Print result
print(paste("The total profit contribution after taking into account the setup costs is",
            revised_total_profit1))
```

```
## [1] "The total profit contribution after taking into account the setup costs is 3541"
```

Question d

Sets:
I: the set of product types, $I = \{1, 2, 3\}$
J: the set of manufacturing departments, $J = \{A, B, C\}$

Indices:
i: product type, $i \in I$
j: manufacturing department, $j \in J$

Parameters:
p_i: the profit contribution per unit of product type i
C_j: the labour hour capacity of manufacturing department j
c_{ij}: the labour hour used for producing a unit of product type i at manufacturing department j
s_i: the setup cost of product type i
w_i: the maximum production quantity of product type i if it is decided to produce

Decision variables:
x_i: the quantity of each product type i to be produced y_i: the decision of producing product type i or not, $y_i \in \{0, 1\}$

Objective function:
Maximizing total profit contribution

$$Max \sum_{i \in I} p_i x_i - \sum_{i \in I} s_i y_i$$

Constraints:
For each manufacturing department j, the total labour hour used must not exceed the labour hour capacity:

$$\sum_{i \in I} c_{ij} x_i \leq C_j, \forall j \in J$$

For each product type i, the produced quantity must not exceed the maximum production quantity:

$$x_i - W_i y_i \leq 0, \forall i \in I$$

For each product type i, the produced quantity must be non-negative:

$$x_i \geq 0, \forall i \in I$$

Question e

```r
# Decision variable coefficient
c2 <- c(22, 25, 28, -400, -525, -550)

# Constraint matrix
```

```r
A2 <- matrix(c(1.5, 3, 2, rep(0,3),
              2, 1, 2.5, rep(0,3),
              0.25, 0.25, 0.25, rep(0,3),
              1,0,0,-185,0,0,
              0,1,0,0,-125,0,
              0,0,1,0,0,-140),
              nrow = 6,
              ncol = 6,
              byrow = TRUE)

# Constraint direction
dir2 <- c(rep("<=",6))

# Constraint RHS
b2 <- c(425, 375, 55, rep(0,3))

# Model
model2 <- lp(direction = "max",
            objective.in = c2,
            const.mat = A2,
            const.dir = dir2,
            const.rhs = b2,
            int = 1:3,
            bin = 4:6)

# Print result
for (i2 in 1:length(model2$solution[1:3])) {
  print(paste("The quantity of product type",
              i2,
              "that should be produced is",
              model2$solution[i2]))
}
```

```
## [1] "The quantity of product type 1 that should be produced is 155"
## [1] "The quantity of product type 2 that should be produced is 64"
## [1] "The quantity of product type 3 that should be produced is 0"
```

```r
print(paste("The projected total profit contribution is", model2$objval))
```

```
## [1] "The projected total profit contribution is 4085"
```

```r
print(paste("Comparing the new total profit of",
            model2$objval,
            "versus the previous one of",
            revised_total_profit1,
            ", it is higher by",
            model2$objval-revised_total_profit1))
```

```
## [1] "Comparing the new total profit of 4085 versus the previous one of 3541 , it is higher by 544"
```

This gap can be explained as in question c solution, product type 3 was only produced by 1 unit which earned 28 profit but costed 550 of fixed setup cost, as a result, total earning for producing product type 3 was -522. Therefore, in the new solution, to ignore that loss, only product type 1 and 2 was produced.

Problem 2

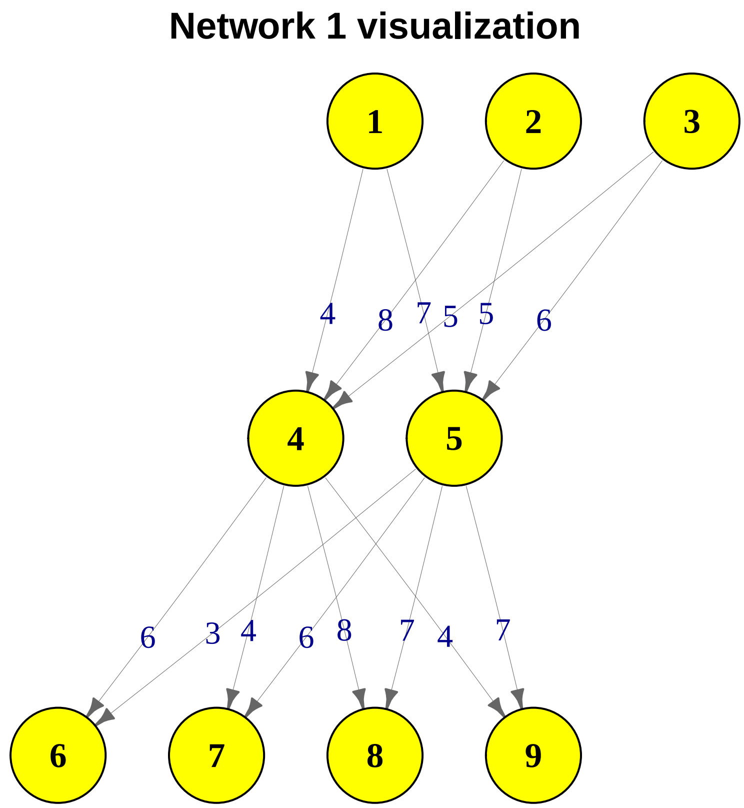
Question a

```r
# Clear enviroment
rm(list = ls())

# Import library
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```r
library(lpSolve)
library(slam)

# Plants to warehouses cost
plant1_cost1 <- data.frame(i = rep(1,2), j = c(4,5), cost = c(4,7))
plant2_cost1 <- data.frame(i = rep(2,2), j = c(4,5), cost = c(8,5))
plant3_cost1 <- data.frame(i = rep(3,2), j = c(4,5), cost = c(5,6))

# Warehouses to customers cost
wh1_cost1 <- data.frame(i = rep(4,4), j = c(6,7,8,9), cost = c(6,4,8,4))
wh2_cost1 <- data.frame(i = rep(5,4), j = c(6,7,8,9), cost = c(3,6,7,7))

# Edges list
edge_list1 <- rbind(plant1_cost1, plant2_cost1, plant3_cost1, wh1_cost1, wh2_cost1)
edge_list1 <- cbind(edge = 1:nrow(edge_list1), edge_list1)

# Create graph
g1 <- graph_from_data_frame(d = edge_list1[, c("i", "j", "cost")], directed = TRUE)

# Adjust display margin
par(mar = c(1, 1, 1, 1))

# Plot the graph
plot(g1,
     layout = layout_as_tree(g1),
     vertex.color=c("Yellow"),
     vertex.size = 30,
     vertex.label.font=2,
     vertex.label.color = 'black',
     vertex.label.cex=1.1,
     edge.color="gray40",
     edge.width= E(g1)$cost/100,
     edge.label = E(g1)$cost,
     edge.label.cex = 1,
     edge.label.font = 1,
```

```
    edge.curved = 0.01,
    edge.arrow.size=.5,
    main = 'Network 1 visualization'
)
```



Question b

Sets:
V: Set of all nodes in the network, $V = \{1, 2, ..., 9\}$
I: Set of origin nodes, $I = \{1, 2, 3\}$
T: Set of transshipment nodes, $T = \{4, 5\}$
J: Set of destination nodes, $J = \{6, 7, 8, 9\}$
A: Set of directed edges between nodes, $(i, k), (k, j) \in A, \forall i, j, k \in V$

Indices:
i: origin nodes, $i \in I$
k: transshipment nodes, $k \in T$
j: destination nodes, $j \in J$

Parameters:
c_{ik}: the delivery cost per unit from node i to node k
c_{kj}: the delivery cost per unit from node k to node j
s_i: the supply capacity of node i
d_j: the demand of node j

Decision variables:

x_{ik}: the delivery quantity from node i to node k
x_{kj}: the delivery quantity from node k to node j

Objective function:
Minimizing the total delivery cost

$$Min \sum_{(i,k)\in A} c_{ik}x_{ik} + \sum_{(k,j)\in A} c_{kj}x_{kj}$$

Constraints:
For each origin node i, the total outbound quantity must not exceed the capacity:

$$\sum_{(i,k)\in A} x_{ik} \leq s_i, \forall i \in I$$

For each destination node j, the total inbound quantity must fulfill the demand:

$$\sum_{(k,j)\in A} x_{kj} \geq d_j, \forall j \in J$$

For each transshipment node k, the inbound and outbound quantity must be equal:

$$\sum_{(i,k)\in A} x_{ik} - \sum_{(k,j)\in A} x_{kj} = 0, \forall k \in T$$

For each edge (i,k) and (k,j), the delivery quantity must be non-negative:

$$x_{ik} \geq 0, x_{kj} \geq 0, \forall (i,k),(k,j) \in A$$

Question c

```
# delivery cost on each edge
c1 <- edge_list1$cost

# constraints for plants
const1_1 <- data.frame(i = c(1,1), j = c(1,2), v = c(1,1))
const2_1 <- data.frame(i = c(2,2), j = c(3,4), v = c(1,1))
const3_1 <- data.frame(i = c(3,3), j = c(5,6), v = c(1,1))

# constraints for warehouses
const4_1 <- data.frame(i = c(4,4,4,4,4,4,4), j = c(1,3,5,7,8,9,10), v = c(-1,-1,-1,1,1,1,1))
const5_1 <- data.frame(i = c(5,5,5,5,5,5,5), j = c(2,4,6,11,12,13,14), v = c(-1,-1,-1,1,1,1,1))

# constraints for customers
const6_1 <- data.frame(i = c(6,6), j = c(7,11), v = c(rep(1,1)))
const7_1 <- data.frame(i = c(7,7), j = c(8,12), v = c(rep(1,1)))
const8_1 <- data.frame(i = c(8,8), j = c(9,13), v = c(rep(1,1)))
const9_1 <- data.frame(i = c(9,9), j = c(10,14), v = c(rep(1,1)))

var_names1 <- paste0('x', edge_list1$i, edge_list1$j) # Name of edge

# Combine all constraints to a dataframe
const_1 <- rbind(const1_1,const2_1,const3_1,const4_1,const5_1,const6_1,const7_1,const8_1,const9_1)

# Create the sparse matrix of direction value of constraints
```

```r
const_sparse_1 <- simple_triplet_matrix(i = const_1$i, j = const_1$j, v = const_1$v)
A1 <- as.matrix(const_sparse_1)

colnames(A1) <- var_names1

# Right-hand side setup
b1 <- c(440, 650, 410, 0, 0, 300, 300, 250, 400)

# Constraints direction
dir1 <- c(rep("<=",3),rep("==",2),rep(">=",4))

# Solve
model1 <- lp(direction = "min",
             objective.in = c1,
             const.mat = A1,
             const.dir = dir1,
             const.rhs = b1,
             all.int = TRUE)

for (i1 in 1:length(model1$solution)) {
  print(paste("The delivery cost from node", edge_list1$i[i1],
              "to node", edge_list1$j[i1],
              "is", model1$solution[i1]))
}
```

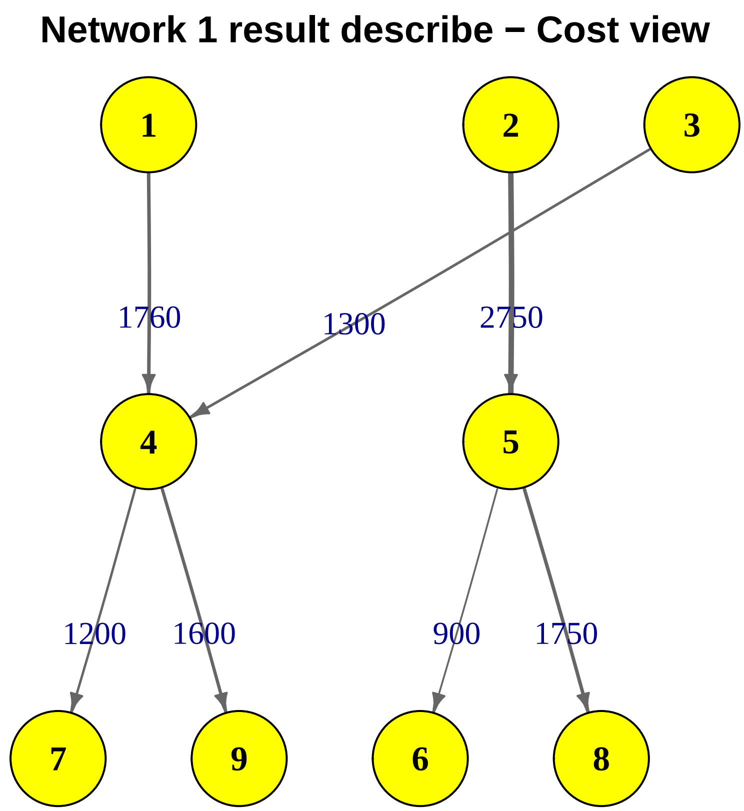
```
## [1] "The delivery cost from node 1 to node 4 is 440"
## [1] "The delivery cost from node 1 to node 5 is 0"
## [1] "The delivery cost from node 2 to node 4 is 0"
## [1] "The delivery cost from node 2 to node 5 is 550"
## [1] "The delivery cost from node 3 to node 4 is 260"
## [1] "The delivery cost from node 3 to node 5 is 0"
## [1] "The delivery cost from node 4 to node 6 is 0"
## [1] "The delivery cost from node 4 to node 7 is 300"
## [1] "The delivery cost from node 4 to node 8 is 0"
## [1] "The delivery cost from node 4 to node 9 is 400"
## [1] "The delivery cost from node 5 to node 6 is 300"
## [1] "The delivery cost from node 5 to node 7 is 0"
## [1] "The delivery cost from node 5 to node 8 is 250"
## [1] "The delivery cost from node 5 to node 9 is 0"
```

```r
print(paste("The optimal network cost is", model1$objval))
```

```
## [1] "The optimal network cost is 11260"
```

```r
# Add results column into edgelist to draw graph
edge_list1$shipping1 <- model1$solution
edge_list1$total_cost1 <- edge_list1$cost * edge_list1$shipping1

# Keep only positive edge
edge_list1 <- edge_list1[edge_list1$shipping1 > 0, ]

# Create graph
g1_new <- graph_from_data_frame(d = edge_list1[, c("i", "j")], directed = TRUE)
E(g1_new)$weight <- edge_list1$total_cost1
```

```r
# Adjust display margin
par(mar = c(1, 1, 1, 1))

plot(g1_new,
     layout = layout_as_tree(g1_new),
     vertex.color=c("Yellow"),
     vertex.size = 30,
     vertex.label.font=2,
     vertex.label.color = 'black',
     vertex.label.cex=1.1,
     edge.color="gray40",
     edge.width= E(g1_new)$weight/1000,
     edge.label = E(g1_new)$weight,
     edge.label.cex = 1,
     edge.label.font = 1,
     edge.curved = 0.01,
     edge.arrow.size=.5,
     main = 'Network 1 result describe - Cost view'
)
```

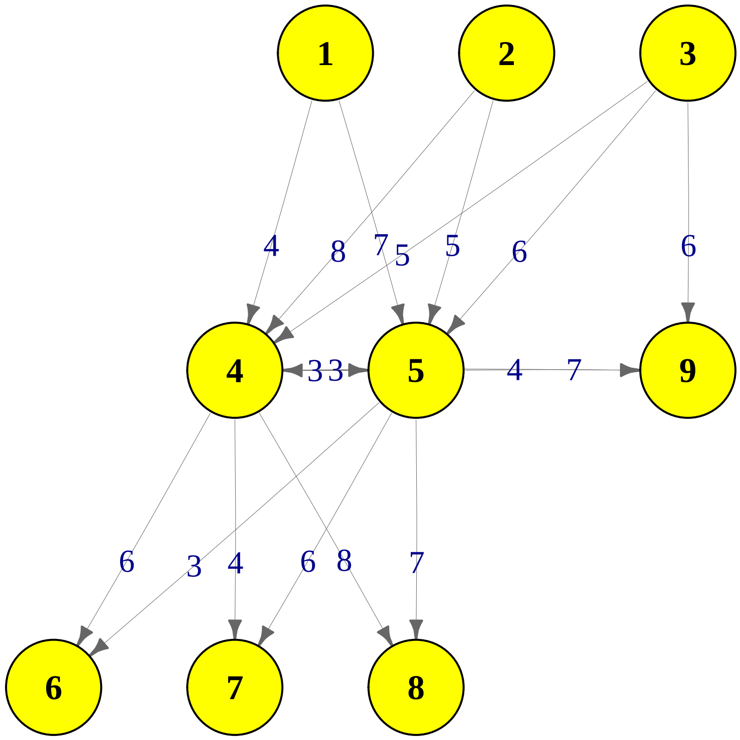


Question d

```r
# Plants to warehouses cost
plant1_cost2 <- plant1_cost1
plant2_cost2 <- plant2_cost1
```

```r
# Plant 3 to warehouses and customers cost
plant3_cost2 <- data.frame(i = rep(3,3), j = c(4,5,9), cost = c(5,6,6))

# Warehouses to warehouse and customers cost
wh1_cost2 <- data.frame(i = rep(4,5), j = c(5,6,7,8,9), cost = c(3,6,4,8,4))
wh2_cost2 <- data.frame(i = rep(5,5), j = c(4,6,7,8,9), cost = c(3,3,6,7,7))

# Edges list
edge_list2 <- rbind(plant1_cost2, plant2_cost2, plant3_cost2, wh1_cost2, wh2_cost2)
edge_list2 <- cbind(edge = 1:nrow(edge_list2), edge_list2)

# Create graph
g2 <- graph_from_data_frame(d = edge_list2[, c("i", "j", "cost")], directed = TRUE)

# Adjust display margin
par(mar = c(1, 1, 1, 1))

# Plot the graph
plot(g2,
     layout = layout_as_tree(g2),
     vertex.color=c("Yellow"),
     vertex.size = 30,
     vertex.label.font=2,
     vertex.label.color = 'black',
     vertex.label.cex=1.1,
     edge.color="gray40",
     edge.width= E(g2)$cost/100,
     edge.label = E(g2)$cost,
     edge.label.cex = 1,
     edge.label.font = 1,
     edge.curved = 0.01,
     edge.arrow.size=.5,
     main = 'Network 2 visualization'
)
```

Network 2 visualization



Question e

Sets:
V: Set of all nodes in the network, $V = \{1, 2, ..., 9\}$
I: Set of origin nodes, $I = \{1, 2, 3\}$
T: Set of transshipment nodes, $T = \{4, 5\}$
J: Set of destination nodes, $J = \{6, 7, 8, 9\}$
A: Set of directed edges between nodes, $(i, k), (i, j), (k, k'), (k, j) \in A, \forall i, j, k, k', l \in V$

Indices:
i: origin nodes, $i \in I$
k (and k'): transshipment nodes, $k, k' \in T$
j: destination nodes, $j \in J$

Parameters:
c_{ik}: the delivery cost per unit from node i to node k
c_{ij}: the delivery cost per unit from node i to node j
$c_{kk'}$: the delivery cost per unit from node k to node k'
c_{kj}: the delivery cost per unit from node k to node j
s_i: the supply capacity of node i
d_j: the demand of node j

Decision variables:
x_{ik}: the delivery quantity from node i to node k
x_{ij}: the delivery quantity from node i to node j
$x_{kk'}$: the delivery quantity from node k to node k'

x_{kj}: the delivery quantity from node k to node j

Objective function:
Minimizing the total delivery cost

$$Min \sum_{(i,k)\in A} c_{ik}x_{ik} + \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{(k,k')\in A} c_{kk'}x_{kk'} + \sum_{(k,j)\in A} c_{kj}x_{kj}$$

Constraints:
For each origin node i, the total outbound quantity must not exceed the capacity:

$$\sum_{(i,k)\in A} x_{ik} + \sum_{(i,j)\in A} x_{ij} \leq s_i, \forall i \in I$$

For each destination node j, the total inbound quantity must fulfill the demand:

$$\sum_{(k,j)\in A} x_{kj} + \sum_{(i,j)\in A} x_{ij} \geq d_j, \forall j \in J$$

For each transshipment node k and k', the inbound and outbound quantity must be equal:

$$\sum_{(i,k)\in A} x_{ik} - \sum_{(k,k')\in A} x_{kk'} - \sum_{(k,j)\in A} x_{kj} = 0, \forall k, k' \in T$$

For each edge (i,k), (i,j), (k,k'), (k,j), the delivery quantity must be non-negative:

$$x_{ik} \geq 0, x_{ij} \geq 0, x_{kk'} \geq 0, x_{kj} \geq 0, \forall (i,k), (i,j), (k,k'), (k,j) \in A$$

Question f

```
# delivery cost on each edge
c2 <- edge_list2$cost

# constraints for plants
const1_2 <- data.frame(i = c(1,1), j = c(1,2), v = c(1,1))
const2_2 <- data.frame(i = c(2,2), j = c(3,4), v = c(1,1))
const3_2 <- data.frame(i = c(3,3,3), j = c(5,6,7), v = c(1,1,1))

# constraints for warehouses
const4_2 <- data.frame(i = c(rep(4,8)), j = c(1,3,5,8:12), v = c(rep(-1,3), rep(1,5)))
const5_2 <- data.frame(i = c(rep(5,8)), j = c(2,4,6,13:17), v = c(rep(-1,3), rep(1,5)))

# constraints for customers
const6_2 <- data.frame(i = c(rep(6,2)), j = c(9,14), v = c(rep(1,2)))
const7_2 <- data.frame(i = c(rep(7,2)), j = c(10,15), v = c(rep(1,2)))
const8_2 <- data.frame(i = c(rep(8,2)), j = c(11,16), v = c(rep(1,2)))
const9_2 <- data.frame(i = c(rep(9,3)), j = c(7,12,17), v = c(rep(1,3)))

var_names2 <- paste0('x', edge_list2$i, edge_list2$j) # Name of edge

# Combine all constraints to a dataframe
const_2 <- rbind(const1_2, const2_2, const3_2, const4_2, const5_2, const6_2,
                 const7_2, const8_2, const9_2)

# Create the sparse matrix of direction value of constraints
```

```r
const_sparse_2 <- simple_triplet_matrix(i = const_2$i, j = const_2$j, v = const_2$v)
A2 <- as.matrix(const_sparse_2)

colnames(A2) <- var_names2

# Right-hand side setup
b2 <- b1

# Constraints direction
dir2 <- dir1

# Solve
model2 <- lp(direction = "min",
             objective.in = c2,
             const.mat = A2,
             const.dir = dir2,
             const.rhs = b2,
             all.int = TRUE)

# Print result
for (i2 in 1:length(model2$solution)) {
  print(paste("The delivery cost from node", edge_list2$i[i2],
              "to node", edge_list2$j[i2],
              "is", model2$solution[i2]))
}
```

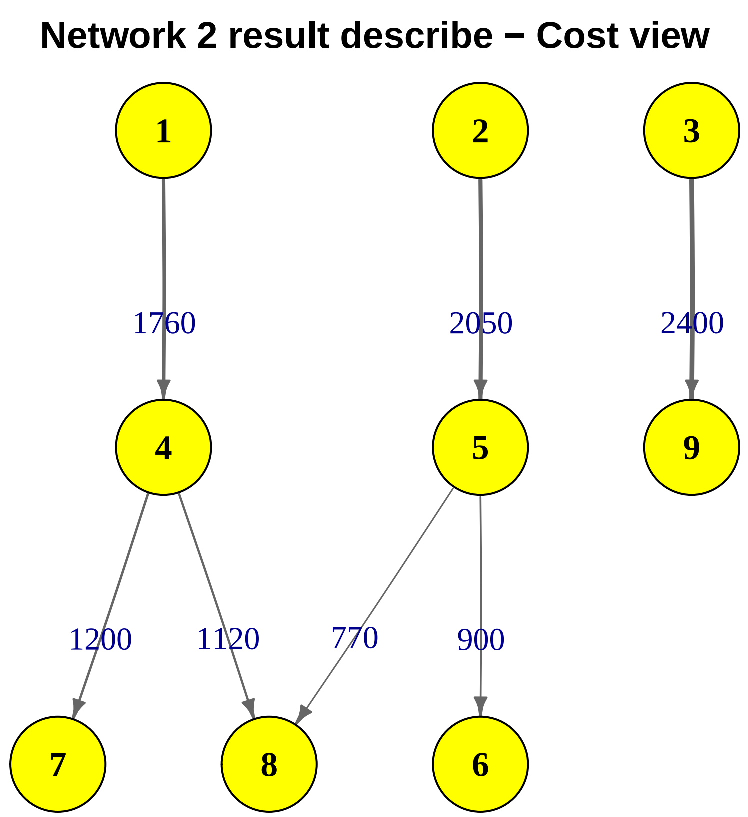
```
## [1] "The delivery cost from node 1 to node 4 is 440"
## [1] "The delivery cost from node 1 to node 5 is 0"
## [1] "The delivery cost from node 2 to node 4 is 0"
## [1] "The delivery cost from node 2 to node 5 is 410"
## [1] "The delivery cost from node 3 to node 4 is 0"
## [1] "The delivery cost from node 3 to node 5 is 0"
## [1] "The delivery cost from node 3 to node 9 is 400"
## [1] "The delivery cost from node 4 to node 5 is 0"
## [1] "The delivery cost from node 4 to node 6 is 0"
## [1] "The delivery cost from node 4 to node 7 is 300"
## [1] "The delivery cost from node 4 to node 8 is 140"
## [1] "The delivery cost from node 4 to node 9 is 0"
## [1] "The delivery cost from node 5 to node 4 is 0"
## [1] "The delivery cost from node 5 to node 6 is 300"
## [1] "The delivery cost from node 5 to node 7 is 0"
## [1] "The delivery cost from node 5 to node 8 is 110"
## [1] "The delivery cost from node 5 to node 9 is 0"
```

```r
print(paste("The optimal network cost is", model2$objval))
```

```
## [1] "The optimal network cost is 10200"
```

```r
# Add results column into edgelist to draw graph
edge_list2$shipping2 <- model2$solution
edge_list2$total_cost2 <- edge_list2$cost * edge_list2$shipping2

# Keep only positive edge
edge_list2 <- edge_list2[edge_list2$shipping2 > 0, ]
```

```r
# Graph
g2_new <- graph_from_data_frame(d = edge_list2[, c("i", "j")], directed = TRUE)
E(g2_new)$weight <- edge_list2$total_cost2

# Adjust display margin
par(mar = c(1, 1, 1, 1))

# Plot the graph
plot(g2_new,
     layout = layout_as_tree(g2_new),
     vertex.color=c("Yellow"),
     vertex.size = 30,
     vertex.label.font=2,
     vertex.label.color = 'black',
     vertex.label.cex=1.1,
     edge.color="gray40",
     edge.width= E(g2_new)$weight/1000,
     edge.label = E(g2_new)$weight,
     edge.label.cex = 1,
     edge.label.font = 1,
     edge.curved = 0.01,
     edge.arrow.size=.5,
     main = 'Network 2 result describe - Cost view'
)
```



Problem 3

Question a

```r
# Clear environment
rm(list = ls())

# Import the data
c1 <- read.csv("D:\\WU\\SCA1\\Exercise 2\\large-scale\\large-scale\\obj_function_coefs_c.csv",
               row.names = 1) %>% unlist()

# Print the results
print(paste("There are", length(c1), "decision variables" ))
```

```
## [1] "There are 5000 decision variables"
```

Question b

```r
# Import library
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::%--%()      masks igraph::%--%()
## x dplyr::as_data_frame() masks tibble::as_data_frame(), igraph::as_data_frame()
## x purrr::compose()       masks igraph::compose()
## x tidyr::crossing()      masks igraph::crossing()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x purrr::simplify()      masks igraph::simplify()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
# Import the data
A1 <- read.csv("D:\\WU\\SCA1\\Exercise 2\\large-scale\\large-scale\\constraint_matrix_A.csv",
               row.names = 1) %>% as.matrix()

b1 <- read.csv("D:\\WU\\SCA1\\Exercise 2\\large-scale\\large-scale\\constraint_vector_b.csv",
               row.names = 1) %>% unlist()

# Print the results
print(paste("There are", length(b1), "constraints" ))
```

```
## [1] "There are 5750 constraints"
```

Question c

```r
# Import library
library(ROI)
```

```
## ROI: R Optimization Infrastructure
```

```
## Registered solver plugins: nlminb, glpk, nloptr.bobyqa, nloptr.crs2lm, nloptr.direct, nloptr.directL
```

```
## Default solver: auto.
```

```r
# Create names for variable
names <- c()
for (i in 1:length(c1)) {
  names[i] <- paste0("x", i)
}

# Set up the solver
lp  <- OP(objective = L_objective(c1, names=names),
          constraints = L_constraint(L = A1, dir = c(rep("<=",5750)), rhs = b1),
          types = rep("C",length(c1)),
          bounds = V_bound(li = 1:5000, lb = c(rep(0,5000))),
          maximum = TRUE)

# Check the solver
lp
```

```
## ROI Optimization Problem:
##
## Maximize a linear objective function of length 5000 with
## - 5000 continuous objective variables,
##
## subject to
## - 5750 constraints of type linear.
## - 0 lower and 0 upper non-standard variable bounds.
```

```r
# Time before solve
t1 <- Sys.time()

# Solve the problem with glpk
solution_glpk <- ROI_solve(lp, solver = "glpk")

# Time after solve
t2 <- Sys.time()

# Time different
print(paste("Solving time:", t2 - t1))
```

```
## [1] "Solving time: 16.4126698970795"
```

```r
# Objective value
print(paste("The optimal objective value is", solution_glpk$objval))
```

```
## [1] "The optimal objective value is 2066.45842404805"
```

Question d

```r
# Print the solution
solution_gplk_df <- data.frame(solution_glpk$solution) %>%
  rename(solution_value = solution_glpk.solution) %>%
  arrange(desc(solution_value))

head(solution_gplk_df)
```

```
##       solution_value
## x465      0.09873975
## x3976     0.08595195
## x1644     0.07341138
## x1866     0.06978700
## x1702     0.06944741
## x2258     0.06892376
```

```r
# Calculate the share of decision variables that is non-zero
print(paste("The share of non-zero decision variables is",
            sum(solution_gplk_df$solution_value != 0) / 5000*100,"%"))
```

```
## [1] "The share of non-zero decision variables is 8.88 %"
```

Question e

```r
# Time before solve
t3 <- Sys.time()

# Solve the with solver
solution_symphony <- ROI_solve(lp, solver = "symphony")

# Time before solve
t4 <- Sys.time()

# Time different
print(paste("Solving time:", t4 - t3))
```

```
## [1] "Solving time: 19.0607590675354"
```

```r
# Objective value
print(paste("The optimal objective value is", solution_symphony$objval))
```

```
## [1] "The optimal objective value is 2066.45842404808"
```

```r
# Print the solution
solution_symphony_df <- data.frame(solution_symphony$solution) %>%
  rename(solution_value = solution_symphony.solution) %>%
  arrange(desc(solution_value))

head(solution_symphony_df)
```

```
##       solution_value
## x465      0.09873975
## x3976     0.08595195
## x1644     0.07341138
## x1866     0.06978700
## x1702     0.06944741
## x2258     0.06892376
```

```r
# Calculate the share of decision variables that is non-zero
print(paste("The share of non-zero decision variables is",
            sum(solution_symphony_df$solution_value != 0) / 5000*100,"%"))
```

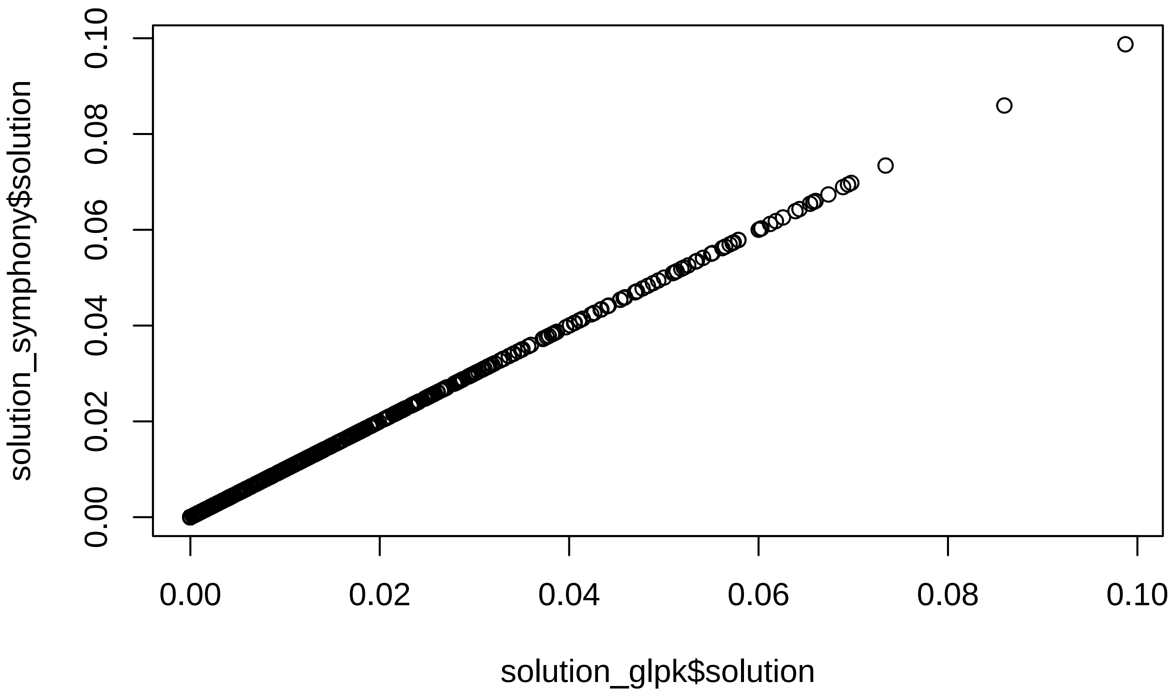
```
## [1] "The share of non-zero decision variables is 8.88 %"
```

```r
# Compare the 2 results by correlation coefficient
print(paste("Correlation coefficient between 2 solvers' optimal decision variables is:",
```

```
            cor(solution_glpk$solution, solution_symphony$solution)))
```

```
## [1] "Correlation coefficient between 2 solvers' optimal decision variables is: 1"
```

```
# Compare the 2 results by visual
plot(solution_glpk$solution, solution_symphony$solution)
```



Problem 4

Question a

```
# Clear environment
rm(list = ls())

# Set up sales function
sales_function <- function (x) {
  s = -3*x[1]^2 - 8*x[2]^2 - 6*x[1]*x[2] + 16*x[1] + 34*x[2]
  return(s)
}

# Parameter
x1 <- c(0,1)
x2 <- c(2,1)
x3 <- c(1,0)
x4 <- c(1,2)
```

```r
# Print results
print(paste("Sales when 0$ is spent on radio ads and $1000 is spent on direct-mail is",
            sales_function(x1)*1000, "$"))
```

```
## [1] "Sales when 0$ is spent on radio ads and $1000 is spent on direct-mail is 26000 $"
```

```r
print(paste("Sales when 2000$ is spent on radio ads and $1000 is spent on direct-mail is",
            sales_function(x2)*1000, "$"))
```

```
## [1] "Sales when 2000$ is spent on radio ads and $1000 is spent on direct-mail is 34000 $"
```

```r
print(paste("Sales when 1000$ is spent on radio ads and $0 is spent on direct-mail is",
            sales_function(x3)*1000, "$"))
```

```
## [1] "Sales when 1000$ is spent on radio ads and $0 is spent on direct-mail is 13000 $"
```

```r
print(paste("Sales when 1000$ is spent on radio ads and $2000 is spent on direct-mail is",
            sales_function(x4)*1000, "$"))
```

```
## [1] "Sales when 1000$ is spent on radio ads and $2000 is spent on direct-mail is 37000 $"
```

Observation: When the same amount of money is allocated to direct-mail and radio advertising, the sales generated from radio advertising is much higher than the the case of direct-mail.

Question b

Decision variables:
$x \in \mathbb{R}^2$, $\mathbf{x}^\top = [x_1, x_2]$: amount of money spending on each advertising method

Objective function coefficients:
$Q \in \mathbb{R}^{2\times 2}$, $Q = \begin{bmatrix} -3 & -3 \\ -3 & -8 \end{bmatrix}$: quadratic term coefficient matrix
$c \in \mathbb{R}^2$, $\mathbf{c}^\top = \begin{bmatrix} 16 & 34 \end{bmatrix}$: linear term coefficient vector

Constraint coefficients:
$A \in \mathbb{R}^{1\times 2}$, $A = \begin{bmatrix} 1 & 1 \end{bmatrix}$
$b \in \mathbb{R}^1$, $b = [3]$

Objective function:
Maximizing the sales generated from advertising

$$\max f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top Q\mathbf{x} + \mathbf{c}^\top \mathbf{x}$$

Constraints:
The total spending on advertising must not exceed the budget

$$Ax \leq b$$

The spending for each advertising method must be non-negative

$$x \geq 0$$

Question c

Solving using quadratic function (the result of this function is not recognized for further tasks)

```r
# Import library
library(ROI)

# Define the quadratic term (Q) and linear term (c)
Q <- matrix(c(-3, -3, -3, -8), nrow = 2, byrow = TRUE)
C <- c(16, 34)

# Define the constraint matrix (A) and constraint values (b)
A <- matrix(c(1, 1), nrow = 1, byrow = TRUE)
b <- c(3)
dircon <- c("<=")

# Setup solver and solve the problem
qp <- OP(maximum = TRUE,
         objective = Q_objective(Q = Q, L = C, names = c("x1","x2")),
         constraints = L_constraint(L = A, dir = dircon, rhs = b),
         types = c("C","C"))
qp
```

```
## ROI Optimization Problem:
##
## Maximize a quadratic objective function of length 2 with
## - 2 continuous objective variables,
##
## subject to
## - 1 constraint of type linear.
## - 0 lower and 0 upper non-standard variable bounds.
```

```r
solution_qp <- ROI_solve(qp, solver = 'nloptr.cobyla',
                         control = list(start = c(0,0)))

print(paste("Optimal decision variable is", solution_qp$solution))
```

```
## [1] "Optimal decision variable is 1.0842021724855e-19"
## [2] "Optimal decision variable is 3"
```

```r
print(paste("Optimial objective value is", solution_qp$objval))
```

```
## [1] "Optimial objective value is 66"
```

Solving using nonlinear programming function (the result of this function is recognized for further tasks)

```r
# Objective function
sales_function
```

```
## function (x) {
##   s = -3*x[1]^2 - 8*x[2]^2 - 6*x[1]*x[2] + 16*x[1] + 34*x[2]
##   return(s)
## }
## <bytecode: 0x0000019d8b37b3c8>
```

```r
# Constraint function
A <- function(x) {
  return(x[1] + x[2])
}

# Constraint direction
```

```r
condir <- c("<=")

# Constraint RHS
b <- 3

# Solve the problem

nlp <- OP(F_objective(sales_function, n = 2),
          constraints = F_constraint(F = A, dir = condir, rhs = b),
          maximum = TRUE)
nlp
```

```
## ROI Optimization Problem:
##
## Maximize a nonlinear objective function of length 2 with
## - 2 continuous objective variables,
##
## subject to
## - 1 constraint of type nonlinear.
## - 0 lower and 0 upper non-standard variable bounds.
```

```r
solution_nlp <- ROI_solve(nlp,
                          solver = "nloptr.cobyla",
                          control = list(start = c(0,0)))

# Print result
print(paste("Optmial decision variable is",
            solution_nlp$solution))
```

```
## [1] "Optmial decision variable is 0.866498589525322"
## [2] "Optmial decision variable is 1.80017914653327"
```

```r
print(paste("Optimial objective value is",
            solution_nlp$objval))
```

```
## [1] "Optimial objective value is 37.5333331724983"
```

```r
print(paste("Budget share for radio advertising is",
            solution_nlp$solution[1]/b*100, "%"))
```

```
## [1] "Budget share for radio advertising is 28.8832863175107 %"
```

```r
print(paste("Budget share for direct-mail is",
            solution_nlp$solution[2]/b*100, "%"))
```

```
## [1] "Budget share for direct-mail is 60.005971551109 %"
```

```r
print(paste("The remaining budget is",
            (1-solution_nlp$solution[1]/b-solution_nlp$solution[2]/b)*100,
            "%"))
```

```
## [1] "The remaining budget is 11.1107421313802 %"
```

```r
print(paste("The expected generated sales due to the advertising campaign is",
            solution_nlp$objval))
```

```
## [1] "The expected generated sales due to the advertising campaign is 37.5333331724983"
```

The budget is not fully utilize in the optimal solution because in the sales generated function we have the negative coefficient with quadratic terms $-3x_1^2 - 8x_2^2$. Therefore, when we utilize all the budget, we will get a enormous negative value by $-3x_1^2 - 8x_2^2$ which the linear term $16x_1 + 34x_2$ cannot cover and the final result will be negative.

Problem 5

Question a

```
# Clear environment
rm(list=ls())

# Read the data
data <- read.csv("D:\\WU\\SCA1\\Exercise 2\\delhi_bus_network_hw2.csv")

# Print the result
print(paste("There are", length(unique(data$source)), "nodes"))
```

```
## [1] "There are 1103 nodes"
```

```
print(paste("There are", length(data$weight), "edges"))
```

```
## [1] "There are 2582 edges"
```

Question b

```
# Create edge_list from dataframe
edge_list <- as.matrix(data)

# Create graph
g <- graph_from_edgelist(el = edge_list[,1:2], directed = TRUE)
E(g)$weight <- edge_list[,3]

# Compute the shortest path when going from stop Sukhram.Nagar to Naroda.Terminus
path <- shortest_paths(g,
                       from = "Sukhram.Nagar",
                       to = "Naroda.Terminus",
                       output = 'both')
print(path$vpath)
```

```
## [[1]]
## + 8/1103 vertices, named, from d57078c:
## [1] Sukhram.Nagar      Arbuda.Mills       Rakhial            Sarangpur
## [5] S.T.(Gita.Mandir)  Parikshitlal.Nagar S.T.Workshop       Naroda.Terminus
```

```
# Print number of bus change
print(paste("Number of bus changes:", length(path$vpath[[1]]) - 1))
```

```
## [1] "Number of bus changes: 7"
```

```
# Compute the traveling time
time <- distances(g,
                  v= "Sukhram.Nagar",
                  to = "Naroda.Terminus",
                  mode = "out",
```

```r
                  weights = E(g)$weight)

print(paste("The traveling time is:", time))
```

```
## [1] "The traveling time is: 8"
```

Question c

```r
# Calculate all the paths
all_path <- distances(graph = g, mode = "out", weights = E(g)$weigth)

# Calculate maximum time
max_time <- max(all_path, na.rm = TRUE)
print(paste("Maximum time if travelling between any pair of bus stop is",
            max_time))
```

```
## [1] "Maximum time if travelling between any pair of bus stop is 33"
```

```r
# Figure out the max path
max_path <- which(all_path == max_time, arr.ind = TRUE)
from_node <- V(g)[max_path[1, 1]]$name
to_node <- V(g)[max_path[1, 2]]$name
print(paste("The longest shortest path is from", from_node,
            "to", to_node))
```

```
## [1] "The longest shortest path is from Sahayog.Primary.School to Gujarat.Electricity.Board"
```

```r
print(paste("Number of bus changes:",
            length(shortest_paths(g,
                                  from = from_node,
                                  to = to_node, output = 'both')$vpath[[1]]) - 1))
```

```
## [1] "Number of bus changes: 25"
```