**1.**

Function adj = random\_graph(n,degree\_p,degree,time\_p,time)

% n: number of nodes

% degree\_p: percentage of degrees

% degree: degrees for nodes

% time\_p: percentage of time

% time: time array

% example: random\_graph(10,[50 30 20],[1 2 3],[40 30 20 10],[1 2 3 5])

Check sum of degree percentages and time percentages are 100 and if not, return;

For 1 to length of percentages of degrees (or degree)

Calculate the probability distribution of degrees and get the number of elements for each degree and assign it to node.

End

Get the number of edges from the number of nodes for each degree

% number of edges = sum(degree\*numofdegree)/2;

For 1 to length of percentages of time

Calculate the probability distribution of times and get the number of edges for each time value

End

Check the given degrees satisfy the graphic conditions using isDegree function

and if not, return.

Set Old degrees as degrees

While sum of degrees are bigger than zero (all the degrees are not yet satisfied)

If counter is bigger than 100 (if algorithm gets same loops more than 100 times)

Initialize the degrees as old degrees

Reset adj as zero matrix

Reset counter as 0

End

Set the current sum as sum of current degrees and compare it with old sum

If both values are equal, increase the counter by one.

Select the node with maximum degrees as first node.

Select the random node as second node

If second node is equal to the first node, go to the next for loop.

Set the flag for finding as 1.(it means that time for edge is not assigned yet)

While the flag of finding is 1

Get the random percentage value

Set the time for edge based on random percentage value and probability distribution for time values and decrease the number of nodes for selected time if the number of nodes for selected one is not zero yet.

If time is assigned to the edge, flag for finding is set as 0

End

Assign time to the new edge and decrease the degrees for selected nodes.

End

**2.**

Function res = isDegree(degrees)

%objective: check degrees are valid

%degrees: array of degrees

If there are non-positive degrees or their sum is odd return false

For k=1:number of nodes-1

Calculate the sum of k numbers of largest degrees

Calculate the sum of remain degrees

Check sum of largest degrees are possible degrees and it not return false

Return true

End

**3.**

Test.m

Initialize the parameters for graph

Set correct\_graph as zero

While correct\_graph is zero

Get the random graph

Calculate the lengths of shortest path for the first node and other nodes and if there is no infinite value, set the correct\_graph as 1.

(This means that given graph is not separated and there is possible path for arbitrary two nodes)

End

Plot graph

Get the parameters for exponential distribution using given demands.

For k = 1 to number of Demands

Select the random node as first node

Select the random node as second node

While first node is different as second node, select the nodes again

If uniform distribution random value between (0,1) is smaller than 0.356

Set demand between selected nodes as zero

Else

Set the demand between selected nodes as the exponential random value

End

End

Get all linked nodes and time between them to the newTravelTime

Save the time and demands in the excel file