Report

bij : demand of transporting data from node i -> node j

aij : cost for unit capacity on a link from node i -> node j

\* higher capacity proportionally more

Goal:

1. design which links will be built and with how much capacity

2. overall cost is minimum

3. assume graph is directed

Zij : will be the capacity we implement on link (i,j)

Cost of link(i,j) = aij \* Zij

The Fast Solution:

The cheapest way of sending bkl amount of flow from node k to l is to send it all along the path for which the sum of the link costs is minimum. If the path consists of nodes k = i1,i2 …… ir =l, the resulting cost is :

bkl (ai1,i2 + …. + air-1,ir)

what we need to do:

1.find a minimum cost path between each pair k,l of nodes, with edge weights aij, can be done by any standard shortest path algorithm

2. set the capacity of link(i,j) to the sum of those bkl values for which (i,j) is on the min cost path found for k,l.

calcaluate Zij for each node in the graph and form the matrix, then call the shortest path algorithm

Using graphstream to visualize the graph

\* More advanced rendering will use gs-ui.jar

Display a graph: Graph.display()