

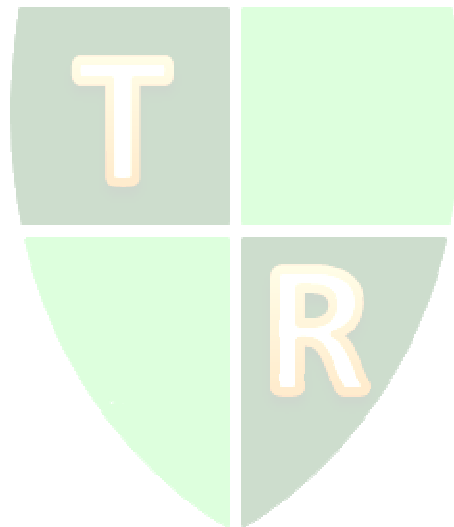
A case for network optimization with R

Jerzy Letkowski
Western New England University

ABSTRACT

Network modeling and optimization is one of the most popular topics taught in academic courses for Management Science / Operations Research (MS / OR). Traditionally, this topic has been and still is taught using spreadsheet technologies and/or specialized software packages. The explosion of application development, using such languages like R and Python, with respect to data analytics, shows lots of opportunities and offers interesting solutions for these languages to be incorporated into teaching MS / OR. This paper is about a simple case for network modeling with R. In particular, it shows an example for the transportation problem along with R-based solutions obtained through the R-implemented *Simplex* method and graphics.

Keywords: R, optimization, graph, network, transportation, linear programming, visualization.



Copyright statement: Authors retain the copyright to the manuscripts published in AABRI journals. Please see the AABRI Copyright Policy at <http://www.aabri.com/copyright.html>

NETWORK MODELING

Visual modeling methods are both easy to develop and easy to understand. Network models fall into this realm. The class of decision problems, explored in this paper, can be effectively treated by network models.

In a decision model, a network is defined as a graph of vertices (nodes) and arcs (edges) and their characteristics (properties). Arcs connect some of the vertices. A graph, G , can be formally defined as a pair of two sets: V (a set of vertices) and A (a set of arcs).

$$G = \{V, A\} \quad (1)$$

For example, if we use integers to label the vertices, then V can be identified by a sequence of integers, $V = \{1, 2, \dots, n\}$. The arcs, A , would then be defined as pairs of such vertices, $A = \{(i, j), i \in V, j \in V\}$. The first graph (G_1) shown in *Figure 1* is defined as:

$$G_1 = \{V_1, A_1\},$$

$$V_1 = \{1, 2, 3, 4\}, A_1 = \{(1,2), (1,3), (2,3), (3,4), (4,3)\}. \quad (2)$$

The second graph (G_2) has the following definition:

$$G_2 = \{V_2, A_2\},$$

$$V_2 = \{1, 2, 3, 4\}, A_2 = \{(1,2), (1,3), (2,1), (2,3), (3,1), (3,2), (3,4), (4,3)\} \quad (3)$$

Depending on how the vertices are connected, a graph can be directed (G_1) or undirected (G_2). In a directed graph, pairs of vertices are connected with arcs depicted as arrows.

Graphs are an interesting way to visualize some data structures such as connections between cities, ports, hubs, computers, pump stations, etc. One of the most popular business applications, using graphs are CPM and PERT methods [1 p. 579]. They are a more sophisticated way to capture the structure and schedule of projects.

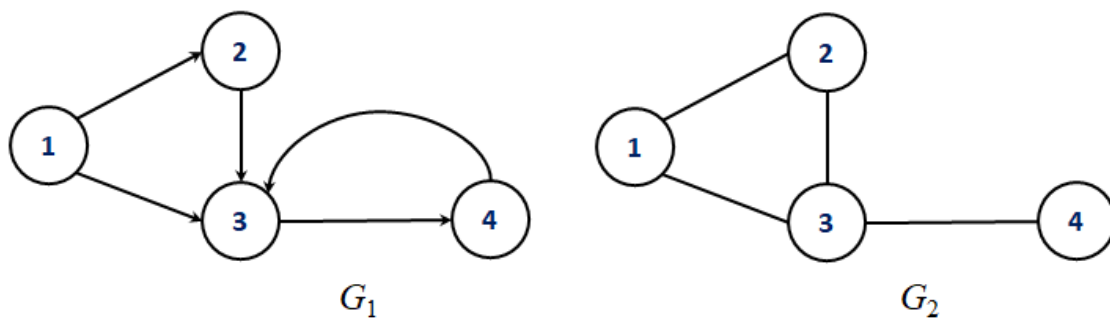


Figure 1. Examples of graphs.

Using graphs in solving decision problems, requires addition data that needs to be added to graph vertices and/or arcs. For example, if nodes represent cities, arcs may be interpreted as connections of particular lengths. A decision problem can then be formulated for finding the shortest or the least expensive connection (path) between vertices of the network. If the vertices represent pump stations, we may be interested in finding the maximum flow through the network of the pumps, given we know the flow capacities of the arcs. Thus, in order to define a decision problem, using a network model, we also need to specify some properties of the graph components as well as a goal we want to be accomplished.

Figure 2 extends the models shown in *Figure 1* by adding arc properties, for example, distances.

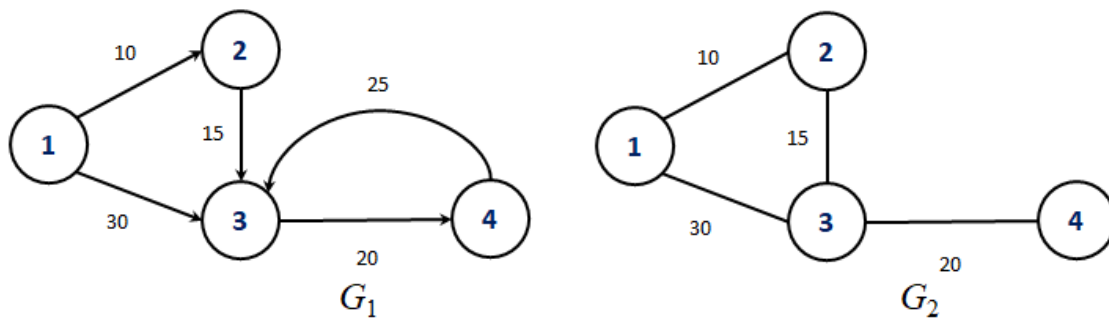


Figure 2. Examples of graphs with arc properties.

Many decision problems can be solved, using algorithms that are based on graph objects and their properties. An interesting collection of the problems are shown in [1], [3], [5], [6]—just to name a few. In this paper, an **R** implementation of the solution algorithm is explored for the *Transportation* problem.

LEARNING OBJECTIVES

By analyzing and completing the case in **R**, students are expected to have a better understanding of the subject matter and they will strengthen their skills of network application programming and visualization in **R**. All the problems (cases), presented in this paper, can be solved numerically by developing appropriate input in form of lists, matrices, and linear constraints. Such an approach can be effective technically but it is usually considered to less comprehensible or even more boring, when it comes to learning.

This paper also explores the aspect of the graphical visualization that, in most situations, is expected to contribute to better understanding of both the problem specifications, and solution procedures, as well as their results. The visual approach can be applied manually (using hand drawing) and/or programmatically (using **R** graphics). It is suggested that the students do both the manual and **R**-programmatic graphics. They will then get a better insight into the graphics world of **R**.

CASE - TRANSPORTATION MODEL

Consider a simple problem of moving (shipping) some goods from m sources directly to n destinations. The following example shows a 3 by 4 network (graph) model ($m = 3, n = 4$). It was drawn, using Microsoft Office - PowerPoint.

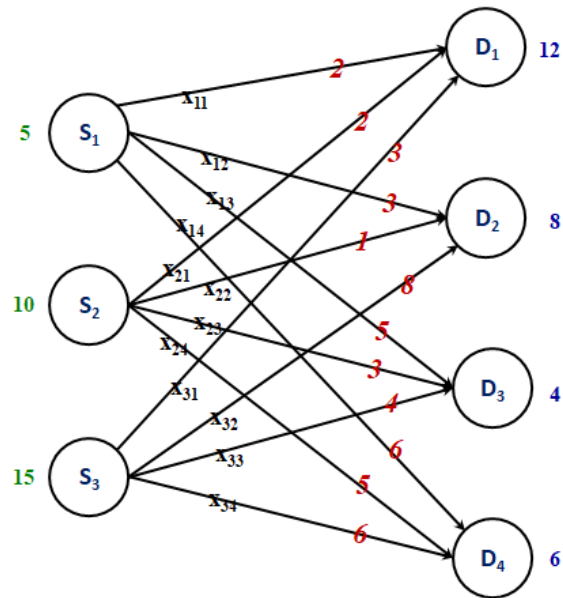


Figure 3. A graphical model for the transportation problem.

The sources (suppliers) are labeled as S_1, S_2, \dots, S_m and the destinations (points of demands)— D_1, D_2, \dots, D_n . Thus the vertices of the graph model include both the source nodes and destination nodes as disjoint collections S and D ($V = \{S, D\}$ and $S \cap D = \emptyset$). Assuming that we know the quantities of the goods available at each source, s_1, s_2, \dots, s_m (shown in green on the above diagram) and quantities of the goods demanded at each destination, d_1, d_2, \dots, d_n (shown in blue). We also know costs to ship one unit of the goods for each connection between the sources and destinations, $\{c(S_i, D_j), S_i \in S, D_j \in D\}$ (shown in red).

We also assume that there exist a feasible solution to this problem. There is a balance between the total supply and the total demand, $\sum(s_1, s_2, \dots, s_m) = \sum(d_1, d_2, \dots, d_n)$ (here $5 + 10 + 15 = 12 + 8 + 4 + 6 = 30$). By the way, this assumption is not restrictive. If there were an unbalanced situation, the problem can still be solved by adding a so called dummy supplier or receiver [1, p. 474].

Our goal is to find shipment quantities, $\{x_{ij} : S_i \in S, D_j \in D\}$ such that the total shipment cost will be minimized. These values are interpreted as outflows from the source vertices and inflows into the destination vertices, respectively.

A network model, for this decision situation uses graph $G = \{V, A\}$, where $V = \{S, D\}$ and $A = (S \times D)$, which is a Cartesian product of the set of sources (suppliers), S , and the set of destinations (points of demand), D . Additionally, the model includes supply and demand

quantities attached to the sources and destinations, respectively. Finally the arcs are mapped to a matrix of the shipment cost, $A \rightarrow \{c(S_i, D_j), S_i \in S, D_j \in D\}$ and to the decision variables, $A \rightarrow \{x(S_i, D_j), S_i \in S, D_j \in D\}$.

An important rule to be observed by the graph model is that, for every vertex, the total inflow is the same as the total outflow. What goes into a vertex, comes out from the vertex (vertex outflow is equal to vertex inflow). For the supplier vertices (S), the inflow is fixed (supply quantities). For the receiver vertices (D), the outflow is fixed (demand quantities). Applying this rule to our example, we get the following constraints:

$$\begin{aligned}
 S_1: x_{11} + x_{12} + x_{13} + x_{14} &= s_1 = 5 \\
 S_2: x_{21} + x_{22} + x_{23} + x_{24} &= s_2 = 10 \\
 S_3: x_{31} + x_{32} + x_{33} + x_{34} &= s_3 = 15 \\
 D_1: x_{11} + x_{21} + x_{31} &= d_1 = 12 \\
 D_2: x_{21} + x_{22} + x_{23} &= d_2 = 8 \\
 D_3: x_{31} + x_{32} + x_{33} &= d_3 = 4 \\
 D_4: x_{31} + x_{32} + x_{33} &= d_4 = 6
 \end{aligned} \tag{4}$$

The first source (S_1) sends x_{11} units to destination D_1 , x_{12} units to D_2 , x_{13} units to D_3 , and x_{14} units to D_4 . The sum of these quantities must be equal to what this source has on hand ($s_1 = 5$). In a similar way the other suppliers (S_2 and S_3) are handled.

In the set of the destination constraints (5), the first destination (D_1) receives x_{11} units from source S_1 , x_{21} units from source S_2 , and x_{31} units from source S_3 . The sum these quantities must be equal to what this destination is demanding ($d_1 = 12$). The other destinations (D_2 , D_3 , and D_4) are managed in a similar way.

Obviously, all the shipment quantities are expected to be non-negative. Thus

$$x_{ij} \geq 0, \quad i = 1, 2, 3; \quad j = 1, 2, 3, 4. \tag{6}$$

Finally, since our goal is to come up with a solution that will minimize the total shipment cost, we can express this goal as a sum of the products of the unit costs and shipment quantities:

$$\text{Min } Z = \tag{7}$$

$$\begin{aligned}
 &c_{11}x_{11} + c_{12}x_{12} + c_{13}x_{13} + c_{14}x_{14} + \\
 &c_{21}x_{21} + c_{22}x_{22} + c_{23}x_{23} + c_{24}x_{24} + \\
 &c_{31}x_{31} + c_{32}x_{32} + c_{33}x_{33} + c_{34}x_{34} = \\
 &2x_{11} + 3x_{12} + 5x_{13} + 6x_{14} + \\
 &2x_{21} + 1x_{22} + 3x_{23} + 5x_{24} +
 \end{aligned}$$

$$3x_{31} + 8x_{32} + 4x_{33} + 6x_{34}$$

SIMPLEX METHOD

From Management Science, we learn that expressions (4) through (7) and (8) through (11) constitute a linear programming model [1, p. 473]. Thus our problem can be solved using the **Simplex** method.

Using the Σ notation for expressions (4) – (7), we can formulate a general transportation model as:

$$\text{Min } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (8)$$

$$\sum_{j=1}^n x_{ij} = s_i, \quad i = 1, 2, \dots, m \quad (9)$$

$$\sum_{i=1}^m x_{ij} = d_j, \quad j = 1, 2, \dots, n \quad (10)$$

$$x_{ij} \geq 0, \quad i = 1, 2, \dots, m; \quad j = 1, 2, \dots, n \quad (11)$$

In **R**, we can use the **linprog** library to implement the **Simplex** algorithm. Unfortunately, we will have to restructure our model to fit the data format for this library. First the decision variables and coefficients of the objective function, specified in the above transportation model as matrices, will have to be flattened—converted to vectors:

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix}$$

becomes vector x :

$$x = (x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{23}, x_{24}, x_{31}, x_{32}, x_{33}, x_{34}), \quad (12)$$

and

$$C = \begin{pmatrix} 2 & 3 & 5 & 6 \\ 2 & 1 & 3 & 5 \\ 3 & 8 & 4 & 6 \end{pmatrix}$$

becomes vector c :

$$c = (2, 3, 5, 6, 2, 1, 3, 5, 3, 8, 4, 6). \quad (13)$$

Thus the objective function becomes a [flat] sum of products:

$$\text{Min } Z = 2x_{11} + 3x_{12} + 5x_{13} + 6x_{14} + 2x_{21} + 1x_{22} + 3x_{23} + 5x_{24} + 3x_{31} + 8x_{32} + 4x_{33} + 6x_{34}$$

All the supply and demand constraints must be expressed in terms of all the decision variables. This can be done by rewriting the existing constraints with all other (irrelevant) variables multiplied by zero. The following examples show how to convert the first constraints defined above in (4) and (5).

Constraint $x_{11} + x_{12} + x_{13} + x_{14} = 5$ gets converted to:

$$1x_{11} + 1x_{12} + 1x_{13} + 1x_{14} + 0x_{21} + 0x_{22} + 0x_{23} + 0x_{24} + 0x_{31} + 0x_{32} + 0x_{33} + 0x_{34} = 5.$$

Constraint $x_{11} + x_{21} + x_{31} = d_1 = 12$ gets converted to:

$$1x_{11} + 0x_{12} + 0x_{13} + 0x_{14} + 1x_{21} + 0x_{22} + 0x_{23} + 0x_{24} + 1x_{31} + 0x_{32} + 0x_{33} + 0x_{34} = 12, \text{ etc.}$$

Wrapping it up, the left-hand-side coefficients of all the constraints become represented by the following matrix:

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (14)$$

The right-hand-side coefficients of all the constraints are expressed as the following vector (b):

$$b = \begin{pmatrix} 5 \\ 10 \\ 15 \\ 12 \\ 8 \\ 4 \\ 6 \end{pmatrix} \quad (15)$$

After installing the **linprog** library, using vectors c and b as well as matrix A , we can solve this problem in **R**.

R Code

```
# Install the library unless it is already set up.
install.packages("linprog")
# Use the library.
library(linprog)
# Define the coefficients of the objective function.
vc = c(2, 3, 5, 6, 2, 1, 3, 5, 3, 8, 4, 6)
vc
[1] 2 3 5 6 2 1 3 5 3 8 4 6
# The right-hand-side coefficients of the constraints.
vb = c(5, 10, 15, 12, 8, 4, 6)
vb
[1] 5 10 15 12 8 4 6
# The left-hand-side coefficients of the constraints. Start with vector a.
va = c(1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
```

```

1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1)
# Convert the vector to a matrix (row by row).
A = matrix(va, nrow=7, ncol=12, byrow=TRUE)
A
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]    1    1    1    1    0    0    0    0    0    0    0    0
[2,]    0    0    0    0    1    1    1    1    0    0    0    0
[3,]    0    0    0    0    0    0    0    0    1    1    1    1
[4,]    1    0    0    0    1    0    0    0    1    0    0    0
[5,]    0    1    0    0    0    1    0    0    0    1    0    0
[6,]    0    0    1    0    0    0    1    0    0    0    1    0
[7,]    0    0    0    1    0    0    0    1    0    0    0    1

```

Notice that such a manual coding of matrix A is not only boring but it is also prone to errors. This matrix is quite regular. The first 3 rows contain 4 consecutive 1s, starting at positions 1, 5 and 9, respectively. The last 4 rows contain digit 1 at positions 1, 1+4, 1+8 (in row 4), at positions 2, 2+4, 2+8, etc. All other values are zeroes. Using these distribution properties, we can generate this matrix programmatically.

R Code

```

# Define the size of the original decision (or cost) matrix.
# m = the number of the suppliers and n = the number of receivers.
m = 3
n = 4

# Create a matrix of zeros, having m + n (here 7) rows and
# m * n (here 12) columns, as shown above in (4.14).
A = matrix(0, nrow=m+n, ncol=m*n)
A
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]    0    0    0    0    0    0    0    0    0    0    0    0
[2,]    0    0    0    0    0    0    0    0    0    0    0    0
[3,]    0    0    0    0    0    0    0    0    0    0    0    0
[4,]    0    0    0    0    0    0    0    0    0    0    0    0
[5,]    0    0    0    0    0    0    0    0    0    0    0    0
[6,]    0    0    0    0    0    0    0    0    0    0    0    0
[7,]    0    0    0    0    0    0    0    0    0    0    0    0

# Produce the first m rows.
for (i in 1:m) {
  for (j in 1:n) {
    A[i, j+n*(i-1)] = 1
  }
}

# Produce the next n rows.

```



```

for (j in 1:n) {
  for (i in 1:m) {
    A[m+j, j+n*(i-1)] = 1
  }
}
# Show the matrix.
A
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]    1    1    1    1    0    0    0    0    0    0    0    0
[2,]    0    0    0    0    1    1    1    1    0    0    0    0
[3,]    0    0    0    0    0    0    0    0    1    1    1    1
[4,]    1    0    0    0    1    0    0    0    1    0    0    0
[5,]    0    1    0    0    0    1    0    0    0    1    0    0
[6,]    0    0    1    0    0    0    1    0    0    0    1    0
[7,]    0    0    0    1    0    0    0    1    0    0    0    1

# Since all our constrains represent equalities (=), we can generate
# a vector of m + n equal signs, using function rep.
const_dir = rep("=", n+m)
const_dir
[1] "=" "=" "=" "=" "=" "=" "="

# We have all the necessary ingredients for the solveLP() function.
library(linprog)

opt = solveLP(cvec = vc, bvec = vb, Amat = A, maximum=FALSE,
const.dir= const_dir, lpSolve=TRUE)
# Find out if the function has succeeded.
opt$lpSolve
[1] TRUE

# Show the solution (shipments).
opt$solution
  1  2  3  4  5  6  7  8  9 10 11 12
  5  0  0  0  2  8  0  0  5  0  4  6

# Show the value of the objective function (the total shipment cost
# for the optimal silution.
opt$opt
[1] 89

```

Visualizing the solution as a matrix will make it easier to understand what is going on. Let's convert the solution back to a matrix (x_{ij}).

R Code

```

# Convert the solution vector, opt$solution to matrix X, row by row.
X = matrix(opt$solution, nrow=m, ncol=n, byrow=TRUE)

```

```

X
# Also customize row and column labels as sources (S1, S2, S3) and
# destinations (D1,D2,D3,D4).
rownames(X) = paste("S",1:m,sep="")
colnames(X) = paste("D",1:n,sep="")
X
  D1 D2 D3 D4
S1  5  0  0  0
S2  2  8  0  0
S3  5  0  4  6

```

Notice that the first paste function combines character "S" with each of numbers 1 through m (here 1 through 3). Argument `sep=""` prevents a space to be inserted between the character and the number. The second paste function does a similar job to generate the column names. Alternative (boring) statements would be `rownames(X) = c("S1", "S2", "S3")` and `colnames(X) = c("D1", "D2", "D3", "D4")`.

Adding the supply and demand values, including the total supply/demand to this matrix will make it even more informative. The supply quantities are stored in the first m elements of vector `b` and the demand quantities—at remaining positions ($m+1$ through $m+n$).

R Code

```

# Extract the supply values from the RHS vector b.
supply = vb[1:m]
# Calculate the total shipment.
total_shipment = sum(vb[1:m])
# Create vector demand, using the extracted demand values from the RHS
# vector b and the total shipment.
demand = c(vb[(m+1):(m+n)], total_shipment)
# Combine matrix X with the supply column.
tr_report = cbind(X, supply)
# ombine matrix X with the demand row.
tr_report = rbind(tr_report, demand)
# Show the transportation solution.
tr_report
  D1 D2 D3 D4 supply
S1  5  0  0  0  5
S2  2  8  0  0  10
S3  5  0  4  6  15
demand 12  8  4  6  30

```

Notice that **R** adopted the names of vectors `supply` and `demand` for the names of the last row and the last column.

It is now easier to interpret the solution. Supplier S_1 ships 5 units to destination D_1 . The latter also gets 2 units from S_2 and 5 units from S_3 . Supplier 2 ships 8 units to receiver 2. Finally supplier S_3 also ships 4 units to D_3 and 6 units to D_4 . This is an optimal solution, having the shipment cost of 89 ($opt\$opt$) monetary units.

VISUALIZATION

The network diagram, shown in *Figure 1*, was created manually, using a presentation program (Microsoft Office - PowerPoint). Interestingly, **R** can generate similar diagrams with a help from package **igraph**. If necessary, install and open this package, create a network (graph) model for this transportation problem, and plot the model.

The following **R** code shows a plain graph (vertices and edges), using the default layout.

R Code

```
# Install package igraph.
packages.install('igraph')

# Load the package.
library('igraph')

# Set a graph (network model), consisting of nodes and edges.
# The X matrix is part of the above solution.
nodes = c(rownames(X), colnames(X))

# Create a vector of the source nodes.
from = c('S1', 'S1', 'S1', 'S1', 'S2', 'S2', 'S2', 'S2', 'S3', 'S3', 'S3', 'S3')

# Create a vector of the destinations nodes.
to = c('D1', 'D2', 'D3', 'D4', 'D1', 'D2', 'D3', 'D4', 'D1', 'D2', 'D3', 'D4')

# Notice that pairs of from and to elements define the edges of the graph.
edges = data.frame(from, to)

# Create the graph.
transp_net = graph_from_data_frame(d = edges, vertices =
nodes, directed = TRUE)

# Display the graph.
plot(transp_net)
```

This diagram uses a default layout (**layout_nicely**) that figures out the vertex coordinates based on the vertices themselves. Other [built-in] layouts can be applied or a matrix of an (x, y, z) grid can be provided for a custom layout. The following modification uses a custom layout as show in Figure 5). This figure also shows the diagram. Its layout is similar to that of Figure 1.

R Code

```
# Set X and Y coordinates for the layout (Z is needed for a 3D diagram).
x = c(0,0,0,1,1,1,1)
```

$$y = c(5, 3, 1, 6, 4, 2, 0)$$

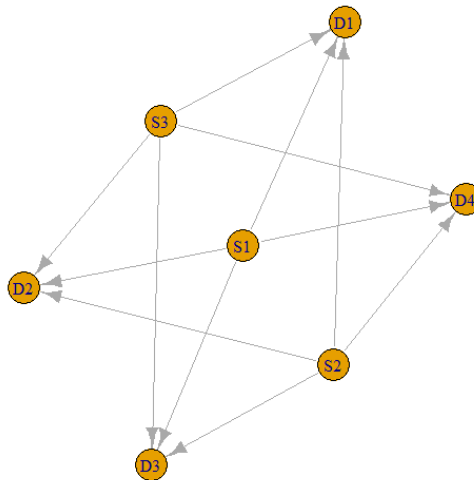


Figure 4 An *igraph* diagram for the transportation problem.

```
# Create a layout matrix.
cust_layout = as.matrix(data.frame(x,y))
# Display the graph.
plot(transp_net, layout = cust_layout)
```

Vertices of *igraph* diagrams are automatically annotated with names defined by the `vertices` argument when the *igraph* object is created. In our example, the object is set by the `graph_from_data_frame()` function. One can also provide annotations (labels) for the edges of the diagram. The following extension of our example shows such labels composed of the optimal solution and unit cost values. For each label the optimal solution is followed by the unit cost enclosed in parentheses. Vector `caption` is applied to hold the labels.

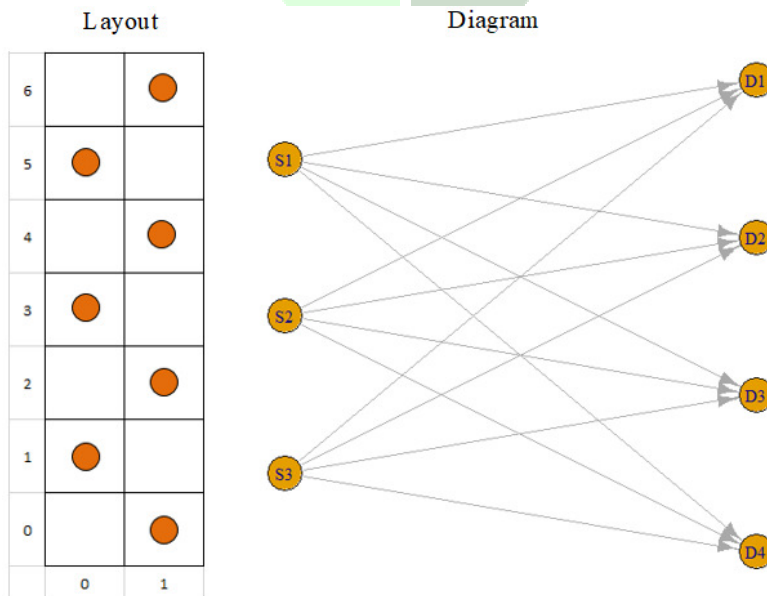


Figure 5 A custom layout and *igraph* diagram.

R Code

```

# Initialize graph vectors.
from = c()
to = c()
caption = c()

# Using the previously developed vectors and matrices, complete the
# vectors.
C = matrix(vc, nrow = m, ncol=n, byrow=TRUE)

for (i in 1:m ) {
  for (j in 1:n ) {
    from = c(from,rownames(X)[i])
    to = c(to,colnames(X)[j])
    cost = C[i,j]
    sol = X[i,j]
    = c(caption, paste(sol, '(', cost, ')', sep=''))
  }
}
caption
[1] "5(1)" "0(1)" "0(1)" "0(1)" "2(0)" "8(0)" "0(0)" "0(0)" "5(0)" "0(0)" "4(0)" "6(0)"

```

The above code for generating the captions (`caption`) is kind of traditional. It is OK but it requires two regular *for* loops. The same output can be obtained using vector-based operations. The following lines are an alternative to the above for loops.

```

# Convert the solution matrix X to a vector row-wise. Notice that for a
# row-wise conversion, the matrix must first be transformed.
sol = as.vector(t(X))
# Convert the cost matrix C to a vector row-wise.
cost = as.vector(t(C))
# Generate the captions using vectors sol and cost.
caption = paste(sol, '(', cost, ')', sep='')
caption
[1] "5(2)" "0(3)" "0(5)" "0(6)" "2(2)" "8(1)" "0(3)" "0(5)" "5(3)" "0(8)" "4(4)" "6(6)"

```

As you can see the captions are the same.

```

# Get the supplier names (symbols) from matrix X.
supp = rownames(X)
supp
[1] "S1" "S2" "S3"
# Get the receiver names (symbols) from matrix X.
rcvr = colnames(X)
rcvr

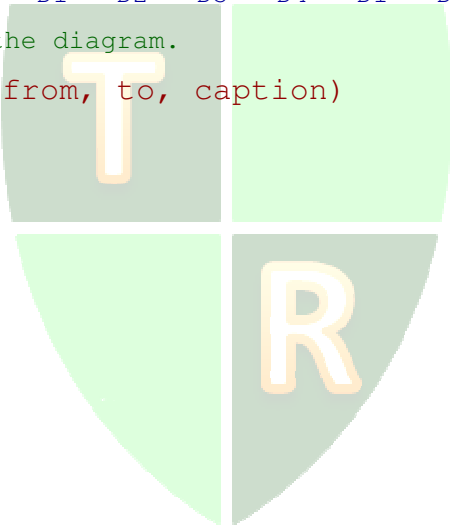
```

```

[1] "D1" "D2" "D3" "D4"
# Calculate the sizes of the vectors.
m = length(supp)
n = length(rcvr)
cat(m, ",", n, sep=" ", "\n")
3,4
# Get the origins of the shipments (from).
from = rep(supp, each=n)
from
[1] "S1" "S1" "S1" "S1" "S2" "S2" "S2" "S2" "S3" "S3" "S3" "S3"
# Get the destinations of the shipments (to).
to = rep(rcvr, m)
to
[1] "D1" "D2" "D3" "D4" "D1" "D2" "D3" "D4" "D1" "D2" "D3" "D4"
# Create the edges of the diagram.
edges = data.frame(from, to, caption)
edges
  from to caption
1   S1 D1    5(2)
2   S1 D2    0(3)
3   S1 D3    0(5)
4   S1 D4    0(6)
5   S2 D1    2(2)
6   S2 D2    8(1)
7   S2 D3    0(3)
8   S2 D4    0(5)
9   S3 D1    5(3)
10  S3 D2    0(8)
11  S3 D3    4(4)
12  S3 D4    6(6)

# Create the nodes of the diagram.
nodes = c(rownames(X), colnames(X))
nodes
[1] "S1" "S2" "S3" "D1" "D2" "D3" "D4"
# Create the igraph object (transportation network).
trans_net = graph_from_data_frame(d = edges, vertices = nodes,
directed = TRUE)
# Attach the labels (caption) to the graph.
edge_attr(trans_net, 'label') = caption
# Show the graph.
plot(trans_net)

```



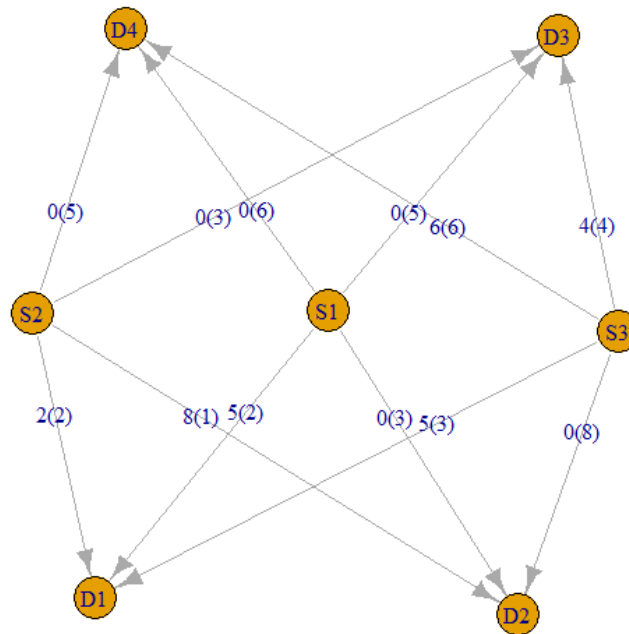


Figure 6 The transportation model with annotated edges.

Some of the *igraph* properties can be customized. The following example shows the same diagram with a larger vertex size (30), increased font size the the vertex labels (1.5), reduced font size of the edge labels (0.8), as well as with decreased size of the arrow heads (0.75).

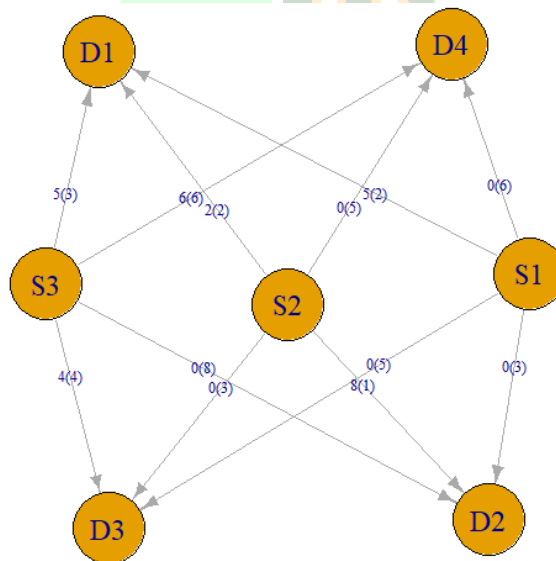


Figure 7 The transportation model with annotated edges and customized properties.

R Code

```
# Display the customized diagram.  
plot(trans_net, vertex.size = 30, vertex.label.cex = 1.5,  
edge.label.cex = 0.8, edge.arrow.size=0.75)
```

Other features of the graphical models can also be controlled. The reader may wish to explore references [2], [4], and many Web resources dedicated to *igraph* in **R**.

TEACHING /LEARNING SUGGESTIONS

Ask students to manually draw the network model and to get a feasible solution. An interesting exercise would be to do it all in a spreadsheet.

Run the **R** code gradually. Apply program segments and discuss the partial outcomes. Ask the students for ideas to come up with alternative **R** statements.

Have the students search for different visualization options, for example, using other **R** packages.

SUMMARY

Doing transportation problems in **R** is an adventure. There are hard things, especially when dealing with preparing input data, and fun aspects when it comes to visualization of the network models. The student learn a lot about computer programming, usually appreciating the capability of **R** with regard to handling vectors and matrices as well as utilizing the powerful libraries.

R should be given a serious consideration as a problem-solving tool and as an easy to adopt programming language. One does not have to be a seasoned programmer in order to turn **R** into a useful application development tool. It has a very high potential for being adopted to courses for Business Analytics, in general, and MS / OR, in particular.

REFERENCES

- [1] Anderson D. R., Sweeney D. J., Willimas T. A. Quantitative Methods for Business, Sixth Edition. West Publishing Company 1995.
- [2] Letkowski, J. R Code for Cases in Network Optimization with **R**. URL: <http://letkowski.us/optimization/network/SanAntonio2020.html>
- [3] Ognyanova K. Network visualization with **R**. URL: <https://kateto.net/network-visualization> (October 3, 2019).
- [4] Ragsdale C. T. Spreadsheet Modeling & Decision Analysis, A Practical Introduction to Management Science, 5E. Thomson, South-Western 2008.
- [5] Sadler J. Introduction to Network Analysis with **R**. URL: <https://www.jessesadler.com/post/network-analysis-with-r/> (October 3, 2019).
- [6] Wagner H. W. Principles of Operations Research, Second Edition. Prentice-Hall, Inc. 1975.
- [7] Winston W. L. Operations Research, Third Edition. Duxbury Press, 1994.

