International Journal Of Core Engineering & Management
Volume-6, Issue-11, 2021 ISSN No: 2348-9510

NETWORK TOPOLOGY GENERATION USING THE SHORTEST PATH MODEL

Krishna Mohan Pitchikala
Graduate Student
University of Texas at Dallas
Texas, USA.

Abstract

Network topology refers to the way in which a network is configured, including the physical or logical way the links and nodes are organized. Building an efficient network topology is essential to build an optimized connection between various networks. These networks include both communication networks and transportation system. An efficient network topology is the one where any two nodes are connected by the most possible shortest paths. In this paper we focus on generating a network with capacities assigned to the links using shortest path-based solution method. We will calculate the total cost and density of the network using the Floyd-Warshall algorithm. We will also generate the network for various capacities of links and will compare how does it affect the cost and density.

Index Terms – Network topology generation, Floyd Warshall algorithm, Shortest path algorithm, Cost calculation in networks

I. INTRODUCTION

The placement of nodes and links in a network is known as network topology. This placement is crucial as it impacts how the data is transferred within that network. Network design is of great importance in several areas including but not limited to the computer network, telecommunications and logistics systems. The shortest path model has been used to create optimized networks by minimizing the time taken, latency and resources especially in transportations. This algorithm known as Floyd-Warshall which can find optimal paths between all available nodes is a perfect solution for the generation of new topologies and optimal distribution of new link strengths. Let us begin the walkthrough by speaking about Floyd-Warshall Algorithm.

The Floyd-Warshall Algorithm helps in finding the shortest paths between all pairs of vertices in the graph. In this graph each connection between the vertices (also known as edge) has a weight, or cost, associated with it. This algorithm works for graphs that are either directed (where the connections have a direction) or undirected (where the connections go both ways). But, it does not work well for graphs that have "negative cycles" – Which represents the loops in the graphs where the total weight of the edges is negative. In our case, since all edge weights are positive, we don't need to worry about negative cycles and so we can apply the Floyd Warshall algorithm [1].

A weighted graph means every connection between two points in a graph has a specific value, like a distance or cost. The Floyd-Warshall algorithm uses a method called dynamic programming. It solves the problem step by step by breaking it into smaller pieces and then combining those solutions towards the end. The idea of the algorithm is clear which is to find the shortest path from one point (A) to another (C). This is done by either using the shortest path that is already found, or by checking if going through a third point (B) makes the total journey shorter (from A to B, then B to C) [1].

II. FLOYD WARSHALL ALGORITHM

The Floyd-Warshall algorithm evaluates all available connections between a given pair of nodes in the graph and chooses the shortest one. It tries to cut down the cost required to travel from one node to another by checking whether going through an additional node would reduce the total distance travelled. This procedure considers every other node every time, making sure danger is minimal and shortest paths persist [4].

A. Steps

1. Initialize distances:
    a. You create a 2D array dist, where dist[u][v] represents the current shortest known distance from node ‘u’ to node ‘v’.
    b. Initially, the distance between any two nodes is set to infinity (∞), except for:
        i. If there’s an edge between nodes ‘u’ and ‘v’, set ‘dist[u][v]’ to the weight of that edge.
        ii. The distance from any node to itself is set to 0 (dist[v][v] = 0), since there's no distance required to stay at the same node.

2. Iterate over all nodes:
    a. The algorithm examines every possible path through an intermediate node ‘p’ to check if there’s a shorter path from node ‘u’ to node ‘v’ via node ‘p’.
    b. For each pair of nodes ‘u’ and ‘v’, if traveling through node ‘p’ results in a shorter path than the current known path, update dist[u][v] to the new shorter distance.

3. Update shortest paths:
    a. This is the key step in the algorithm. This checks if the current known distance dist[u][v] can be improved by going through node ‘p’. If the new path through `k` is shorter, it updates dist[u][v] with the shorter distance

B. Pseudo code

Pseudo code for Floyd Warshall Algorithm [11]

```
let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
for each edge (u, v) do
    dist[u][v] ← w(u, v)  // The weight of the edge (u, v)
for each vertex v do
    dist[v][v] ← 0
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
```

III. APPROACH

We will create a software that designs a network based on inputs of node connections, traffic demands, and costs. The program will:

1. Take inputs on node connections and traffic between nodes.
2. Use the shortest path algorithm to design a network that minimizes the total cost.
3. Output the network design with link capacities and total cost.

Inputs to the program:

- Number of nodes (N)
- Traffic demand values(b_{ij})
- Unit cost values (a_{ij})

Output from the program:

- Network Topology
- Total Cost to design the network
- Graphs to denote the dependency between Cost and 'K'
- Graphs to denote the dependency between Density and 'K'

('K' is the number that denotes the number of low-cost links in the network, and it changes through the program)

Step-by-step-breakdown:

1. A 25-digit number is generated by appending our ID (student Id) to itself until the desired length is obtained. This is represented by 'd' and d_i corresponds to the ith digit in d
2. A traffic demand matrix "b" is generated using the formula $b_{ij} = |d_i - d_j|$.

3. Generated a unit cost matrix "a" which is computed with varying values of k from 3 to 13.
4. On each of the generated matrix Floyd-Warshall algorithm is applied to find the shortest paths in this weighted graph with 25 nodes.
5. Cost and density are calculated at each iteration of k
6. All these values are stored in a list for plotting purpose
7. Network is generated and is displayed for the specific values of K which are 3 ,8 and 13.
8. Graph is plotted for 'K' versus Cost and 'K' versus Density.

IV. FLOW CHART

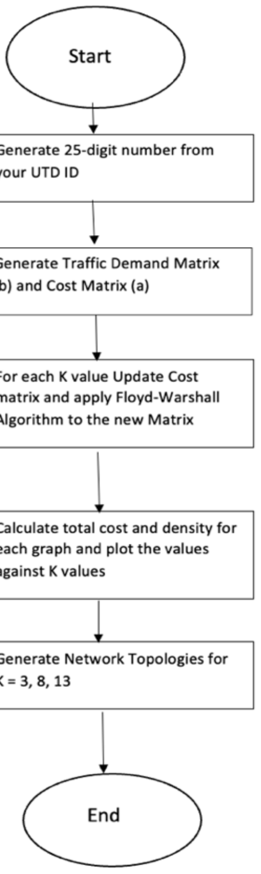
Start
Generate 25-digit number from your UTD ID
Generate Traffic Demand Matrix (b) and Cost Matrix (a)
For each K value Update Cost matrix and apply Floyd-Warshall Algorithm to the new Matrix
Calculate total cost and density for each graph and plot the values against K values
Generate Network Topologies for K = 3, 8, 13
End

Fig. 1. Flowchart of the program

V. RESULTS

The values of cost and density with respect to K can be shown below

Table I. K vs Cost vs Density

K	Total Cost	Density
3	12674	0.165
4	10872	0.2066
5	5978	0.2083
6	4895	0.25
7	4755	0.291
8	3900	0.33
9	4128	0.375
10	3589	0.4167
11	3548	0.4583
12	3067	0.5
13	3136	0.5416

Hence, we can conclude that **Total cost is Inversely Proportional to the value K**. This can be clearly shown in graph below

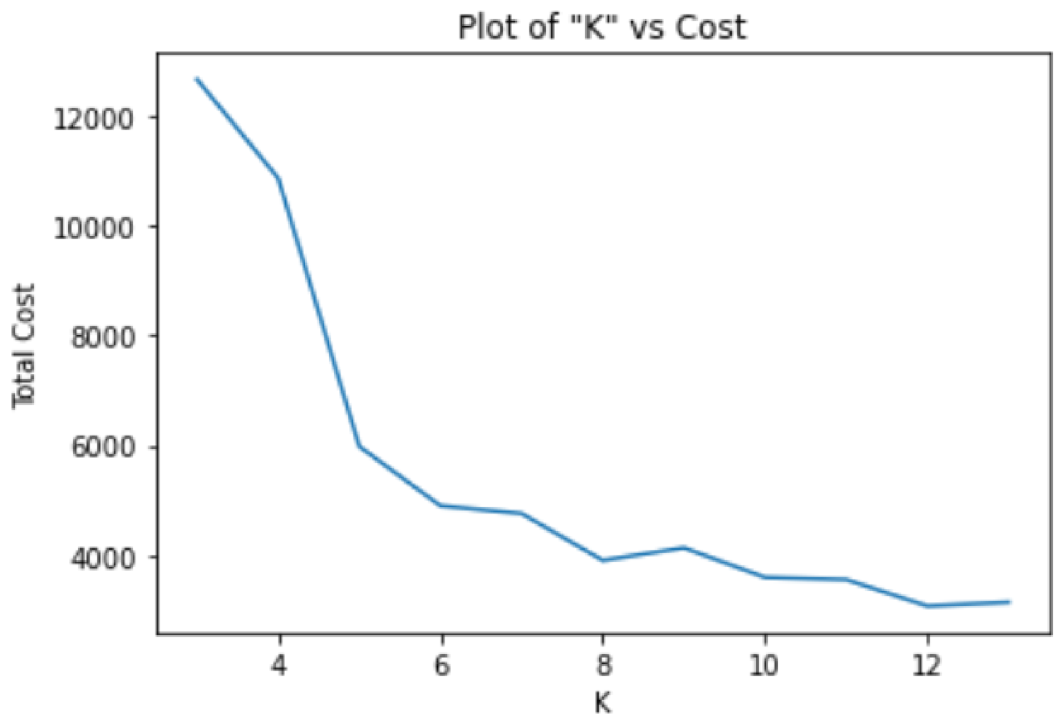

Fig. 2. Plot of Total Cost vs K

International Journal Of Core Engineering & Management
Volume-6, Issue-11, 2021 ISSN No: 2348-9510

Also, we can conclude that **Density is directly Proportional to the value K**. This is clearly understood from the below plot

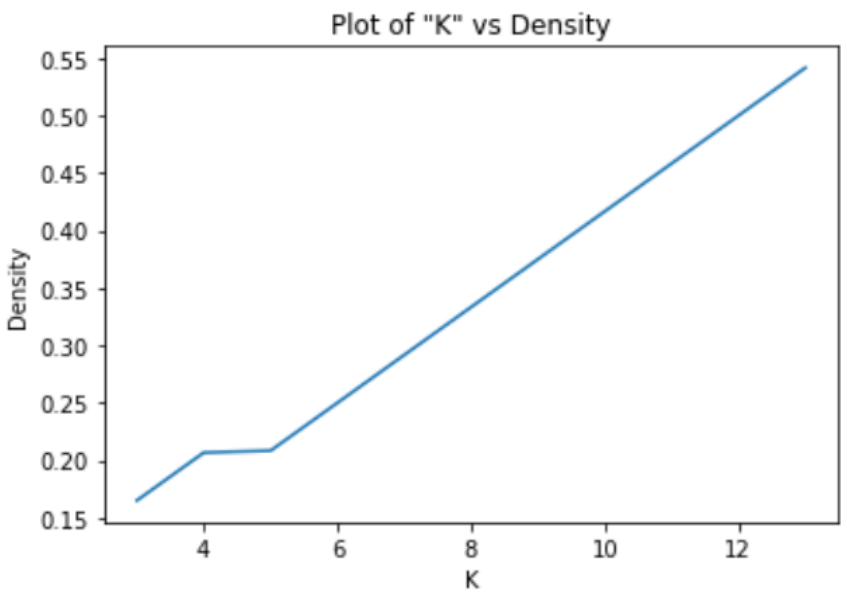

Fig. 3. Plot of Density vs K

Network Topologies for the K values 3, 8 and 13

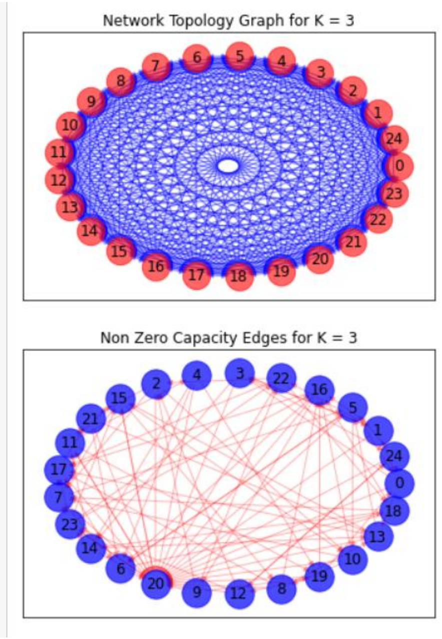
Network Topology Graph for K = 3
Non Zero Capacity Edges for K = 3

Fig. 4. Network Topology for K=3

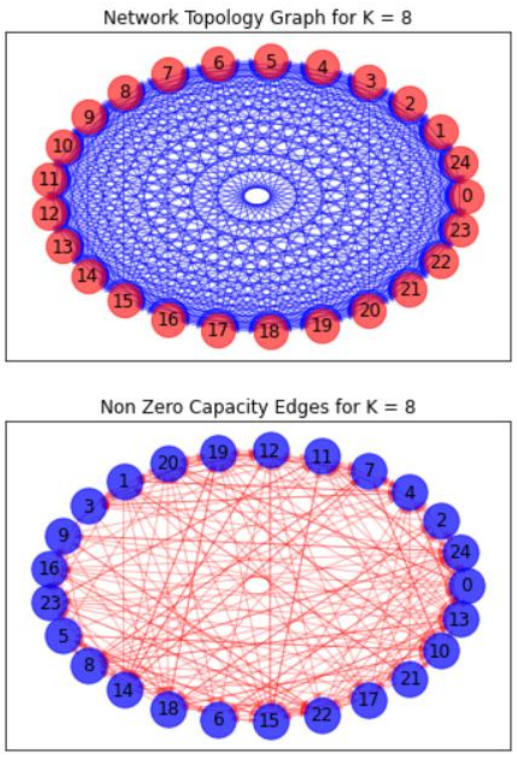
Network Topology Graph for K = 8
Non Zero Capacity Edges for K = 8

Fig. 5. Network Topology for K=8

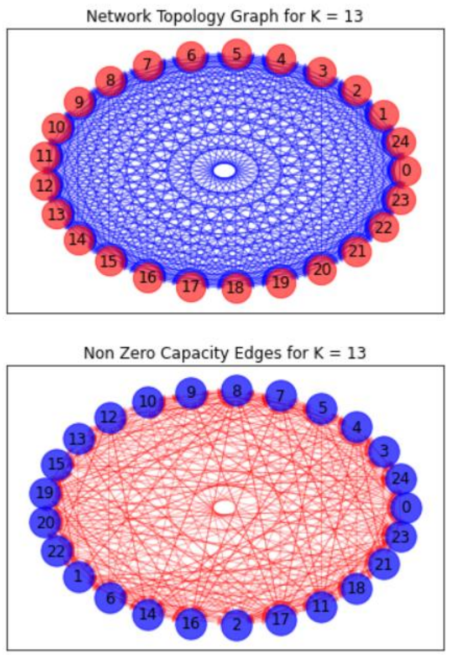
Network Topology Graph for K = 13
Non Zero Capacity Edges for K = 13

Fig. 6. Network Topology for K=13

International Journal Of Core Engineering & Management
Volume-6, Issue-11, 2021 ISSN No: 2348-9510

VI. CONCLUSION

As we see the cost and density of the network depends greatly on the network parameter 'K'. This is because the change in value corresponds to the change in the cost matrix 'a'. The variable aij takes only 3 values 0, 250 and 1. Here 0 represents cost to self-node and all the other edges are of cost 250. But depending upon the values of K we change 'K' number of edges form each node to be 1 which indicates the lower cost edge compared to all the other edges.
So, when we increase the number of low-cost edges on the graph the density of the graph increases obviously as we concentrate on finding the shortest path edges and increment in K results in increase of number of such edges. Whereas the cost of the edges is cheaper compared to all the other edges and hence it is obvious that Cost of the network decreases with increase in the value 'K'.

REFERENCES

1. https://brilliant.org/wiki/floyd-warshall-algorithm/
2. Kohei Arai, "Routing Protocol based on Floyd-Warshall Algorithm Allowing Maximization of Throughput" International Journal of Advanced Computer Science and Applications(IJACSA), 11(6), 2020
3. https://doi.org/10.48550/arXiv.1807.10787
4. https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm/
5. https://networkx.org/documentation/latest/_downloads/networkx_reference.pdf
6. https://courses.cs.vt.edu/~cs3114/Fall10/Notes/T22.WeightedGraphs.pdf
7. Laurito, M. Bonaventura, M. E. Pozo Astigarraga and R. Castro, "TopoGen: A network topology generation architecture with application to automating simulations of software defined networks," 2017
8. https://www.tek-tools.com/network/best-network-topology-software
9. https://www.cs.toronto.edu/~lalla/373s16/notes/APSP.pdf
10. https://transportgeography.org/contents/methods/graph-theory-measures-indices/cost-graph/
11. https://steemit.com/programming/@drifter1/programming-java-graph-all-pair-shortest-path-algorithms-floyd-warshall-johnson