Algorithmic Aspects of Telecommunication Networks

CS 6385.001- Fall 2017

PROJECT 1

AN APPLICATION TO NETWORK DESIGN

Submitted by:

Ponmalar Silambaram Chandrabose (pxs162030)

CONTENTS:

An Application to Network Design

Objective:

Create software that is capable of doing the following:

- As input, it receives the number of nodes (N), the traffic demand values (bij) between pairs of nodes, and the unit cost values for the potential links (aij).
- As output, the program generates a network topology, with capacities assigned to the links, according to the studied model, using the shortest path based fast solution method (see at the end of the referred lecture note). The program also computes the total cost of the designed network.

To show graphically the following:

- Dependence of k on cost
- Density dependency on k
- Obtained Network Topologies for k=3,8 and 15

To give a brief justification of why the diagrams or graphs look the way they do.

Overview:

Steps in developing the algorithm:

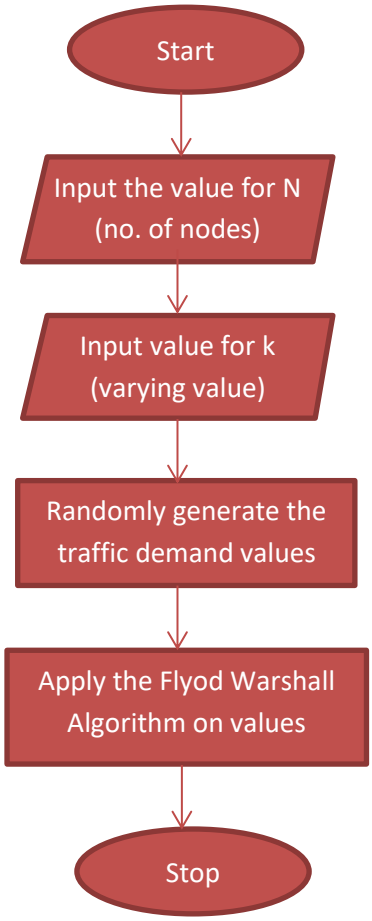
1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution in a bottom-up manner.

The plan overview:

The input for the number of nodes to be drawn in the network topology is gathered from the user Then, get from the user the input value for the parameter k (This keeps varying). Then, gather the traffic demand values between the nodes from the range [0, 1, 2, 3] through random generation during compilation.

Once these values are provided and accounted for, the code will generate a sample outline graph with the respective values. The program then applies the Floyd Warshall's Algorithm to find the shortest path between the nodes.

Then, the cost and density of the resultant graph is generated. This can be diagrammatically represented as follows:



The Floyd-Warshall Algorithm:

This algorithm compares all possible paths through a graph between each pair of vertices. There are a large number of vertices available in the graph and every combination of edges is tested by the Floyd-Warshall algorithm. This is performed by incrementally improving an estimate on the shortest path between two vertices.

Let's consider a graph G having vertices or nodes numbered 1 through N. I will be using a function names shortestPath with 3 arguments i,j, and k where i and j which are nothing but the nodes, whereas k is nodes that will be considered for finding the shortest path between each i and each j, and when k value is provided then the set of nodes from 1 to *k* are considered for the shortest path as intermediate points. Given this function, our objective is to find the shortest path from each *i* to each *j* using only nodes 1 to *k* + 1. For each the true shortest path can be

- A path that does not go through k (vertices in set {1,…, k-1})
- A path that does go through k (intermediate vertices in {1,...., k-1}})

The base cases for the definition of shortest path (i,j,k) is:

Shortestpath(i,j,0)=w(i,j)

And the recursive case is:

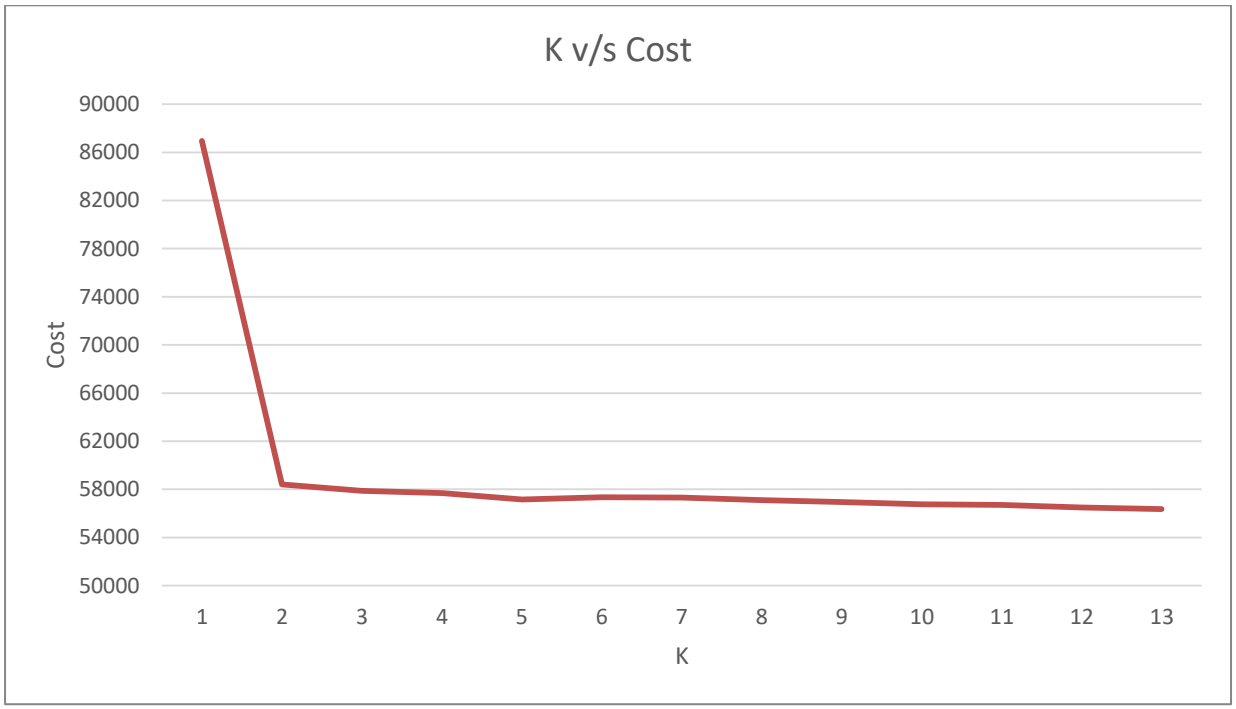
Shortestpath (i,j,k)= min(Shortestpath(i,j,k-1), Shortestpath(I,k,k-1)+ Shortestpath (k,j,k-1))

The pseudocode for the Floyd-Warshall Algorithm is as follows:

```
/*let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
*/
 for each vertex v
   dist[v][v] ← 0
 for each edge (u,v)
   dist[u][v] ← w(u,v)  // the weight of the edge (u,v)
 for k from 1 to |V|
   for i from 1 to |V|
     for j from 1 to |V|
       if dist[i][j] > dist[i][k] + dist[k][j]
         dist[i][j] ← dist[i][k] + dist[k][j]
       end if
```

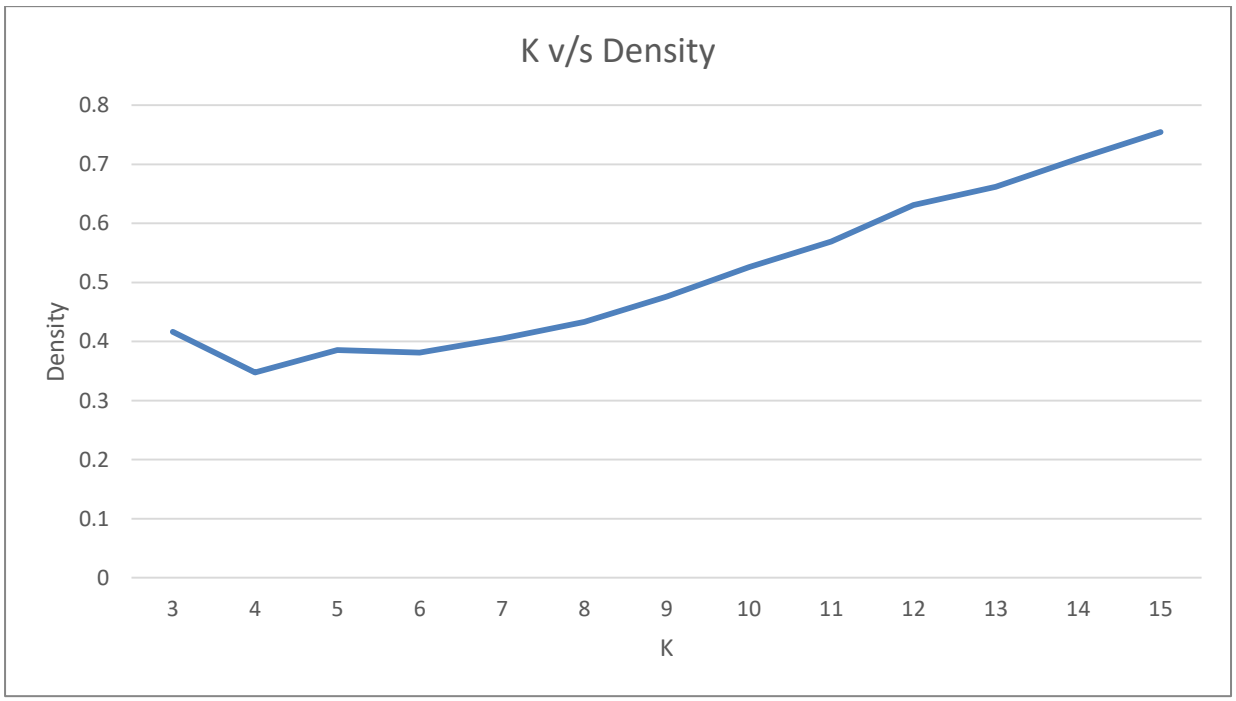
Graphical representation of Dependency:

K v/s Cost:



From the line chart of K and cost, it is clear to see that the cost is highest when k=3 and lowest when k=15. This implies that as the number of edges increases, the cost goes down.

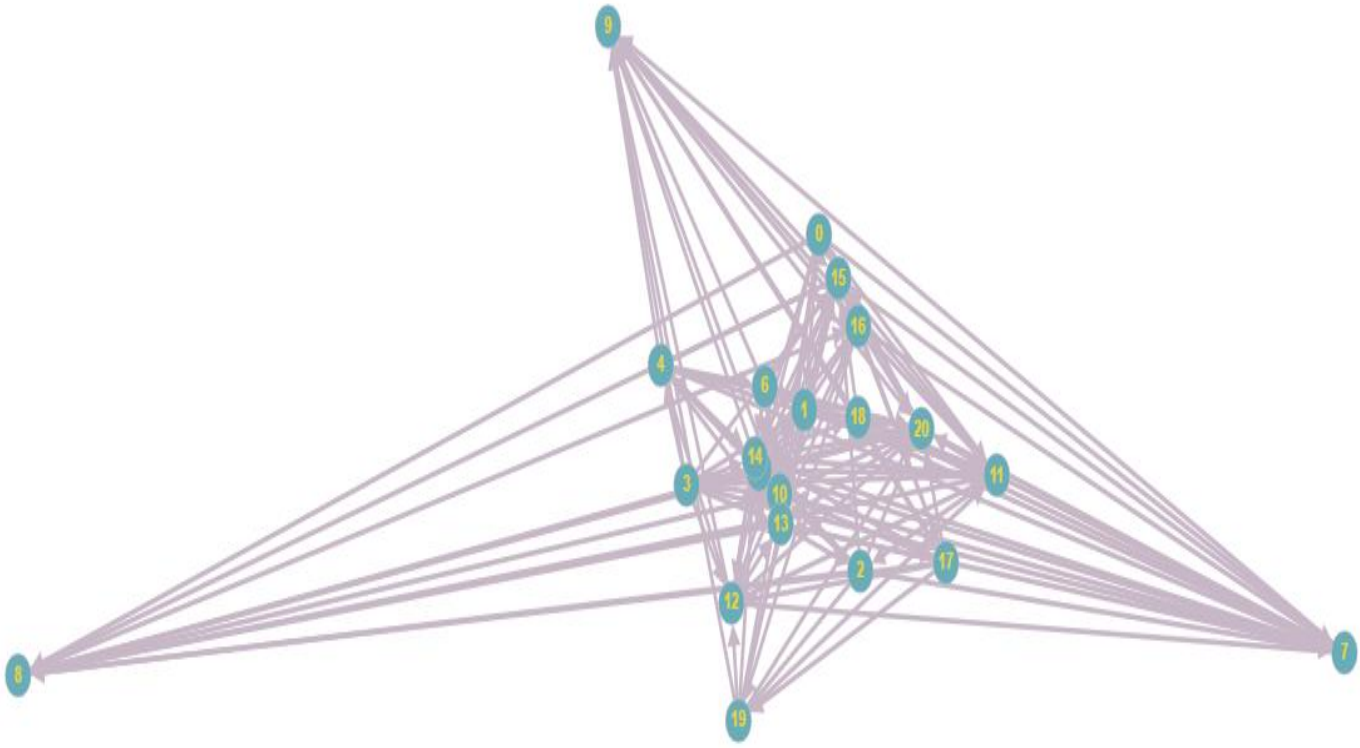
K v/s Density:



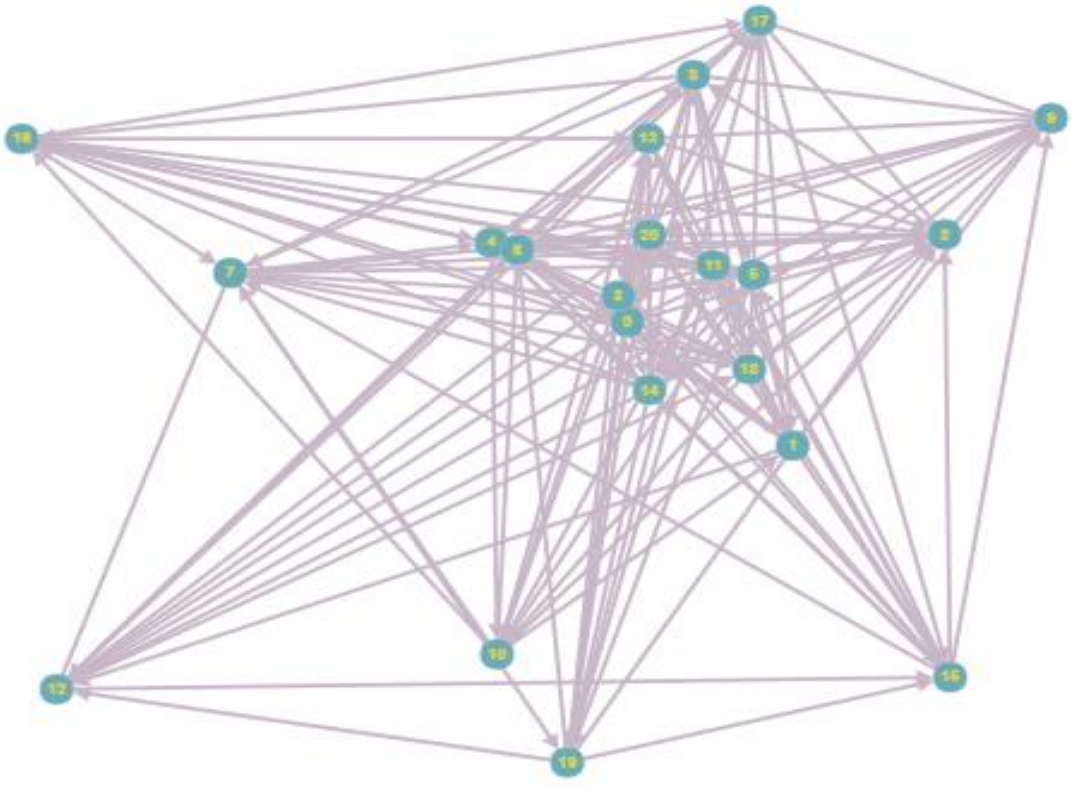
From the above graph between K and Density it is evident that the density of the system goes up as the total number of edges increases.

Graphs:

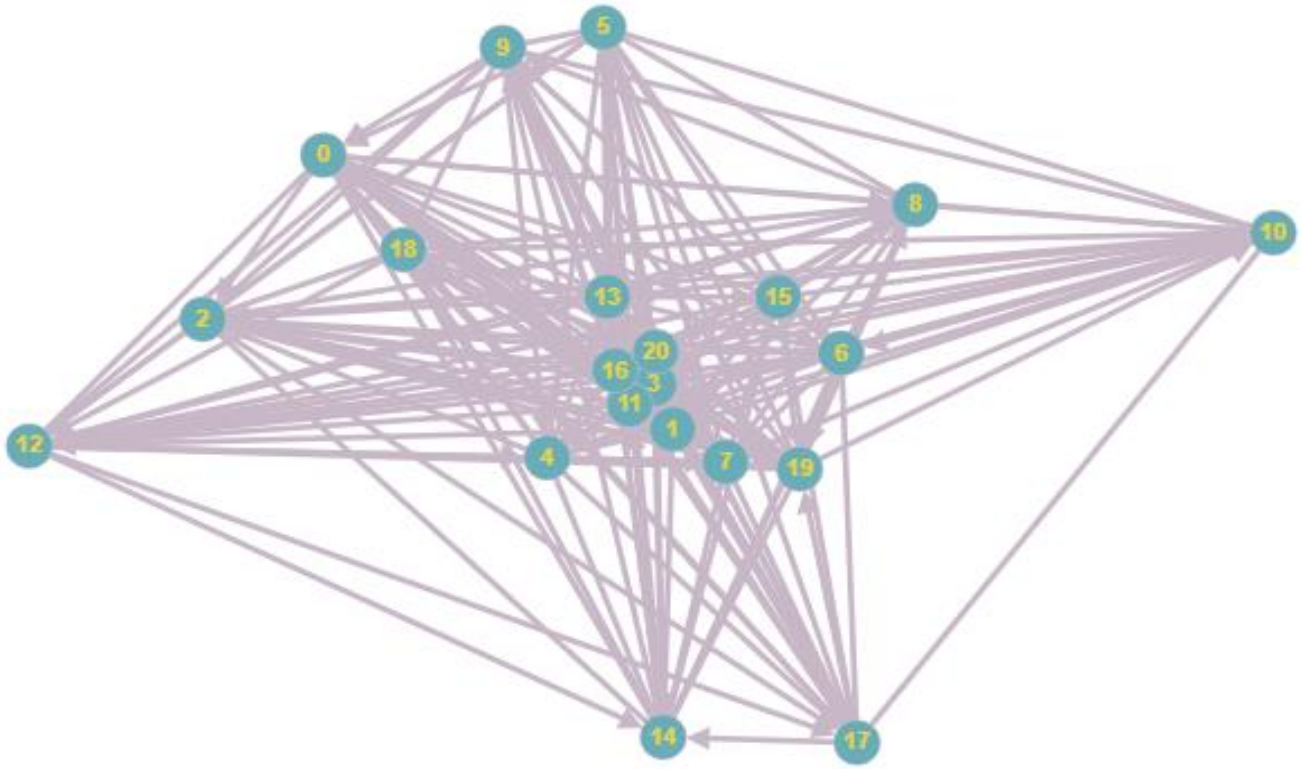
K=3:



K=8:



K=15:



Appendix:

The runnable code:

https://onedrive.live.com/embed?cid=9138B87757D637D3&resid=9138B87757D637D3%21859&authkey=AB5XESIHyj9YFLw

References:

1. https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm
2. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. (1990). *Introduction to Algorithms* (1st ed.). MIT Press and McGraw-Hill. ISBN 0-262-03141-8
3. http://graphonline.ru/en/

.