Air-traffic Graph Network Analysis

Team Neo

EBAC 4 (Part Time), Institute of Systems Science, NUS

[Anusuya Manickavasagam: A0163300Y,

Chinnasubbareddygari Mohan Reddy: A0163433L ,

Gello Mark Vito: A0163448Y,

Indumathi Nagarajan: A0163441M,

Muni: A0163382E, Pradeep: A0163453H]

**Abstract**

This document will help the user explain various aspects of graph analysis done using different tools like Neo4J, CytoScope, and Gephi.

# Neo4J

Installation 🡪 First we need to install neo4J on our machine. Please google it as how to install neo4J. Alternatively you can follow this [LINK](https://neo4j.com/download/).

Data Loading 🡪 Once neo4J is installed, please create a new project and place the below attached cleaned source files in the path **Neo4j Desktop\Application\neo4jDatabases\database-981e9b63-b640-48cd-b512-506bb07a6962\current\import**.



Once done, please run the below scripts one by one in neo4J browser. This will load the relational data from csv files to more discoverable neo4J graph database.

1. CREATE CONSTRAINT ON (ap:AirPort) ASSERT ap.id IS UNIQUE;
2. USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM

'file:///airports.csv' AS line

WITH line

FOREACH (x IN CASE WHEN line.`airport\_id` IS NULL THEN [] ELSE [1] END |

MERGE (ap:AirPort {id: line.`airport\_id`})

SET ap.name = line.`airport\_name`,

ap.iata = line.`airport\_iata`,

ap.icao = line.`airport\_icao`,

ap.latitude = line.`airport\_latitude`,

ap.longitude = line.`airport\_longitude`,

ap.altitude = line.`airport\_altitude`,

ap.timezone = line.`airport\_timezone`)

1. CREATE CONSTRAINT ON (ctry:Country) ASSERT ctry.name IS UNIQUE;
2. USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM

'file:///airports.csv' AS line

WITH line

FOREACH (x IN CASE WHEN line.`country` IS NULL THEN [] ELSE [1] END |

MERGE (ctry:Country {name: line.`country`}))

1. CREATE CONSTRAINT ON (ct:City) ASSERT ct.name IS UNIQUE;
2. USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM

'file:///airports.csv' AS line

WITH line

FOREACH (x IN CASE WHEN line.`city` IS NULL THEN [] ELSE [1] END |

MERGE (ct:City {name: line.`city`}))

1. USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM

'file:///airports.csv' AS line

WITH line

MATCH (ap:AirPort {id: line.`airport\_id`})

MATCH (ct:City {name: line.`city`})

MERGE (ap)-[loc:IN\_CITY]->(ct)

1. USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM

'file:///airports.csv' AS line

WITH line

MATCH (ctry:Country {name: line.`country`})

MATCH (ct:City {name: line.`city`})

MERGE (ct)-[loc:IN\_COUNTRY]->(ctry)

1. USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM

'file:///routes.csv' AS line

WITH line

FOREACH (x IN CASE WHEN line.`source\_airport\_id` IS NULL THEN [] ELSE [1] END |

MERGE (ap:AirPort {id: line.`source\_airport\_id`}))

1. USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM

'file:///routes.csv' AS line

WITH line

FOREACH (x IN CASE WHEN line.`dest\_airport\_id` IS NULL THEN [] ELSE [1] END |

MERGE (ap:AirPort {id: line.`dest\_airport\_id`}))

1. USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM

'file:///routes.csv' AS line

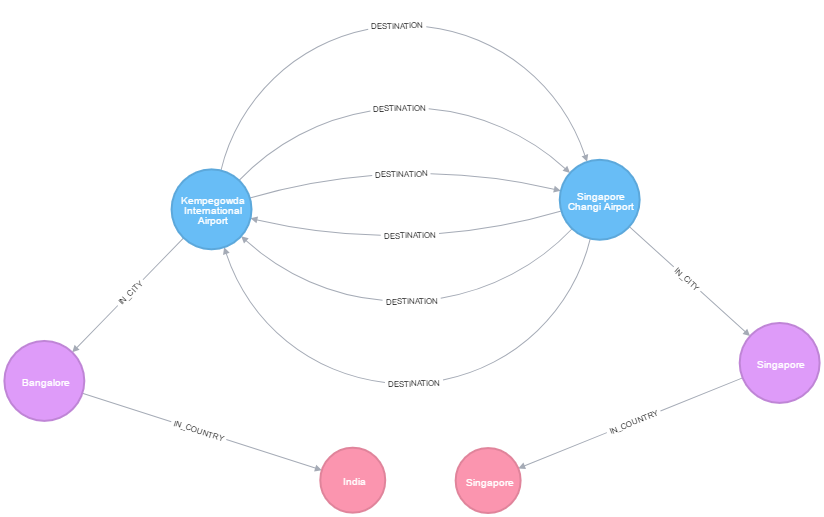
WITH line

MATCH (sap:AirPort {id: line.`source\_airport\_id`})

MATCH (dap:AirPort {id: line.`dest\_airport\_id`})

MERGE (sap)-[dt:DESTINATION { stop\_cnt: TOINT(line.`stops`), codeshare: line.`codeshare`, airline: line.`airline\_id`}]->(dap)

Above loading scripts will create the overall schema like below:



Now please run the below cypher scripts in Neo4J console to get answers of individual questions as mentioned below:

#################

# **Graph Schema**

#################

MATCH (sctry)<-[:IN\_COUNTRY]-(sct)<-[:IN\_CITY]-(sap)-[:DESTINATION]->(dap)-[:IN\_CITY]->(dct)-[:IN\_COUNTRY]->(dctry)

WHERE (sctry.name = 'Singapore') and dct.name = 'Bangalore'

RETURN sctry, sct, sap, dap, dct, dctry

#################################################

# **Which all airport does not serve any airline in Singapore**

#################################################

MATCH (ctry)<-[:IN\_COUNTRY]-(sct)<-[:IN\_CITY]-(sap)

WHERE NOT (sap)-[:DESTINATION]->() and (ctry.name = 'Singapore')

RETURN ctry, sct, sap

################################################

# **Which all airport serves at least one airline in Singapore**

################################################

MATCH (ctry)<-[:IN\_COUNTRY]-(sct)<-[:IN\_CITY]-(sap)

WHERE (sap)-[:DESTINATION]->() and (ctry.name = 'Singapore')

RETURN ctry, sct, sap

#########################################

# **Shortest path between Singapore and Chicago**

#########################################

MATCH (ct1:City {name: 'Singapore'}),(ct2:City {name: 'Chicago'}),

p = allShortestPaths((ct1)-[\*1..4]-(ct2))

RETURN p

###############################################################

# **Which all airport can be used as transit between Singapore and Mumbai**

###############################################################

MATCH (xy)-[:DESTINATION]->(ap)

WITH xy, count(\*) AS Count

MATCH (sct:City {name: 'Singapore'}),(dct:City {name: 'Mumbai'}),

p=(sct)<-[:IN\_CITY]-(sap)-[dt:DESTINATION]->(xy)-[dt1:DESTINATION]->(dap)-[:IN\_CITY]->(dct)

WHERE (toFloat(sap.latitude) < toFloat(xy.latitude) < toFloat(dap.latitude)) and (toFloat(sap.longitude) > toFloat(xy.longitude) > toFloat(dap.longitude))

RETURN p

####################################################################################

# **Which all airport can be used as transit between Singapore and Mumbai with transit airports handling less than 50 destinations**

####################################################################################

MATCH (xy)-[:DESTINATION]->(ap)

WITH xy, count(\*) AS Count

WHERE Count < 50

MATCH (sct:City {name: 'Singapore'}),(dct:City {name: 'Mumbai'}),

p=(sct)<-[:IN\_CITY]-(sap)-[dt:DESTINATION]->(xy)-[dt1:DESTINATION]->(dap)-[:IN\_CITY]->(dct)

WHERE (toFloat(sap.latitude) < toFloat(xy.latitude) < toFloat(dap.latitude)) and (toFloat(sap.longitude) > toFloat(xy.longitude) > toFloat(dap.longitude))

RETURN p

####################################################################################

# **Which all airport can be used as transit in their ascending order of traffic in terms of airline served**

####################################################################################

MATCH (xy)-[dt:DESTINATION]->()

WITH xy, count(distinct dt.airline) AS Count

MATCH (sct:City {name: 'Singapore'}),(dct:City {name: 'Mumbai'}),

p=(sct)<-[:IN\_CITY]-(sap)-[dt1:DESTINATION]->(xy)-[dt2:DESTINATION]->(dap)-[:IN\_CITY]->(dct)

WHERE (toFloat(sap.latitude) < toFloat(xy.latitude) < toFloat(dap.latitude)) and (toFloat(sap.longitude) > toFloat(xy.longitude) > toFloat(dap.longitude))

RETURN xy ORDER BY Count

# Cytoscope

Cytoscope is a useful graph analysis tool. We used this tool for observing the General Graph network measures.

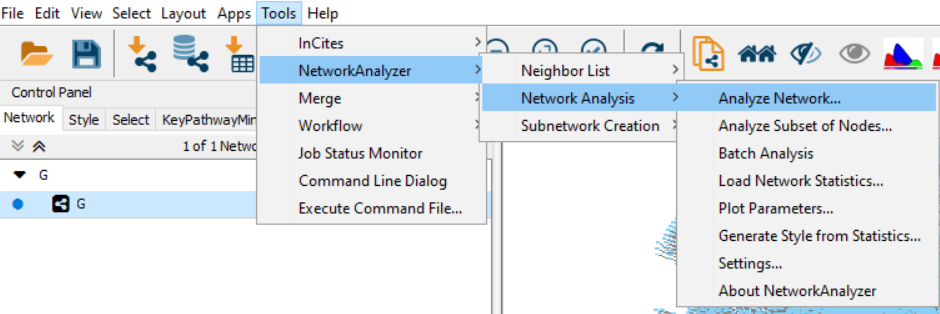
## How to use it

### Install and launch Cytoscope.

### Open the file .GraphML to import the graph in Cytoscope.

### It will take some time to map the graph on visualization workspace.

### From Tools -> NetworkAnalyzer->Network Anlaysis->Analyze Network run the network analysis as shown below:



### It will show up the general network measure like Degree distribution, centrality analysis, diameter, density, cluster coefficient etc.

* Similarly using the Motif\_discovery pattern, we can observe the recurrent pattern in the graph.

# Gephi For Key Players and Key Community Detection

Gephi is another popular tool to run the graph network analysis. We have used it to find the Key communities and Key players in our graph database.

## How To Use It

* Install and launch gephi
* Open the .graphml file in Gephi
* It will take some time to load the graph
* On the right hand side there are different statistical runs which can be done
* Click on Page Rank and Modularity.
* This will show the key Players and Key community by distribution in the graph database.

## References

* Neo4J - <https://neo4j.com/download/>
* Cytoscope - http://www.cytoscape.org/download.php
* Gephi - <https://gephi.org/users/download/>