Connect Singapore

Huang Qirui
Chua Rui Hong
Ng See Jay

1. Introduction

The public transport system provides access to virtually all places in Singapore at an affordable price. It undoubtedly plays an essential role in the lives of many Singaporeans. In 2016, 67% of journeys during the morning peak hour in Singapore were made using public transport (Abdullah & Tan, 2018). In 2018, a total of 7.54 million journeys were made using bus or train per day, marking the 14th year of consecutive increase in public transport ridership (Tan, 2019).

However, not all places are equally well-connected. Despite the importance of the public transport system, there is a lack of studies analysing the varying connectivity of places in Singapore through public transport. As such, this project aims to do so by using using graph theory to create a model of the public transport system, then analysing its efficiency.

We also aim to investigate the improvements to connectivity that will be brought about through the building of new MRT lines and stations, up to the year 2030, as detailed in the LTA's Land Transport Master Plan (LTMP) 2013.

2. Terminology

Term	Explanation
Train	Mass Rapid Transit (MRT) and Light Rapid Transit (LRT)
Node	A bus stop, bus interchange, or train station in Singapore.
Optimal connection/route	The fastest route from one node to another, through the optimal usage of MRT, LRT, and public buses (i.e. Public Transport services, excluding taxis).
Subzone	A geographical subzone in Singapore, as defined by the Urban Redevelopment Authority (URA), or referring to the set of all nodes found in that subzone.
Place	Node or Subzone.

3. Research Questions

The following are our research questions (RQs):

1. What is the optimal route between every possible pair of nodes?
2. What is the connectivity of each node and subzone,
   a. by GC (Time)?
   b. by GC (Speed Index)?
3. How and to what extent will the new MRT lines and stations that will be built up to 2030 improve connectivity?

For research question 2a, nodes and subzones will be measured by their **General Connectivity (Time)/GC (Time).** GC (Time) quantifies how convenient it is to travel from a place, based on how much time it takes to travel from that place to other places. The formulae for GC (Time) are as follows:

$$\textit{General Connectivity (Time) of a node} = \frac{\textit{Sum of time taken to travel to every other node}}{\textit{Total number of nodes} - 1}$$

$$\textit{General Connectivity (Time) of a subzone} = \frac{\textit{Sum of the GC (Time) of the constituent nodes of the subzone}}{\textit{Number of nodes in the subzone}}$$

For research question 2b, nodes and subzones will be measured by their **General Connectivity by Speed Index/GC (SI)**. GC (SI) quantifies how effective the public transport system is in servicing places, based on the time efficiency of travelling from those places. GC (SI) does not take into account the geographic location of each node. Formula for the calculation of GC (SI):

$$\textit{General Connectivity (Speed Index) of a node} = \frac{\textit{Real Speed}}{\textit{Expected Speed}}$$

$$\textit{General Connectivity (Speed Index) of a subzone} = \frac{\textit{Sum of the GC (SI) of the constituent nodes of a subzone}}{\textit{Number of nodes in the subzone}}$$

Below is the formula for the calculation of speed of travel between 2 nodes:

$$\textit{Speed of travel between 2 nodes} = \frac{\textit{Straight line distance}}{\textit{Time taken}}$$

Formula of Real Speed is given below:

$$\textit{Real Speed} = \frac{\textit{Sum of speed of travel to every other node}}{\textit{Total number of nodes}}$$

The calculation of Expected Speed involves the variable of Distance Index, whose formula is given below:

$$\textit{Distance Index} = \frac{\textit{Sum of straight line distance to every other node}}{\textit{Total number of nodes}}$$

Expected Speed is then derived from a node's Distance Index using the best-fit line of the graph of Real Speed against Distance Index.

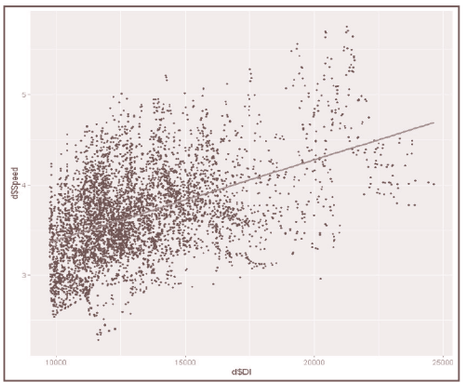


Figure 1. Graph of Real Speed against Distance Index. Made with R.

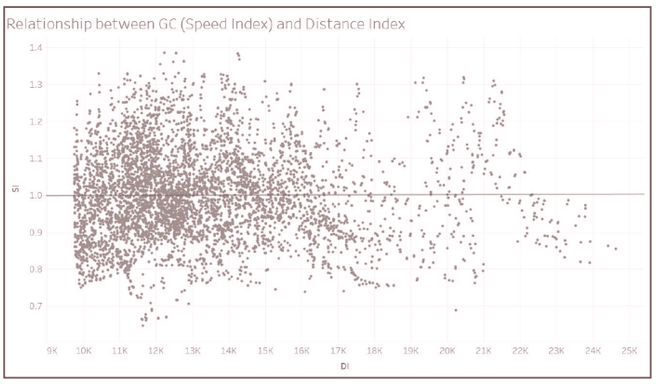


Figure 2. Relationship between GC (SI) and Distance Index. Made with Tableau.

There is no relationship observed between GC (SI) and Distance Index, our measure of node geographic centrality, indicating that we have successfully removed factors related to a node's geographic centrality when determining the efficiency of public transport services through speed index.

4. Literature Review

Dijkstra's Algorithm is a shortest path algorithm that is guaranteed to find the fastest route from a given node to other nodes. Ibrahim (2007) concluded that Dijkstra's Algorithm is a "useful graph theoretic mechanism for optimisation process of network connectivity".

In 2019, the United Kingdom National Infrastructure Commission published a discussion paper on transport connectivity in the UK. They measured urban and inter-urban connectivity, through public transport, private cars, and both combined, mainly at peak hours but also considering off-peak hours.

In 2018, the Land Transport Authority (LTA) published the Land Transport Master Plan (LTMP) 2040. Among other goals, the LTA stated in the LTMP 2040 that connectivity will be improved such that 8 in 10 households will be in a 10 minute walk from a MRT Station, and 9 in 10 households will only need 45 minutes to travel to the Central Business District. Various methods, including the expansion of the rail network, would be employed.

5. Methodology

5.1. *Data Sources*

Data used in our project was primarily gathered from the following websites:

1. Land Transport Datamall (https://www.mytransport.sg/content/mytransport/home/dataMall.html, multiple Application Programming Interfaces (APIs) were used)
2. Land Transport Authority Website (https://www.lta.gov.sg/content/ltaweb/en/public-transport.html)
3. Data.world (Downloaded from https://data.world/hxchua/train-stations-in-singapore and https://data.world/hxchua/mrtfaretime)
4. Google Maps Platform Distance Matrix API (Service accessed from https://cloud.google.com/maps-platform/)
5. Data.gov.sg (Downloaded from https://data.gov.sg/dataset/master-plan-2014-subzone-boundary-no-sea)

5.1.1 *Bus and Train Routes, Frequencies, Station Locations and Train Travel Times*

Data regarding bus service routes, bus service frequencies, and the locations of all bus stops were gathered from Land Transport Datamall. As bus service frequencies were expressed in ranges, we took the middle of the range to be the expected waiting time.

Data from 8 different time periods were collected for bus service frequencies and travel times were collected to better gauge the actual time needed to travel from one place to another as some places such as the Central Business District are heavily affected by peak periods. The 8 different times collected were:

1. Wednesday 0730 (Weekday morning peak)
2. Wednesday 1530 (Weekday afternoon off-peak)
3. Wednesday 1800 (Weekday evening peak)
4. Wednesday 2200 (Weekday night off-peak)
5. Saturday 0730 (Weekend morning peak)
6. Saturday 1530 (Weekend afternoon off-peak)
7. Saturday 1800 (Weekend evening peak)
8. Saturday 2200 (Weekend night off-peak)

These timings coincide with the 4 time periods for which LTA provided data for the frequencies of buses (0630 - 0830, 0831 - 1659, 1700 - 1900, After 1900).We noted that buses travelled slower during peak periods, but came at higher frequencies.

The locations of train stations and a matrix of the travel times between each pair of train stations were downloaded from data.world. The uploader of these datasets had obtained them from TransitLink Electronic Guide and Land Transport Datamall & OneMap Singapore respectively. As no data were available on train service frequencies, we used an estimation of once every 2 minutes during peak periods and once every 4.5 minutes during non-peak periods. To reflect the time spent walking to the train platform after entering a train, we added 0.5 minutes to the waiting time to get 2.5 minutes and 5 minutes respectively.

There were 5023 bus stops and 157 train stations, making a total of 5180 nodes.

5.1.2 ***Walking Routes and Times***

All possible walking routes between pairs of nodes not more than 500m apart were generated, using the coordinates of the bus stops and train stations we had gathered. The haversine formula was used to find the distance between pairs of nodes. The walking speed was set to be 1.11m/s, travelling in a straight line.

5.1.3 ***Bus Travel Times***

For bus travel times, we compiled a list of pairs of bus stops (A, B), where one can travel from bus stop A to bus stop B by taking a bus for 1 stop, and the corresponding bus services for each connection. To travel between 2 bus stops, we found that a bus virtually always takes the same route as the one recommended by Google Maps for a car to drive. Because each pair of directly connected bus stops were usually quite close to one another, there was usually a route between them that was undoubtedly the best, usually using a large road where vehicles can drive quickly. For cases where A and B were far apart, chances were that it was because the bus went on an expressway, which would also be the quickest route for a car to take. Hence, Google Maps Platform's Distance Matrix API was used to find the travel time for a car between all directly-connected pairs of bus stops (A, B). Even though Google Maps API has a mode using public transit, we did not use it as it factored in bus waiting times which would cause the time taken to be inaccurate.

As Google Maps API takes cars to travel faster than public buses, a multiplier of 1.316 was applied to the travel times obtained. This multiplier was obtained by averaging the ratios of

bus travel times to car travel times for 7 pairs of directly-connected bus stops that were at least 5km apart.

Furthermore, there was an addition of 20 seconds whenever a bus stops at a bus stop in order to take into account the increase in time of a bus journey caused by slowing down and stopping. For trains, a waiting time of 150 seconds was added for peak hours and 300 seconds for non-peak hours.

5.1.4 ***Subzone and Town Boundaries***

From Data.gov.sg, we downloaded a KML file showing the boundaries of each subzone, as well as which town each subzone was in. Each subzone is made up of one or more polygons. From the KML file, we extracted the sets of the coordinates of the vertices for each of the polygons for each of the 323 subzones.

Not all of the subzones were found to contain nodes. Some, such as Jurong Island and Bukom, were found to contain 0 nodes. 308 subzones currently have already have at least 1 node. Marina East subzone will get Founders' Memorial MRT station on the TEL as its first node, bringing to total number of subzones in RQ3 to 309.

5.1.5 ***New MRT Lines and Stations***

A list of new train stations to be built by 2030 for RQ3 was created using data from the Land Transport Authority's website. The locations of the new train stations were obtained from location maps of each station published by the LTA.

The corresponding connections between these train stations and with existing train stations were obtained from the train system map from the LTA. The speed of travel between each new pair of directly connected train stations was estimated to be 12m/s, travelling in a straight line.

This speed was based on our calculations which put the average travel speed on the most recently constructed Downtown Line at 11.8m/s. The estimated speed is slightly higher as the newer trains are expected to be faster.

5.2 RQ1: ***What is the optimal route between every possible pair of nodes?***

First, we collated data on bus and train services and generated walking routes between nodes not more than 500m apart. Then, we implement a modified Dijkstra's algorithm in C++ (program in Appendix A).This allowed us to take into account the bus or train service currently being used at each step, so that waiting time when changing transport services could be accounted for.

This is an overview of how Dijkstra's algorithm works:

1. Maintain a list of nodes to visit next, sorted ascendingly by total time taken.

Table 1. An example of what this list may look like while the program is executing.

Current node	75009	84599	92081	83361	…
Current time taken/s	1281	1284	1284	1285	…
Current service	10(1)	WALK	196e(2)	42(1)	…

For bus services, the number in the bracket indicates the direction.

2. Maintain a table storing the lowest possible time to travel from the starting node to each node for each previous mode of transport used. This table is initialised with all the times as infinity.

Table 2. An example of what this table might look like while the program is executing. The best routes may have not yet been found, so the current minimum times may be greater than the true minimum times.

Destination	1012				1013		…	1570000	
Last used Service	WALK	12(2)	61(1)	80(2)	WALK	960(2)		WALK	TRAIN
Minimum time/s for this service	928	873	1482	873	∞	505	…	13922	2349
Overall minimum time/s	873				505			2349	

While we used the original bus stop codes set by LTA for bus stops, we assigned the train stations codes (10000, 20000 … 1570000) so that they would be integers instead of strings containing characters (etc. DT8, EW23). For example, node 1570000 corresponds to Yishun MRT Station.

3. Insert the starting node into the **list** of nodes to visit next, as the first node.

4. While there are still nodes in the list, get the first node in the list and set it to be the current node. Remove it from the list.
5. If the sum of the current time and the time taken to travel to an adjacent node is lower than the current minimum time to travel to that node as found in the table, then a better, faster route to that node has just been found. Hence, update the current minimum time for that node and insert that node into the list, with its updated time as the current time taken.
    a. In this step, we modified Dijkstra's Algorithm to check whether there is a change of transport service, such as changing from one service to another, and factored in waiting time when applicable.
6. Repeat step 3 until the list becomes empty, which means that no further updates were made and the fastest route from the starting node to every other node has been found.

To ensure the reliability of results, we also modified Dijkstra's Algorithm to store and output the routes taken, so that we could easily validate the accuracy of the routes. Afterward, we generated the optimal route between every possible pair of nodes.

5.3 RQ2: ***What is the connectivity of each node and subzone by GC (Time) and GC (SI)?***

We classified each node into its subzone using the even-odd algorithm, using a C++ program (program in Appendix B), as this information had not been provided by the LTA. Using the optimal routes generated in RQ1, we calculated the GC (Time) and GC (SI) of every node and used these values to find the GC (Time) and GC (SI) of every subzone.

5.3.1 ***Even-Odd Algorithm***

To determine the subzone each node was in, we employed the even-odd rule. Each subzone was represented as a polygon or set of polygons (for subzones with non-contiguous parts) on a plane, based on the coordinates of its vertices. Each node was represented as a point on that plane based on its longitude and latitude. Through the use of the even-odd algorithm, we successfully classified all 5180 nodes into their respective subzones.

This is an overview of how our Even-Odd algorithm works:

1. Represent a node as a point.
2. Represent a subzone as a polygon or set of polygons.
3. For each polygon of the subzone, draw a ray from the point to infinity in any direction. This direction must be the same every time. For simplicity, we set the

direction to be the positive x-direction, or the direction with increasing longitude and constant latitude.

   a. Check if the point lies on an edge or vertex of the polygon, which would result in a special case. We did not find any such cases.

4. Count the number of segments from the given polygon that the ray crosses. If the number is **odd**, the point is **inside** that polygon, and hence inside that subzone. If the number is **even**, the point is **outside**.
5. Repeat steps 2-4 for every subzone until the subzone where the point is in is found.
6. Repeat steps 1-5 for all nodes.

5.4 RQ3: *How and to what extent will the new MRT lines and stations that will be built up to 2030 improve connectivity?*

We gathered information about the upcoming MRT lines and stations, then generated another set of results, similar to in RQ1 and RQ2. As our goal was to measure the concrete improvements to connectivity brought about by the new MRT lines and stations, we investigated only GC (Time) for RQ3.

6. Results

6.1 RQ1: *What is the optimal route between every possible pair of nodes?*

We successfully generated the optimal route between every pair of nodes. Multiple transport modes were able to be used in the same route and the program would choose to walk to a nearby node within 500m when that would result in a shorter journey overall.

Some examples of the optimal routes generated by Dijkstra's Algorithm are as follows:

1. Aft Lim Chu Kang Lane 8 (34049) to Changi Village Ter (99009) (2h 26min 12s)

   Step 1: Take Bus 975 for 35 stops to Blk 414 bus stop

   Step 2: Walk to Choa Chu Kang MRT Station

   Step 3: Take the MRT to Tanjong Pagar MRT Station

   Step 4: Walk to Opp MAS Bldg bus stop

   Step 5: Take Bus 661 for 1 stop to Blk 149A bus stop

   Comment: Bus 661 is a city-direct bus and the road distance between Opp MAS Bldg and Blk 149A is 20.8km. Because it is inconvenient to travel to Blk 149A from the nearest EWL station, Tampines, it turns out to be faster to take the express bus directly from Tanjong Pagar.

   Step 6: Take Bus 19 for 10 stops to Opp Maranatha B-P Ch bus stop

Step 7: Walk to Changi Village Ter.

2. Bishan MRT Station to Aft AMK Ind Pk 2 (66451) (28min 39s)

   Step 1: Take the MRT to Yio Chu Kang MRT Station

   Step 2: Walk to Yio Chu Kang Int

   Step 3: Take Bus 72 for 8 stops to Aft AMK Ind Pk 2

6.2 RQ2: ***What is the connectivity of each node and subzone by GC (Time) and GC (SI)?***

Using the even-odd algorithm, we classified all 5180 nodes into their subzones.

6.2.1 RQ2a: ***General Connectivity by Time***

After obtaining the GC (Time) value of every node, we discovered that the distribution of GC (Time) was slightly skewed right.

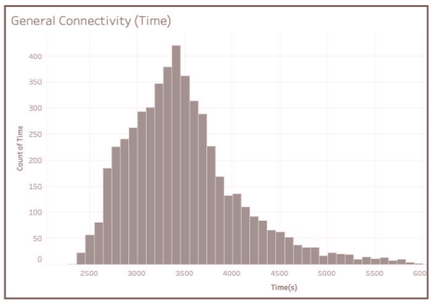


Figure 3. Distribution of GC (Time). Made with Tableau.

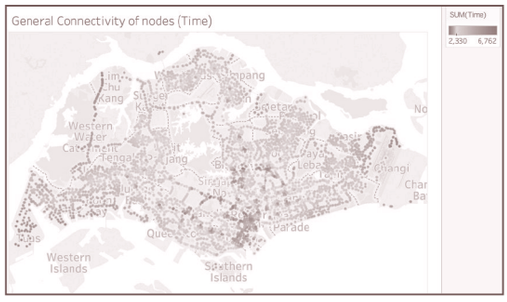


Figure 4. Heatmap of GC (Time) for each node. Dark red indicates the lowest or best GC (Time) and dark blue indicates the highest or worst. Made with Tableau.

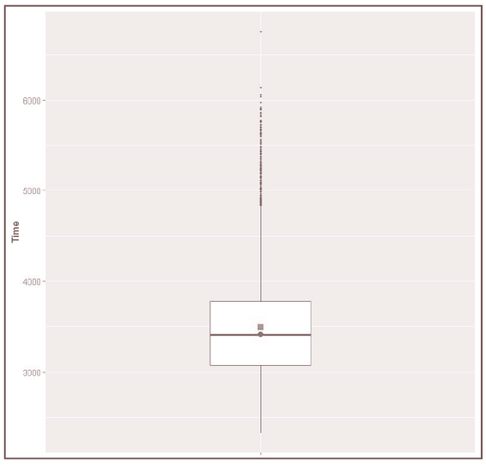


Figure 5. Boxplot of GC (Time). Red square indicates mean (3469s) and blue circle indicates median (3411s). Made with R.

The best, or minimum, GC (Time) was 2330s, and the worst was 6751s, with a range of 4421s. The mean GC (Time) was 3497s and the median was 3411s. The first and third quartile values for GC (Time) were 3076s and 3777s respectively, with an interquartile range

of 702s. The standard deviation of the distribution was 605s.

Two factors were observed to have a large bearing on a node's GC (Time). Firstly, nodes in central locations tended to have higher GC (Time) values than less central nodes. Secondly, MRT stations and nodes near them tended to have higher GC (Time) values than nodes far from MRT stations. Below are the top and bottom nodes.

Table 3. Top 5 and Bottom 5 nodes by General Connectivity (Time)

Node	GC (Time)/s	Rank	Subzone
Bishan MRT Station	2330.41	1	Bishan East
Bishan Stn *(bus stop)*	2347.03	2	Bishan East
Botanic Gardens MRT Station	2363.61	3	Tyersall
Opp Bishan Stn *(bus stop)*	2379.34	4	Marymount
Newton MRT Station	2380.04	5	Newton Circus
…	…	…	…
Bef Tuas Sth Ave 14	5972.14	5176	Tuas View Extension
Prologis	6037.53	5177	Changi Airport
See Hup Seng	6055.96	5178	Tuas View
Halliburton	6133.63	5179	Tuas View
Larkin Ter	6750.65	5180	Johor Bahru

The average GC (Time) for a train station was 2870s. This was 626 seconds faster than the average GC (Time) for all nodes, which is 3496s, thus train stations were found to have a 17.9% shorter mean travel time to other nodes.

Similarly, subzones with low GC (Time) value were located centrally, as can be seen from the map below. Below are the top and bottom subzones.

Table 4. Top 5 and Bottom 5 subzones by General Connectivity (Time)

Subzone	Planning Area (Town)	GC (Time)/s	Rank
Newton Circus	Newton	2482.65	1
Mackenzie	Rochor	2502.19	2
Dhoby Ghaut	Museum	2524.09	3
Raffles Place	Downtown Core	2587.71	4
Farrer Park	Rochor	2589.99	5

…	…	…	…
Changi Point	Changi	5146.05	303
Lim Chu Kang	Lim Chu Kang	5291.83	304
Tuas View	Tuas	5308.61	305
Johor	Johor	5505.98	306
Tuas View Extension	Tuas	5594.07	307

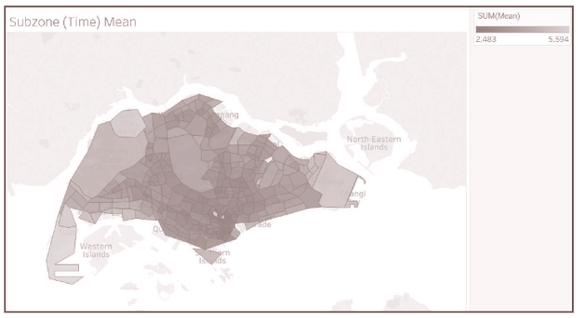
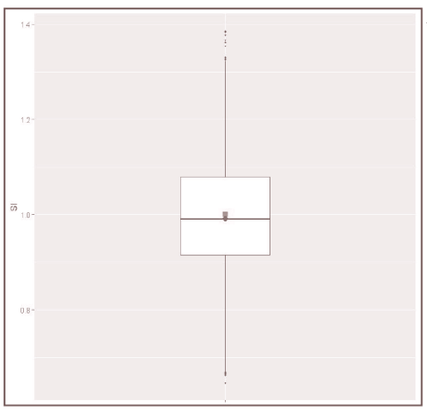


Figure 6. General Connectivity (Time) of Subzones. Made with Tableau.

6.2.2 RQ2b: *General Connectivity by Speed Index*

There was a normal distribution of GC (SI) across nodes. The speed indices ranged from 0.6466 to 1.3858, having a range of 0.7392. The mean was 0.9999 and the median was 0.9907. The mean and median were close to 1 since speed indices were derived from a best-fit line. The first and third quartile values were 0.9150 and 1.0787 respectively, with an interquartile range of 0.1637. The standard deviation was 0.1206. The yellow dots on the heatmap, representing nodes with high GC (SI) values, coincided with Singapore's MRT Lines. This made sense as places with MRT lines were more effectively served and connected by public transport as MRT travels faster compared to buses. Train stations had a mean GC (SI) of 1.183 while the overall mean was 1.000.



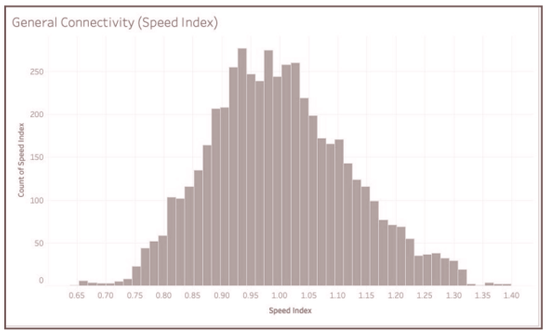


Figure 7. Distribution of GC (SI). Made with Tableau.

Figure 8. Boxplot of GC (Speed Index). Made with R.

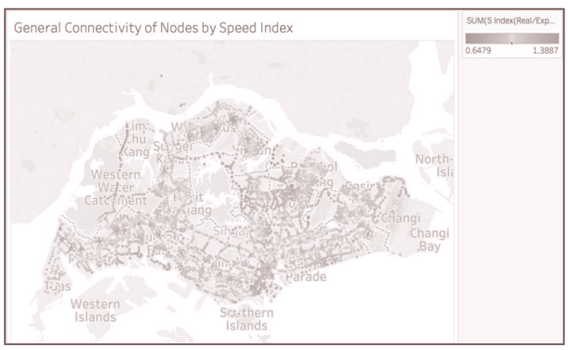


Figure 9. Heatmap of GC (SI). Yellow indicates the best GC (SI) and purple indicates the worst. Made with Tableau.

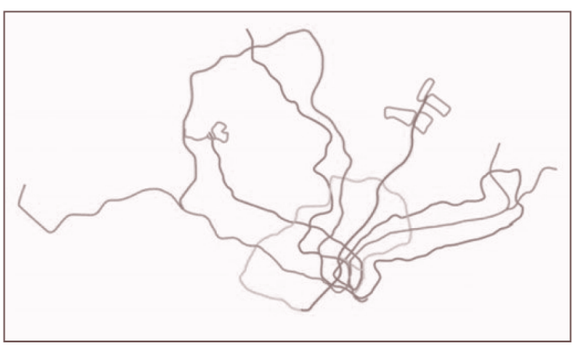


Figure 10. Geographical MRT Map. Brown line represents the Thomson-East Coast Line which has yet to be completed. Thus, please ignore it for now.

On the other hand, the centrality of a node had little to no bearing on its GC (SI). Below are the top and bottom nodes in terms of GC (SI).

Table 5. Top 5 and Bottom 5 nodes by General Connectivity (Speed Index)

Node	GC (Speed Index)	Rank	Subzone
Outram Park MRT Station	1.3858	1	Singapore General Hospital
Tanjong Pagar MRT Station	1.3850	2	Tanjong Pagar
Sembawang MRT Station	1.3835	3	Sembawang Central
Sembawang Stn *(bus stop)*	1.3773	4	Sembawang Central
Opp Sembawang Stn *(bus stop)*	1.3673	5	Sembawang Central

…	…	…	…
Aft West Camp Rd	0.6663	5176	Seletar Aerospace Park
St Aero	0.6646	5177	Airport Road
Air Force Sch	0.6631	5178	Airport Road
Bef West Camp Rd	0.6616	5179	Seletar Aerospace Park
Bef Rsaf Roundabout	0.6466	5180	Airport Road

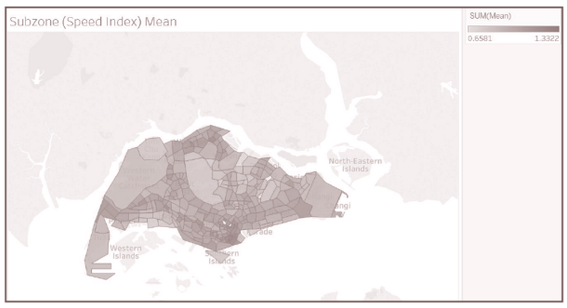


Figure 11. GC (SI) of Subzones. Made with Tableau.

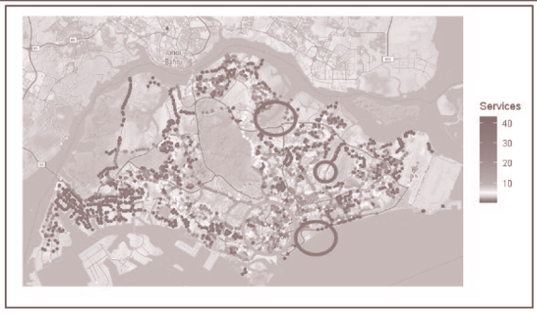


Figure 12. Number of bus services for each bus stop. Orange and blue circles represent the locations of top and bottom 3 subzones in terms of GC (SI) respectively. Made with ggmap library in R.

Availability of bus services affected the GC (SI) values of subzones as well. As can be seen from the maps above, subzones with low GC (SI) were located in areas with few bus services. Below are the top and bottom subzones.

Table 6. Top 5 and Bottom 5 subzones by General Connectivity (Speed Index)

Subzone	Planning Area (Town)	GC (SI)	Rank
Tanjong Pagar	Downtown Core	1.3322	1
Raffles Place	Downtown Core	1.2928	2
Central Subzone	Downtown Core	1.2702	3
China Square	Outram	1.2685	4
Dhoby Ghaut	Museum	1.2445	5
…	…	…	…
Johor	Johor	0.7812	303
Jurong Port	Jurong East	0.7707	304
Marina East (MP)	Marine Parade	0.7433	305

Seletar Aerospace Park	Seletar	0.6949	306
Airport Road	Paya Lebar	0.6581	307

6.2.3: *Relationship between GC (Time) and GC (SI)*

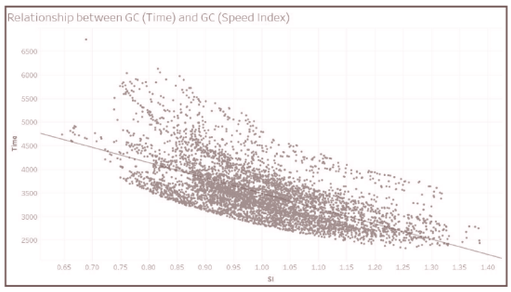


Figure 13. Relationship between GC (Time) and GC (SI). Made with Tableau.

As observed from the graph, there was a clear relation between general connectivity by Time and general connectivity by Speed Index, with a p-Value at 2.2×10^{-16} and an adjusted R^2 linear regression line fit of 41.83%. Hence, we can confirm that the efficiency with which the public transport system services a node does indeed affect the connectedness of a node.

6.3 RQ3: *How and to what extent will the new MRT lines and stations that will be built up to 2030 improve connectivity?*

For all nodes, at least 820 of 5240 routes to other nodes utilised connections passing through at least 1 new node, with the maximum being 5176. This shows that the new MRT lines brought about some improvement on the connectivity of every node.

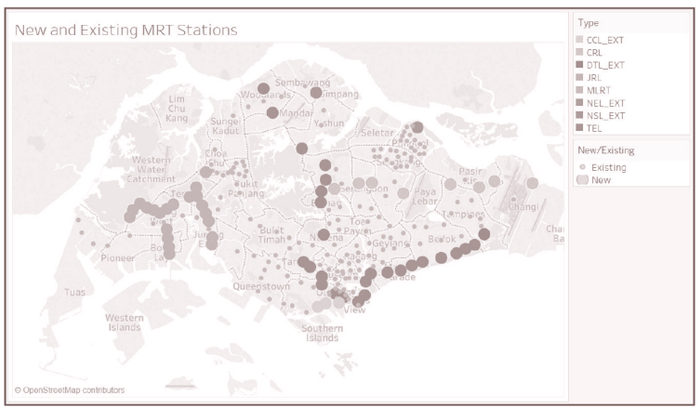


Figure 14. The new MRT lines and stations added in RQ3. Grey dots indicate existing MRT and LRT stations. Made with Tableau.

6.3.1: *New MRT Stations*

The new MRT lines added include the Thomson-East Coast Line with 32 stations, the Jurong Region Line with 24 stations, the first phase of the Cross-Island Line with 12 stations, 3 additional stations adding on to the Circle Line, 2 Downtown Line extensions, Punggol North MRT station belonging to the North-East Line and Canberra Station belonging to the North-South Line. This raised the number of nodes from 5180 to 5241.

6.3.2: *Improvements in GC (Time)*

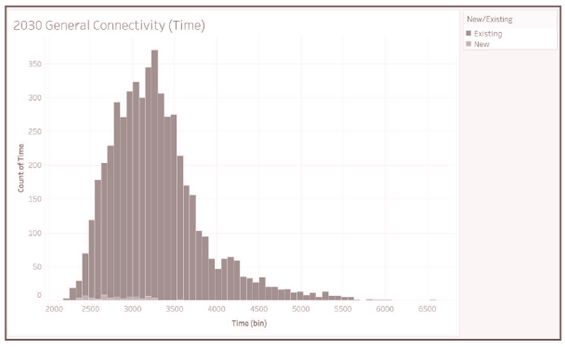


Figure 15. Distribution of the new General Connectivity (Time) values. Blue indicates nodes that already exist and orange indicates the 61 new nodes. Made with Tableau.

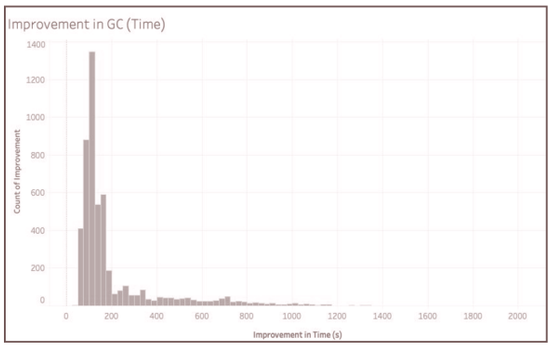


Figure 16. Distribution of improvement in GC (Time). Made with Tableau.

The distribution of the new GC (Time) graph is still skewed right. The new average overall GC (Time) improved from 3496s to 3280s, with an improvement of 216s or 6.17%. The standard deviation of GC (Time) decreased from 605s to 551s, a notable drop of 54s or 8.94%. This indicates that the gap between poorly-served and well-served subzones decreased. The new top and bottom nodes and subzones can be found in the following tables.

Table 7. Top 5 and Bottom 5 nodes by RQ3 General Connectivity (Time)

Node	GC (Time)/s	GC (Time) Improvement/s	Rank	Subzone
Bishan MRT Station	2228.55	101.86	1	Bishan East
Caldecott MRT Station	2240.34	199.86	2	Toa Payoh West
Bishan Stn *(bus stop)*	2245.3	101.73	3	Bishan East
Botanic Gardens MRT Station	2257.43	106.18	4	Tyersall
Outram Park MRT Station	2266.42	176.63	5	Singapore General Hospital
…	…		…	…
Bef Tuas Sth Ave 6	5792.67	109.09	5237	Tuas View
Bef Tuas Sth Ave 14	5862.94	109.20	5238	Tuas View Extension
See Hup Seng	5946.88	109.08	5239	Tuas View
Halliburton	6024.43	109.20	5240	Tuas View Extension
Larkin Ter	6589.57	161.08	5241	Johor

Table 8. Top 5 and Bottom 5 subzones by RQ3 General Connectivity (Time)

Subzone	Planning Area (Town)	GC (Time)/s	GC (Time) Improvement/s	Rank
Newton Circus	Newton	2379.123	103.52	1
Mackenzie	Rochor	2406.83	95.36	2
Maxwell	Downtown Core	2421.28	382.91	3
Dhoby Ghaut	Museum	2437.383	86.71	4
Tanjong Pagar	Downtown Core	2479.55	111.21	5
…	…	…	…	…
Changi Bay	Changi Bay	4811.985	204.69	303
Lim Chu Kang	Lim Chu Kang	4993.898	297.93	304
Tuas View	Tuas	5199.479	109.13	305
Johor	Johor	5342.833	163.15	306
Tuas View Extension	Tuas	5484.859	109.21	307

In addition, the following statistics were found for the new GC (Time) values, counting all 5241 nodes. The best, or minimum, GC (Time) was 2229s, and the worst was 6590s, with a range of 4361s. The mean GC (Time) was 3280s and the median was 3210s. The first and third quartile values for GC (Time) were 2895s and 3531s respectively, with an interquartile range of 636s. The standard deviation of the distribution was 551s.

6.3.2a: ***Example of routes with best improvement***

The current route from Aft Defu Ave 2 to Aft Changi Ferry Road takes 1h 32 min 1s while the new route making use of the Cross Island Line would take only 29 min 9s.

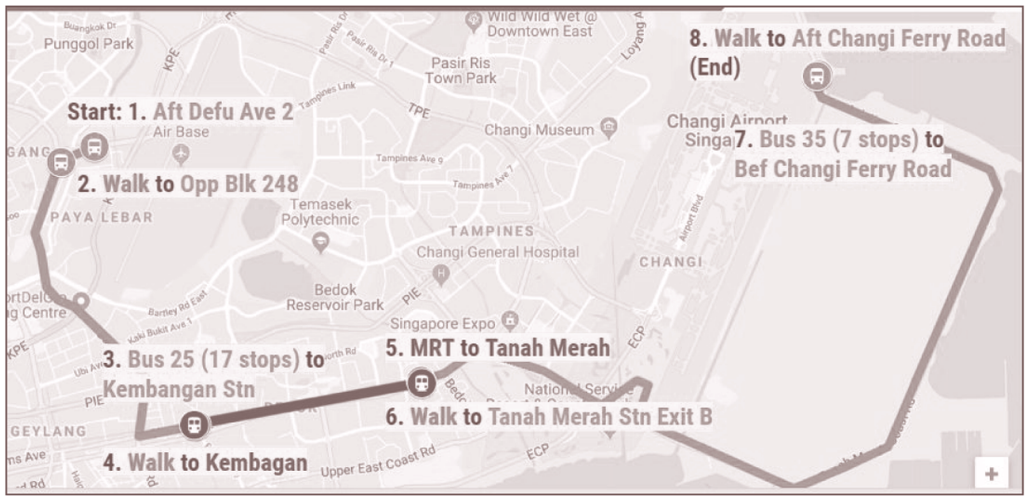


Figure 17. The current best route from Aft Defu Ave 2 to Aft Changi Ferry Road, taking 1h 32min 1s. Made with Google My Maps and Google Slides.

Figure 18. The new best route, taking 29min 9s, with a time improvement of 1h 2min 52s. Made with Google My Maps and Google Slides.

6.3.2b: ***Improvement of certain areas***

Figure 19. Heatmap showing the improvement of GC (Time) from RQ2 to RQ3 for each node. Made with Tableau.

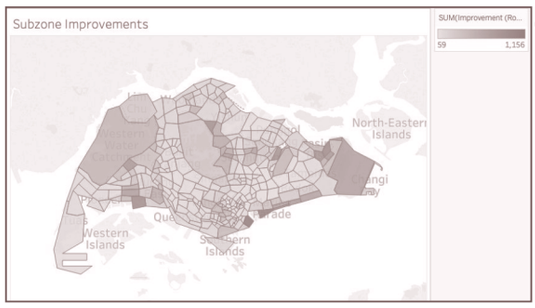


Figure 20. Heatmap showing the improvement of GC (Time) from RQ2 to RQ3 for each subzone. Made with Tableau.

The majority of subzones and nodes which received great improvements coincided with the location of the new MRT lines, indicating that the introduction of new MRT lines would greatly improve connectivity of a node or subzone. These areas are highlighted in the map below.

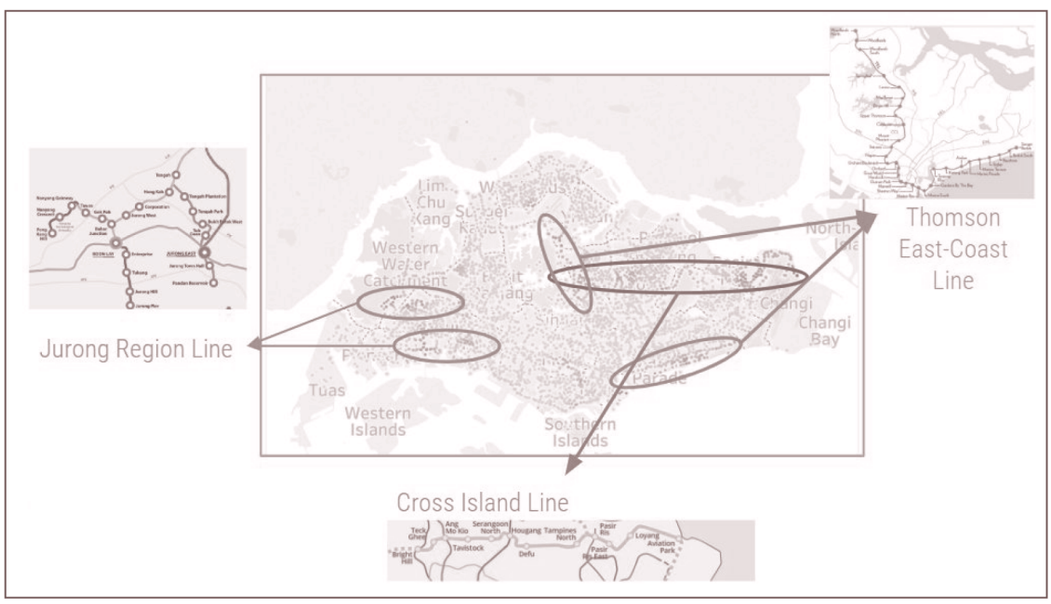


Figure 21. Explanations for some places with bigger than average improvements in GC (Time). Made with Tableau and Google Slides.

Specific improvements in General Connectivity (Time) for nodes and subzones can be found in the tables below.

Table 9. Top 5 and Bottom 5 nodes by improvements in General Connectivity (Time)

Node	GC (Time) Improvement/s	New GC (Time)/s	Rank	Subzone
Bef Changi Ferry Rd	2006.94	3845.4	1	CHANGI AIRPORT

Aft Changi Ferry Rd	1978.08	3859.26	2	CHANGI AIRPORT
Prologis	1869.04	4168.49	3	CHANGI AIRPORT
Dhl	1598.04	4299.82	4	CHANGI AIRPORT
Jurong Island Checkpt	1563.64	3482.41	5	JURONG PORT
…	…	…	…	…
Blk 311	56.65	3112.69	5176	KAMPONG UBI
Blk 322	56.6	3216.75	5177	KAMPONG UBI
Blk 319A	30.27	3405.84	5178	ANCHORVALE
Blk 306 Cp	27.14	3518.48	5179	ANCHORVALE
Blk 313B	26.02	3530.19	5180	ANCHORVALE

Table 10. Top 5 and Bottom 5 subzones by improvements in General Connectivity (Time)

Subzone	Planning Area (Town)	GC (Time) Improvement/s	Rank
Marina South	Marina South	1155.75	1
Marina East (Mp)	Marine Parade	1057.11	2
Loyang West	Pasir Ris	948.4	3
Jurong Port	Jurong East	945.48	4
Nee Soon	Yishun	904.57	5
…	…	…	…
Geylang Bahru	Kallang	66.6	303
Macpherson	Geylang	64.89	304
Kampong Ubi	Geylang	61.96	305
Kaki Bukit	Bedok	61.58	306
Airport Road	Paya Lebar	59.3	307

7. Conclusion

In this project, we found the fastest route between every possible pair of nodes in Singapore, then quantified the varying connectivities of all nodes and subzones, for the current transport

network, and when future MRT lines are added. A limitation of our project was that some of the data used to find the shortest routes between nodes were of limited accuracy, especially walking routes, which were estimated using the straight line distance. Three possible applications of our results would be as follows, especially if the project could be redone with better starting data.

Firstly, the RQ2 and RQ3 results could be used to evaluate the predicted effectiveness of the new MRT lines, and show which areas are the most in need of better transport services. For example, while the GC (Time) of Nee Soon subzone improved by over 15 minutes (905s) to 3155s in RQ3, Seletar's GC (Time) improved by 70s from 4040s to 3970s. If more residences are to be built in Seletar, then transport connections there should first be improved.

Secondly, the RQ1 results could be used by commuters to find the shortest route to their destination. The data for some nodes could be downloaded and referred to offline when pathfinding services such as Google Maps are not available.

Lastly, our results may also be helpful to homebuyers who can take into consideration how well-connected the areas around potential homes are before making a decision.

8. References

Abdullah, Z., & Tan, C. (2018). More opting to travel by public transport: Survey. *The Straits Times*, Retrieved from https://www.straitstimes.com/singapore/transport/more-opting-to-travel-by-public-transport-survey

Chua, H. X. (2019). Train Fare and Travel Time between Stations in Singapore. Retrieved July 28, 2019, from https://data.world/hxchua/mrtfaretime

Chua, H. X. (2019). Train Stations in Singapore. Retrieved July 28, 2019, from https://data.world/hxchua/train-stations-in-singapore

Dynamic Data. (n.d.). Retrieved July 28, 2019, from https://www.mytransport.sg/content/mytransport/home/dataMall.html

Hormann, K., & Agathos, A. (2004). Graph The point in polygon problem for arbitrary polygons. *Computational Geometry, 20*(3), 131-144. https://doi.org/10.1016/S0925-7721(01)00012-8

Ibrahim, A. A. (2007). Graph Algorithms and Shortest Path Problems: A Case of Dijkstra's Algorithm and the Dual Carriage Ways in Sokoto Metropolis. *Trends in Applied Sciences Research,2*(4), 348-353. doi:10.3923/tasr.2007.348.353

Land Transport Authority (n.d.). *LAND TRANSPORT MASTER PLAN (LTMP) 2040*. Retrieved from https://www.lta.gov.sg/content/ltaweb/en/about-lta/what-we-do/ltmp2040.html

Routes & Directions | Google Maps Platform | Google Cloud. (n.d.). Retrieved July 28, 2019, from https://cloud.google.com/maps-platform/routes/

Singapore Department of Statistics (2015). *STATISTICS SINGAPORE - Map of Planning Areas/Subzones in Singapore* [PDF file]. Singapore. Retrieved from https://www.singstat.gov.sg/-/media/files/publications/population/population2015-map1.pdf

Tan, C. (2019). Bus and train ridership up, taxi rides down. *The Straits Times*, Retrieved from https://www.straitstimes.com/singapore/transport/bus-and-train-ridership-up-taxi-rides-down

United Kingdom National Infrastructure Commission (2019). *Transport Connectivity - Discussion paper* [PDF file]. Retrieved from https://www.nic.org.uk/wp-content/uploads/Transport-Connectivity-discussion-paper.pdf

Urban Redevelopment Authority. (2014). Master Plan 2014 Subzone Boundary (No Sea). Retrieved from https://data.gov.sg/dataset/master-plan-2014-subzone-boundary-no-sea

9. Appendix: Programs

Appendix A: Dijkstra's Algorithm Program

```
#define __USE_MINGW_ANSI_STDIO 0
#include <bits/stdc++.h>
#define CLEN 7408
#define WLEN 63458
#define MLEN 157
#define WTLEN 732
using namespace std;
int TWAIT, BWAIT;
streambuf *coutbuf;
string direc;

class comp {
  public:
    bool operator()(pair<pair<int, int>, string> A, pair<pair<int, int>, string> B)
{
        return A.first.second>B.first.second;
    }
};

inline void dijkstra(int START, int k, int REF[10000], vector<pair<int, pair<int,
string> > > v[10000], map<string, int> wtimes) {
  map<string, int> dist[10000];
  priority_queue<pair<pair<int, int>, string>, vector<pair<pair<int, int>, string>
>, comp > pq;
  dist[START][""]=0;
  pq.push(make_pair(make_pair(START,0),""));
  int a, b;
  string s;
  while (!pq.empty()) {
    a=pq.top().first.first;
    b=pq.top().first.second;
    s=pq.top().second;
    pq.pop();
    if(dist[a].find(s)!=dist[a].end()) {
      if (dist[a][s]<b) continue;
    }
    for (pair<int, pair<int, string> > i : v[a]) {
      if (i.first==START) {
        dist[START][i.second.second]=0;
        continue;
      }
      if ((s=="TRAIN")&&(i.second.second=="TRAIN")) continue;
      else if (i.second.second=="WALK") {
        if (dist[i.first].find(i.second.second)!=dist[i.first].end()) {
          if (b+i.second.first<dist[i.first][i.second.second]) {
            dist[i.first][i.second.second]=b+i.second.first;

pq.push(make_pair(make_pair(i.first,dist[i.first][i.second.second]),i.second.second
));
          }
        }
        else {
          dist[i.first][i.second.second]=b+i.second.first;

pq.push(make_pair(make_pair(i.first,dist[i.first][i.second.second]),i.second.second
));
        }
      }
```

```
      else if (i.second.second=="TRAIN") {
        if (dist[i.first].find(i.second.second)!=dist[i.first].end()) {
          if (b+i.second.first+TWAIT<dist[i.first][i.second.second]) {
            dist[i.first][i.second.second]=b+i.second.first+TWAIT;

pq.push(make_pair(make_pair(i.first,dist[i.first][i.second.second]),i.second.second
));
          }
        }
        else {
          dist[i.first][i.second.second]=b+i.second.first+TWAIT;

pq.push(make_pair(make_pair(i.first,dist[i.first][i.second.second]),i.second.second
));
        }
      }
      else {
        if (s!=i.second.second) {
          if (wtimes[i.second.second]<0) continue;
          if (dist[i.first].find(i.second.second)!=dist[i.first].end()) {
            if
(b+i.second.first+wtimes[i.second.second]+BWAIT<dist[i.first][i.second.second]) {

dist[i.first][i.second.second]=b+i.second.first+wtimes[i.second.second]+BWAIT;

pq.push(make_pair(make_pair(i.first,dist[i.first][i.second.second]),i.second.second
));
            }
          }
          else {

dist[i.first][i.second.second]=b+i.second.first+wtimes[i.second.second]+BWAIT;

pq.push(make_pair(make_pair(i.first,dist[i.first][i.second.second]),i.second.second
));
          }
        }
        else {
          if (dist[i.first].find(i.second.second)!=dist[i.first].end()) {
            if (b+i.second.first+BWAIT<dist[i.first][i.second.second]) {
              dist[i.first][i.second.second]=b+i.second.first+BWAIT;

pq.push(make_pair(make_pair(i.first,dist[i.first][i.second.second]),i.second.second
));
            }
          }
          else {
            dist[i.first][i.second.second]=b+i.second.first+BWAIT;

pq.push(make_pair(make_pair(i.first,dist[i.first][i.second.second]),i.second.second
));
          }
        }
      }
    }
    }
  ofstream out("OUT/"+direc+"/"+to_string(REF[START])+".txt");
  cout.rdbuf(out.rdbuf());
  for (int i=0;i<k;i++) {
    cout << REF[START] << " " << REF[i] << " ";
    a=1000000007;
    for (pair<string, int> j:dist[i]) a=min(a, j.second);
    cout << a << "\n";
  }
  cout.rdbuf(coutbuf);
  cout << "OUT/"<<to_string(REF[START])<<".txt\n";
```

```
}

int main() {
  vector<pair<int, pair<int, string> > > v[10000];
  map<string, int> wtimes;
  int REF[10000]={};
  string FILE;
  int TIME;
  cin >> FILE >> TIME >> TWAIT >> direc;
  TWAIT+=30;
  BWAIT=20;
  cout << "READING FILES...\n";
    coutbuf = cout.rdbuf();
  ifstream bustimings(FILE);
    streambuf *cinbuf = cin.rdbuf();
    cin.rdbuf(bustimings.rdbuf());
  int a, b, c, k=0, t[CLEN];
  for (int i=0;i<CLEN;i++) {
    cin >> a >> b >> t[i];
    t[i]=(int)((float)t[i]*1.315986395);
  }
  cout << FILE << "\n";
  ifstream connections("connections.txt");
    cin.rdbuf(connections.rdbuf());
  for (int i=0;i<CLEN;i++) {
    cin >> a >> b >> c;
    if (find(REF, REF+10000, a)>=REF+10000) {
      REF[k]=a;
      k++;
    }
    if (find(REF, REF+10000, b)>=REF+10000) {
      REF[k]=b;
      k++;
    }
    for (int j=0;j<c;j++) {
      string s;
      cin >> s;
      v[find(REF, REF+10000, a)-REF].push_back(make_pair(find(REF, REF+10000,
b)-REF,make_pair(t[i],s)));
    }
  }
  cout << "connections.txt\n";
  ifstream walkingroutes("walkingroutes.txt");
    cin.rdbuf(walkingroutes.rdbuf());
  for (int i=0;i<WLEN;i++) {
    cin >> a >> b >> c;
    if (find(REF, REF+10000, b)>=REF+10000) {
      REF[k]=b;
      k++;
    }
    if (find(REF, REF+10000, c)>=REF+10000) {
      REF[k]=c;
      k++;
    }
    v[find(REF, REF+10000, b)-REF].push_back(make_pair(find(REF, REF+10000,
c)-REF,make_pair(a,"WALK")));
    v[find(REF, REF+10000, c)-REF].push_back(make_pair(find(REF, REF+10000,
b)-REF,make_pair(a,"WALK")));
  }
  cout << "walkingroutes.txt\n";
  ifstream mrtroutes("mrtroutes.txt");
    cin.rdbuf(mrtroutes.rdbuf());
  for (int i=0;i<MLEN*MLEN;i++) {
    cin >> a >> b >> c;
    if (find(REF, REF+10000, a)>=REF+10000) {
      REF[k]=a;
```

```
      k++;
    }
    if (find(REF, REF+10000, b)>=REF+10000) {
      REF[k]=b;
      k++;
    }
    v[find(REF, REF+10000, a)-REF].push_back(make_pair(find(REF, REF+10000,
b)-REF,make_pair(c,"TRAIN")));
  }
  cout << "mrtroutes.txt\n";
  ifstream waitingtimes("waitingtimes.txt");
    cin.rdbuf(waitingtimes.rdbuf());
  for (int i=0;i<WTLEN;i++) {
    float FM[4];
    string SERVICE, SN;
    cin >> SERVICE >> SN >> FM[0] >> FM[1] >> FM[2] >> FM[3];
    if (FM[TIME]<0) wtimes[SERVICE+"("+SN+")"]=-1;
    else wtimes[SERVICE+"("+SN+")"]=60*FM[TIME];
  }
  cout << "waitingtimes.txt\n";
  cout << "OUTPUTTING...\n";
  for (int i=0;i<k;i++) {
    dijkstra(i, k, REF, v, wtimes);
  }
  cout.rdbuf(coutbuf);
  cout << "DONE";
}
```

Appendix B: Even-Odd Rule Program

```
#include<bits/stdc++.h>
#include"base.h"
#include"areas.h"
#include"busstops.h"
//Used to read in the data from source files
using namespace std;
szstruct csz;
vector<vector<cdstruct> >cszgeoms;
vector<cdstruct>cgeom;
long double errormargin=0.00001;
set<long long>alr_vertices;
bsstruct cbs;
ofstream outfile("busstopswsubzones.csv");
string subzone[5022];
string town[5022];

void testGeom(){
  int inpoly=0;
  long double grad,rise,run,tester;
  for(int i=1,j=0;i<cgeom.size();j=i++){
    if((cgeom[i].lat>cbs.nlat)!=(cgeom[j].lat>cbs.nlat)){
      rise=cgeom[i].lat-cgeom[j].lat;
      run=cgeom[i].lon-cgeom[j].lon;
      grad=run/rise;
      tester=grad*(cbs.nlat-cgeom[j].lat)+cgeom[j].lon;
      if(cbs.nlon<tester+errormargin)inpoly++;
    }
  }
  if(inpoly%2==1){
    subzone[cbs.index]=csz.name;
```

```
    town[cbs.index]=csz.townname;
  }
}

void testSubzone(){
  for(vcdit=cszgeoms.begin();vcdit<cszgeoms.end();vcdit++){
    cgeom=*vcdit;
    testGeom();
  }
}

void PIP(){
  for(bssit=bs.begin();bssit<bs.end();bssit++){
    cbs=*bssit;
    for(szsit=sz.begin();szsit<sz.end();szsit++){
      csz=*szsit;
      cszgeoms=csz.ncoords;
      testSubzone();
    }
  }
}

int main(){
  readSubzones();
  readBusStops();
  //Functions implemented in the header files to read in the data
  PIP();
  FILE *f = fopen("mrtwsz.csv","w");
  for(bssit=bs.begin();bssit<bs.end();bssit++){
    cbs=*bssit;
    cout<<cbs.subzone;
fprintf(f,"%d,%d,%s,%s,%s,%s,%s,%s,\n",cbs.index,cbs.code.c_str(),cbs.name.c_str(),
cbs.oglat.c_str(),cbs.oglon.c_str(),cbs.roadname.c_str(),subzone[cbs.index].c_str()
,town[cbs.index].c_str());
  }
}
```