1. getCompressedNode

%input:

Prevnd: previous node

Nd: current node

fulltravelTimeMatrix: travel time matrix for full graph

%output:

Compressed node

Append the current node to the compressed node

For each node in full graph

If node has not link with the current node, then go to the next iteration

Get the order of node (this node is a neighbor node to the current node)

If order of node is 2 and neighbor node is not previous node or current node,

append the node to compressed node and call getCompressedNode function in recursively using the current node as a previous node and selected node as a current node.

End

end

3. Main.m

-Initialize the parameters for simulation

Gen\_num<-100;

Population<-30

…

Minroutelen<-10

Maxroutelen<-35

-read demand and travel time data form excel files

-set elements of demand matrix and traveltime matrix for full graph

-get compressed nodes

numCompressNodes<-0 (number of nodes in compressed graph)

compressednode<-{}

for node=1 to number of nodes in full graph

[ind\_neigh]<- indexes of neighbor nodes

Order\_node<-order of current node (using getOrderOfNode function)

If order of node is greater than 2

numCompressNodes<-numCompressNode +1

append node as the node of compressed graph

else if order of node is equal to 2

for each of both neighbor nodes

get the order of neighbor node

if order of neighbor node is equal to 1(neighbor node is isolated node)

numCompressNodes<-numCompressNode +1

get the route starting from isolated node and ending at the node which has order of two and its neighbor node has order not equal to 2 and append it as a node of compressed graph

elseif order of neighbor node is greater than 2

get the route starting from current node and ending at the node which has order of two and its neighbor node has order not equal to 2 and append it as a node of compressed graph

end

end

else if node is isolated node

if neighbor node has order which is not equal to 2

numCompressNodes <- numCompressNodes+1

Append node as a node of compressed graph

end

end

end

get the travel and demand matrix for compressed graph

for each pair of nodes in a compressed node

demand<-total sum of demand between od pairs in full graph which has end stops in selected nodes which are contained in nodes of compressed graph

traveltime<-maximum of travel time between two node in full graph

Generate the initial solution and get the optimized solution using ma (this part is as same as in full graph)

3. Experiment to get the graph

We should do experiment using random graphs which have different compress rate.

To get the various graphs which has different compress rate, we should use the function random\_graph(n,degree\_p,degree,time\_p,time).

To change compress rate, we should change parameter degree\_p.

If the percentages of degree 2 increase, the compressing rate will be increased.

If the percentages of degree 2 decrease, the compressing rate will be decreased.

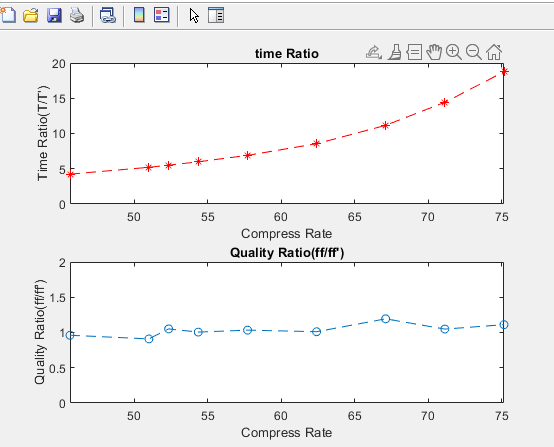
We use the same fitness function and same parameters for full graph and compressed graph.

We also 100 generations and 30 populations for MA algorithm.

ff\_fun(solution,demand,traveltime,0.4,0.5,0.1,0.6,0.3,0.1,5,10);

We did the experiments for 18 times, 9 for compressed graph and 9 for full graph.

The experiment results are as following.



The result shows that when compressing ratio increases, time Ratio increases squarely.

The optimization result when we use compressed graph is similar as the result when we use the full graph.

