Visualizing the underlying MRT network

By: Ernest Ng

Exploring the Data

This dataset contains data from ez-link tap-ins and tap-outs between 9 AM and 10 AM in a single day.

There are 381249 rows of data and I notice that we have many possible travel routes between each MRT station.

Some MRT stations have trailing capital letters which might represent a possible MRT line.

1. Problems Faced

➔ **Multiple train lines exist**

For station names with consecutive capital letters behind, they might exist on multiple train lines E.g Bugis DTL, Bugis NSEW

➔ **Reachability**

For one origin station, the dataset shows that many destination stations can be reached. To plot a network graph, we want to focus on stations that can be reached from one origin in 1 station travel. To determine reachability, we want to retrieve the journeys with the shortest possible travel time to get stations on the left and right of the origin station.

➔ **Does the station exist at the end or in the middle of the train line?**

There will definitely be stations that exist on the extreme end of the train line. How do we differentiate these stations from the ones that exist in the middle?

How do you choose which stations are adjacent with the ORIGIN station?

Short Answer

Look at all possible travel paths TO and FROM the origin station.

Aggregate the time taken for each and get the 2 paths with the shortest travelling time.

Assumptions:

Given a train line with stations (M-X-Y-Z)

Let's assume that time taken to travel from X to M and X to Y will be shorter than that from X to Z for all origin stations.

Time taken to travel 1 station < Time taken to travel 2 stations

Tip

t(X->Z) < t(X->Z)

t(X->M) < t(X->Z)

2. Initial Solutions to said Problems

➔ **Dealing with stations that exist on multiple train lines**

Treat each station on different lines as different stations. Eg treat Bugis DTL and Bugis NSEW as different stations. Since they are now considered different stations, we can get their adjacent stations.

➔ **How to determine reachability of a station?**

For all trips associated with origin X, get mean/median for each possible destinations. Sort the trips with origin X in increasing order and take the shortest 2. By our assumption, this will give us the stations adjacent to X.

➔ **End point or Middle point?**

If destinations for those shortest 2 paths are reachable from each other, then we can conclude that X is an endpoint, else X is in the middle of its train line. If there is only 1 possible shortest path from origin X after sorting, then we can also say that X is an endpoint.

Work Flow

1. Data Exploration
2. Data Cleaning
3. Data Manipulation (Feature Engineering)
4. Graph Implementation*

*I will continuously iterate over Steps 3 and 4 to make improvements to the resulting network graph.

Python Dependencies:

- Numpy
- Pandas
- MatPlot
- NetworkX

Data Exploration

- **Check data type for each column**

  Station names and time values are all given as string. Note to parse the string time value into integers so that I can get the travel times from origin to destination.

- **Get list of all possible MRT station names**

  I find that there are 154 possible stations in this network. This can be useful when I plot my graph and I can double check this value with the number of nodes in my graph to ensure that all my stations are in the network.

Data Cleaning

- **Parse string time into time(s) and get the travel time from origin to destination**

  Simply spilt the time string by ":" and we can get the time in seconds by multiplying the appropriate values to each separated value. I observe that the maximum travel time here is 38736 seconds while the shortest is 20 seconds. Median time is 1332 seconds. I created a new column for this 'travel_time"

- **Check for any self-loops where origin and destination nodes are the same. Eg Kent Ridge - Kent Ridge**

  I want to remove these rows if any, since it does not make sense for a MRT station to have a train line that loops back to itself. After removing 2031 of these rows, I am left with 37218 rows.

Before jumping into data manipulation...

Since there are many rows where station pairs are the same, I have to decide on whether to use mean/median/mode to get the final travel time. I rule out mode since this might not be a unique value and focus on mean or median.

Choosing between mean and median:

Mean	Median
Robust to outliers	Sensitive to outliers
Robust to skewed distributions	Sensitive to skewed distributions

A safe choice would be to use median since we have seen previously that the smallest travel time is 20 seconds and the longest is 38736 seconds. However, I feel it is better to stay flexible on this choice and change as we observe our results

Data Manipulation

➔ Grabbing the 2 shortest available paths (Median Approach)

Using a pivot table, I group the destinations and their respective origins to take the median travel times.

	destination	origin	travel_time
0	Admiralty	Woodlands	313.0
1	Admiralty	Sembawang	413.5
2	Admiralty	Marsiling	502.0
3	Admiralty	Yishun	611.0
4	Admiralty	Kranji	669.0

These values are sorted in ascending order, notice that if we were to take the **first 2 values**, origin stations would be **Woodlands and Sembawang**. A quick check on the actual MRT network shows that this is indeed accurate!

➔ Testing logic on the actual MRT network

Consider the MRT line: Joo Koon - Pioneer - Boon Lay

	destination	origin	travel_time
7780	Pioneer	Boon Lay	286.0
7781	Pioneer	Lakeside	480.0

This is wrong since the 2 MRT stations adjacent to Pioneer should be Joo Koon and Boon Lay. **Let us test with mean travel time instead**

➔ Grabbing the 2 shortest available paths (Mean Approach)

Using a pivot table, I group the destinations and their respective origins to take the mean travel times.

	destination	origin	travel_time
0	Admiralty	Sembawang	451.206897
1	Admiralty	Woodlands	486.059701
2	Admiralty	Marsiling	554.028169
3	Admiralty	Yishun	654.310345
4	Admiralty	Kranji	727.285714

Stations adjacent to Admiralty are still correct

	destination	origin	travel_time
7780	Pioneer	Boon Lay	325.000000
7781	Pioneer	Joo Koon	484.941176

Stations adjacent to Pioneer are now correct. We proceed to assume that the mean travel times give us more confidence in selecting the adjacent stations

➔ Take the top 2 shortest path for each destination station according to the sorted mean travel times.

For each unique destination station, I selected the rows with the 2 smallest possible mean travel times

	destination	origin	travel_time
0	Admiralty	Sembawang	451.206897
1	Admiralty	Woodlands	486.059701
2	Aljunied	Paya Lebar NSEW	311.650000
3	Aljunied	Lavender	449.774194
4	Ang Mo Kio	Yio Chu Kang	378.882353
5	Ang Mo Kio	Bishan CCL	424.021277

Based on my assumptions, the origin stations associated with the 3 destinations shown, should be adjacent to their respective destination.

I counted a total of 153 unique destinations so this means that every station will be considered in at least one path, except for 1 (Ten Mile Junction)

➔ Structuring (origin, destination) paths to represent undirected edges

I remove the order of origin -> destination by adding the 2 stations in a list and sort them in alphabetical order to ensure I get a structured order of station pairs

*See next slide to see how I represented station pairs

➔ Continued....

	destination	origin	travel_time	combined
177	Marsiling	Admiralty	553.492754	Admiralty,Marsiling
0	Admiralty	Sembawang	451.206897	Admiralty,Sembawang
243	Sembawang	Admiralty	363.285714	Admiralty,Sembawang
1	Admiralty	Woodlands	486.059701	Admiralty,Woodlands
294	Woodlands	Admiralty	412.826923	Admiralty,Woodlands

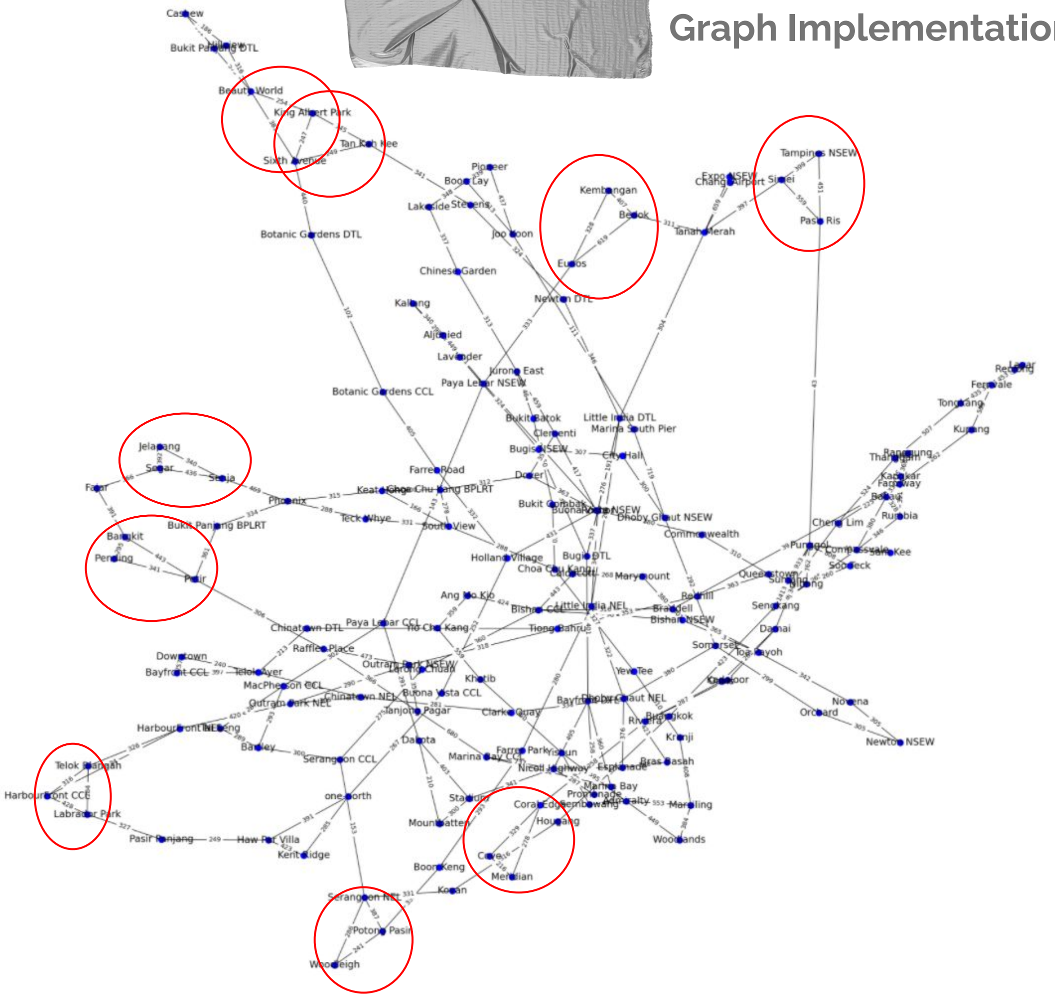
By representing the station pairs like this, I can group by 'combined' values and take the mean travel time. This reduces the 2 directional weighted edges into a single **undirected weighted edge**

➔ Getting ready to plot!

Using a pivot table, I take the mean travel times based on each 'combined' unique values.

By my assumption, we should have an edge between every station since we have found the 2 adjacent stations for every station and have reduced incoming and outgoing edges into an undirected edge

Graph Implementation



Observations:

1. Many cycles were formed containing 3 stations.
2. The circled cycles are the obvious one but I am very sure that are many more in this network.

Thoughts:

1. It does not make sense for a 3-way station loop to be implemented in a real world MRT network.
2. A quick check on the actual MRT network tells me that there are no 3-way cycles present.
3. We need to find a way to add the edges such that these cycles can be avoided

Proposal:

1. Take rows where edge(Node i, Node j) is represented by more > 1 data row. This will give us confidence that the edge is strong and proceed to plot strongly connected components with these nodes
2. Using edges that have not been added in, we will add them in one by one, each time checking for a 3-way cycle and doing cyclic pruning if any..

Why my Proposal works

➔ Strength of connection between 2 stations

	destination	origin	travel_time	combined
0	Admiralty	Sembawang	451.206897	Admiralty,Sembawang
243	Sembawang	Admiralty	363.285714	Admiralty,Sembawang
1	Admiralty	Woodlands	486.059701	Admiralty,Woodlands
294	Woodlands	Admiralty	412.826923	Admiralty,Woodlands
4	Ang Mo Kio	Yio Chu Kang	378.882353	Ang Mo Kio,Yio Chu Kang
299	Yio Chu Kang	Ang Mo Kio	339.777778	Ang Mo Kio,Yio Chu Kang

➔ Compared to....

	destination	origin	travel_time	combined
177	Marsiling	Admiralty	553.492754	Admiralty,Marsiling
302	Yishun	Admiralty	595.115789	Admiralty,Yishun
135	Kallang	Aljunied	340.333333	Aljunied,Kallang
3	Aljunied	Lavender	449.774194	Aljunied,Lavender
2	Aljunied	Paya Lebar NSEW	311.650000	Aljunied,Paya Lebar NSEW

Consider paths with multiple travel times (See **red arrows**). Since there is an incoming and outgoing edge incident on both stations, we can confidently say that there is a **high chance of a connection between the 2.**

Hence, they are adjacent to each other.

These paths are 'one-way', meaning that we cannot fully be certain that those 2 stations are connected in an adjacent manner

Separate data

All duplicated rows (This will contain our **STRONG edges**)

All unique rows (This will contain the **WEAKER edges**)

STEP 1: Form graph using STRONG edges first

*Expected an incomplete graph with many connected components

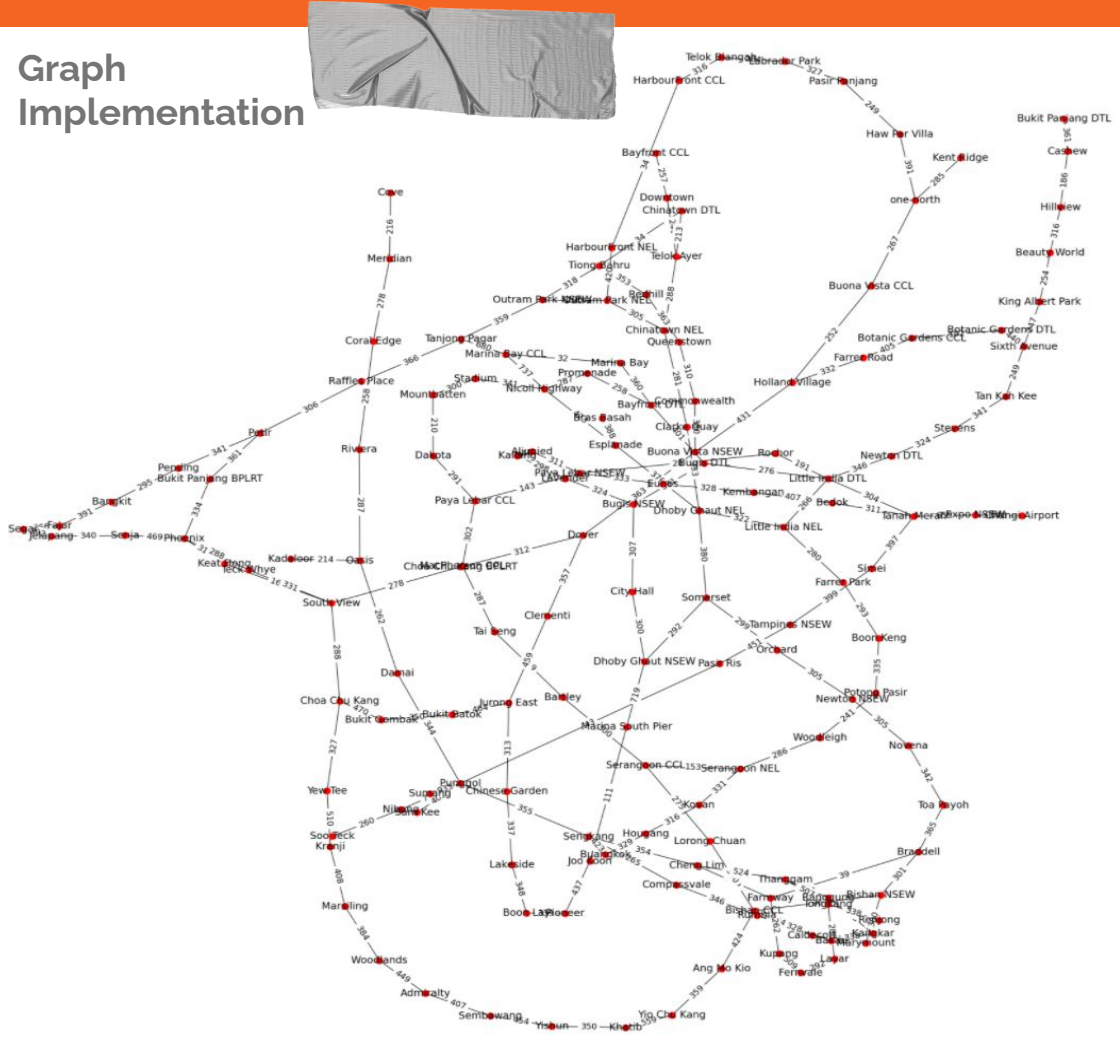
STEP 2a: Iteratively add a weaker edge one by one,

STEP 2b: At each iteration, check for a 3-way cycle and remove the edge with longest travel time in that cycle if any.

I am trying to model a customised cyclic pruning algorithm for this network

Refer to next page for graph after Steps 2a and 2b

Graph Implementation



Observations:

1. Significant decrease in 3-way cycles
2. When compared to the actual MRT network, most of the station orders are accurate except for some.

Thoughts:

1. Although there is a decrease in number of cycles, we cannot be sure that cyclic pruning did not remove important nodes in the process..
2. A quick check on the actual MRT network tells me that, in fact, some correct edges were replaced with incorrect ones
3. The graph could be inaccurate when compared to the actual network for a couple of reasons:

- My assumption that time taken to travel 1 station away will be shorter than that of 2 stations. However, I noticed that in some cases, this assumption does not hold.
- By taking the 2 shortest mean travel does not fully guarantee that we will get the 2 adjacent stations. We might be taking the first adjacent station then the next station following the first.

Conclusions

➔ **About the data**

Using the appropriate aggregation techniques and making sensible assumptions, we were able to approximate the adjacent stations of a given station just by travelling time.

➔ **Possibilities with Network Graph**

We can derive the betweenness centrality for each node and assess their importance in connecting the whole MRT network. With such a graph, we can run algorithms to the representation to quantify the impact of changes to the transport system eg. removal/addition of MRT stations, optimal position to increase connectedness of the transport network. One widely used application would be recommending the shortest possible path to a user if he wants to travel somewhere!

THANK YOU!