SMTP (Math) Research Paper

Connect Singapore

Huang Qirui 3S1 (10) (Leader)

Chua Rui Hong 3S1 (5)

Ng See Jay 3S1 (23)

Hwa Chong Institution

(High School)

Table of Contents

Abstract

Introduction

Rationale

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).

Providing connectivity to virtually all places in Singapore at very affordable prices, the public transport system is undoubtedly very important to Singaporeans. However, not all areas in Singapore are equally well-connected. 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 through analysing and applying graph theory on the data provided by the Land Transport Authority (LTA) and the Government at Data.gov.sg. 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.

Terminology

Term	Explanation
Node (etc. Opp Natl JC, bus stop 41071)	A bus stop, bus interchange, Mass Rapid Transit (MRT) station, or Light Rapid Transit (LRT) station in Singapore.
Town (etc. Bukit Timah)	A geographical planning area in Singapore, as defined by the Urban Redevelopment Authority (URA), or referring to the set of all nodes found in that town.
Subzone (etc. Coronation Road subzone)	A geographical subzone in Singapore, as defined by the URA, or referring to the set of all nodes found in that subzone.
Area	Blanket term covering town and subzone. Note: when we refer to connectivity between an *area* and another/all *area(s)*, the two references to *area* mean areas of the same type (town or subzone).
Optimal connection	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).
General connectivity (of 1 area/node)	Average ease of getting from nodes in one area to **all other nodes** (including other nodes within that area) through optimal connections, measured through time taken or average speed.
Connectivity/	General, intra-area, and inter-area connectivity.

Connectivities	

Literature Review

Pathfinding Algorithms

Graph theory would be useful in helping us find the shortest path from one node to another through a network. Thomson and Richardson (1995) wrote that Graph Theory allows a road network to be easily represented. One pathfinding algorithm is Dijkstra's algorithm, which is a widely used shortest path algorithm for its efficiency. It is guaranteed to find the shortest route. It works on a directed and weighted graph that contains only non-negative edge weights. A real life application of Dijkstra's algorithm had been to use it in directing network routing to find the fastest or the most cost-efficient route. Previous researches have concluded that Dijkstra's algorithm is a "useful graph theoretic mechanism for optimisation process of network connectivity" (Ibrahim, 2007). Similarly, in our project we will also consider using Dijkstra's algorithm to find the optimal time to travel from every node to every other node so as to measure the connectivity of the node.

Point-in-Polygon Problem

In order to analyse the data for subzones in Singapore, we have to classify the nodes in their subzones. To do that, all nodes and subzones are represented graphically as points and polygons respectively. Though trivial for humans, it is not easy for a computer to find whether a given point lies in a polygon. Hormann and Agathos (2004) state that there are two main methods for doing this, the *even–odd rule* and the *winding number* method. In our project, we

will need to implement an algorithm to find which of Singapore's planning areas and subzones each node is in. We will most likely use the *even-odd rule* as its efficient implementation is much simpler. Let R be the point to be tested. "[A] line is drawn from R to some other point S that is guaranteed to lie outside the polygon. If this line RS crosses the edges … of the polygon an odd number of times, the point is inside P, otherwise it is outside" (Hormann & Agathos, 2004). Special attention must be given to cases where the point lies on an edge of the polygon, in which case the algorithm will otherwise fail.

Transport Connectivity Studies

There have been numerous studies on transport connectivity. However, most of them are on the overall connectivity of the network rather than that of a place. One exception is the following. In 2019, the United Kingdom National Infrastructure Commission (NIC) 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.

For urban connectivity, observed connectivity for an area was measured by averaging the travel times from each point in it to its centre, weighted by demand, using the proxies of population and employment. The connectivity value was calculated by dividing the observed connectivity with the "crow fly connectivity", the weighted average time taken to travel between the points and the centre in a straight line at 50 kilometres per hour. Inter-urban connectivity was measured in the same way, but between centres of different places instead of between points and a centre. The results helped advise the NIC on what areas needed new infrastructure or infrastructure upgrades to help increase connectivity.

In our project, we will aim to do a similar study of Singapore, though our methodology does have some differences from the one used by the NIC. For example, we are unable to assign weightages to nodes based on demand due to the lack of any proxy data that are specific enough.

Existing Plans to Improve Connectivity

The Urban Redevelopment Authority (URA) and the Land Transport Authority (LTA) (2018) published the Land Transport Master Plan 2040. In it, it mentions that the government plans to improve connectivity between places such that 8 in 10 households will be in a 10 minute walk from a MRT Station and 9 in 10 households will only 45 minutes to travel to the Central Business District. This will be done by the addition of 3 new lines, the Cross Island Line, which intends to connect the East and the West regions, the Thomson East-Coast Line, which intends to connect the North and the South-East areas to the City and the Jurong Region Line, which intends to improve connectivity in the Western areas. The government, unfortunately, did not release any findings about the connectivity of Singapore and hence this project aims to use a mathematical approach in determining areas with low connectivity before proposing solutions to join them together.

Methodology

Objectives and Research Questions

The following are the objectives of this research:

1. To determine how well-connected different places of Singapore is.

2. To determine how effective the public transport system is in servicing different places.
3. To find trends in the connectivities of subzones and towns.
4. To analyse the effectiveness of proposed MRT stations in improving the connectivities of different places in Singapore..

The objectives are condensed into 3 research questions.

1. What is the general connectivity of every node in Singapore?
2. What are the general connectivities of every subzone in Singapore and what trends can be found in their connectivities?
3. How will the upcoming MRT lines increase connectivity?

Data Sources

Various data sources were searched to provide the needed information for this research.

1. Land Transport Datamall provided the bus service routes, bus service frequencies, as well as data on the location of bus stops.
2. data.gov.sg provided the boundaries of subzones and towns in Singapore.
3. data.world provided the information on the MRT and LRT stations and travel times.
4. Google Maps Platform Directions/Distance Matrix Application Programming Interface (API) provided the travel times between ordered pairs of bus stops with direct bus connection(s)

Data Generation

Data Sources

Firstly, the walking routes were generated as walking time between nodes has to be taken into account. The locations of the existing bus stops, MRT and LRT Stations (5180 in total) were coded into a programme as graphical representations. Next we need to determine to time it takes to walk between near nodes for transit. We decided that the walking routes cannot exceed 500 metres as people would usually take public transport instead of walking if the distance is too long. A walking speed of 1.11 m/s is assumed as Google Maps use a walking speed of about 1.25m/s. However, Bus Stops and MRT stations are usually crowded, hence reducing the walking speed slightly. The time taken for the walking routes were then recorded.

Research Question 1

Following that, we obtained the average bus service frequencies and bus travel times. The data is collected from Google Maps. We collected bus service frequencies and travel times from 8 different periods. This is 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 while others such as residential areas may be less so. The 8 different times are

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)

The average bus travel time and service frequency is then calculated.

For MRT and LRT travel times, the data is provided already and there should not be a difference between peak and off-peak as the trains move at the same speed.

Furthermore, an addition of 20 seconds is added whenever a bus stops at a bus stop in order to take into account possible increase in time of a bus journey when stopping. Similarly for MRT and LRT, a waiting time of 150 seconds is added for peak hours and 300 seconds for non-peak hours.

After all the information on travelling times is collected, we implemented Djikstra's algorithm, a computer programme that allows us to find the shortest route between any 2 nodes. This allows us to find the optimal route to travel between any 2 nodes. This is the best way to measure connectivity since people usually take the fastest route when travelling.

The travelling times between every 2 pair of nodes allow us to determine the general connectivity. General connectivity(GC) is measured in 2 different ways, by **time** and **speed index**.

GC is measured by time so as to determine how well-connected a place is. The formula for GC of a node (time) is as follows:

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

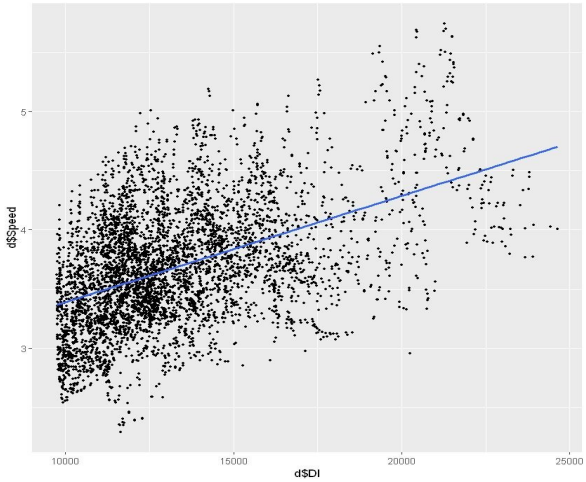
GC is also measured in speed index. Speed Index is a measure of how effective the public transport system is in servicing an area. This is done by removing the location of a node as a factor. In order to calculate Speed Index, speed and distance index must first be determined. They are measured as follows:

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

Hence, the average speed of a node is:

$$Average\ speed\ of\ travel\ of\ a\ node\ \frac{Sum\ of\ speed\ of\ travel\ to\ every\ other\ node}{Total\ number\ of\ nodes}$$

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



As can be seen from the graph, there is a positive correlation (regression value of 22.7%) between distance index and average speed, showing that location does indeed affect speed.

In order to eliminate location as a factor, speed index is instead chosen. The expected speed is derived from the best fit line while the real speed is the speed value determined above.

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

Research Question 2

The General Connectivity (GC) of a subzone can be determined by its constituent nodes through this formula (for both time and speed index):

$$General\ Connectivity\ of\ a\ subzone\ = \frac{Sum\ of\ the\ General\ Connectivity\ of\ the\ constituent\ nodes\ of\ a\ subzone}{Number\ of\ nodes\ in\ a\ subzone}$$

To determine which node is in which subzone, we employed the even-odd algorithm. We represented each node as a coordinate of a graph based on their longitude and latitude and each subzone as a polygon based on their vertices. Through the use of the even-odd algorithm, we successfully classify which subzone each node is in.

Algorithms Used

Dijkstra's Algorithm

The Dijkstra's Algorithm is a shortest path algorithm that can be used to find the shortest route to get from a given node to every other node. We may be using this algorithm to determine the shortest route between any 2 nodes. The time taken to travel the shortest route will be then used as the optimal time taken to travel between those 2 nodes and be analysed on. This is an overview of how the Dijkstra's algorithm work:

1. First, it maintains a list of nodes to visit next sorted ascendingly by distance
2. Now, it inserts the starting node to the list
3. While there are still nodes in the list, it will get the first node in the list and remove it from the list

4. If a shorter route that passes through the current node to an adjacent node is discovered, it will update the current shortest distance for that node and insert the updated node into the list
5. Repeat step 3 until the list becomes empty, which means that no further updates were made and the shortest route from the starting node to every other node have been found

Examples of shortest routes generated by Dijkstra's Algorithm:

1. Chinese High School (41051) to Opp Blk 5022 (54309)

   Step 1: Walk to Tan Kah Kee MRT Station

   Step 2: Take the MRT to Ang Mo Kio MRT Station

   Step 3: Walk to Ang Mo Kio Station Bus Stop

   Step 4: Take Bus 73a to Blk 5022

   Step 5: Walk to Opp Blk 5022

2. Chinese High School (41051) to Larkin Terminal (46239)

   Step 1: Walk to Tan Kah Kee MRT Station

   Step 2: Take the MRT to Kranji MRT Station

   Step 3: Walk to Opp Kranji Station Bus Stop

   Step 4: Take Bus 170X to Johor Bahru Checkpoint

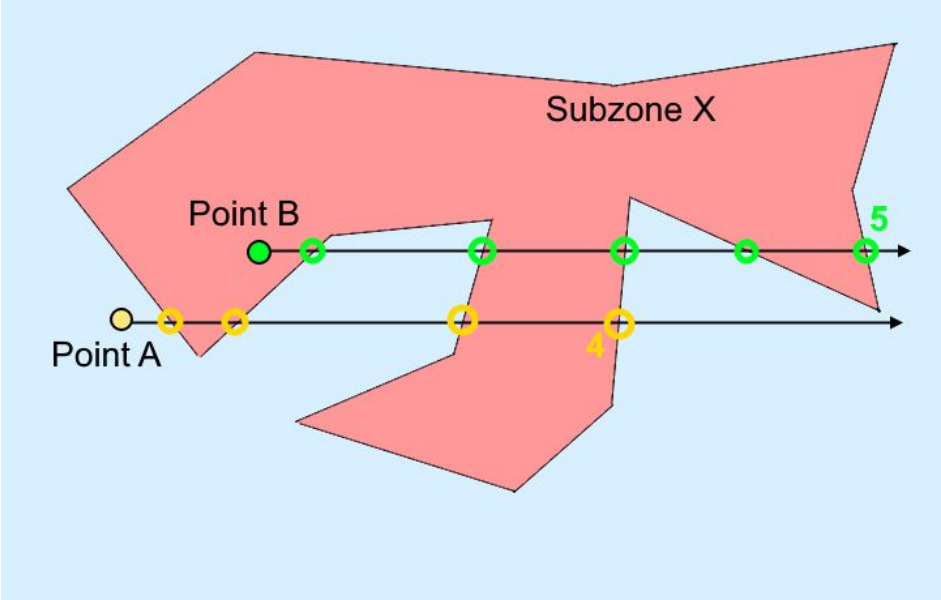
   Step 5: Take Bus 170 to Larkin Terminal

Even-Odd Algorithm

The Even-Odd Algorithm, as mentioned earlier, allows us to determine if a point is inside a polygon. We will be using it to classify the nodes in their respective subzones for RQ2. This is an overview of how it will work:

1. Represent a subzone as a polygon and a node as a point.
2. From the point, draw a ray to infinity in a direction of a point at the edge of the subzone polygon.
3. Count the number of segments from the given polygon that the ray crosses. If the number is **odd**, the point is **inside**. If the number is **even**, the point is **outside**.

Examples of the even-odd algorithm:



As can be seen from the example, the number of times Point B crosses Subzone X is 5, an odd number, thus B is inside Subzone X, the number of times Point A crosses Subzone X is 4, an even number, thus A is outside Subzone X.

Results

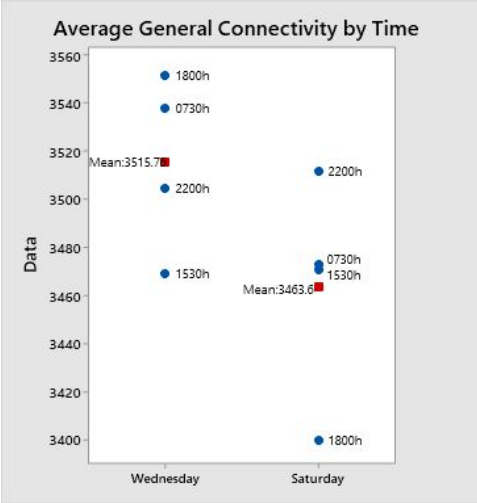
Research Question 1

What is the general connectivity of every node in Singapore?

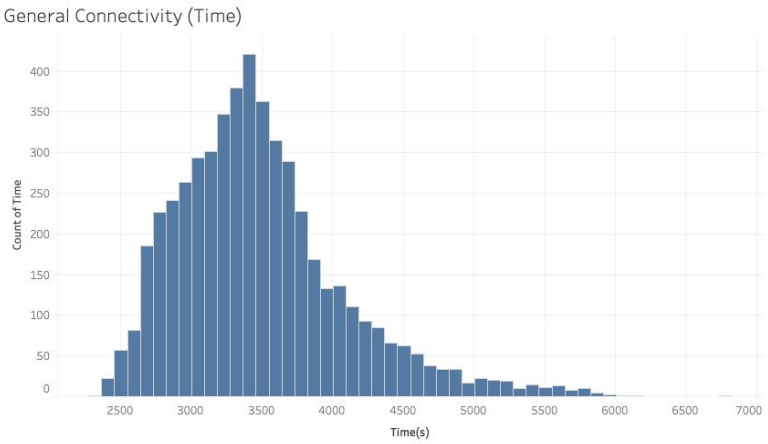
The dataset can be found in the Appendix.

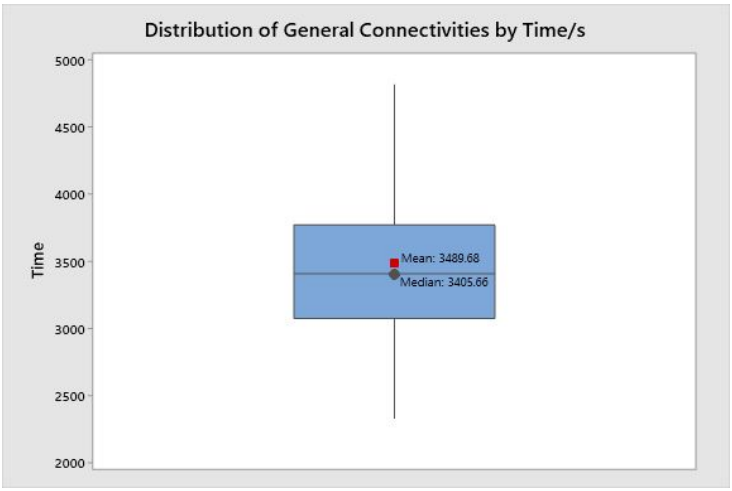
General Connectivity by Time

The average general connectivity by time of all 8 different time periods of data collected. Time value is preferred to be low.



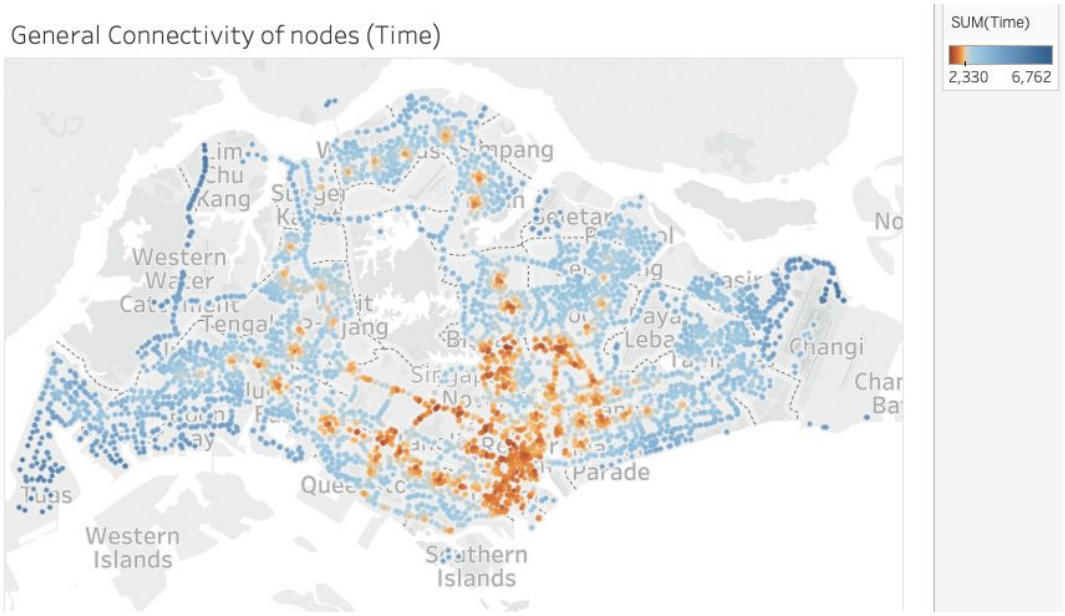
The general distribution of the average general connectivity by time





The following are some statistics of general connectivity by time. There are 5180 total number of nodes ranging from 2230.03s to 6761.58s. The range is 4431.55. The mean is 3489.68s and the median is 3405.66s. The first and third quartile is 3072.11s and 3769.87s with an interquartile range of 697.67s. There is a standard deviation of 601.08s

Below is a map of all the nodes in Singapore arranged by their General Connectivities by Time:



As expected, most of the orange dots (nodes with low time value) are located in central locations while the nodes displayed as blue (nodes with the highest time value) are located at the extreme ends of Singapore.

Here are the 3 nodes who has the smallest general connectivity (time) value. All 3 nodes are MRT interchanges with 2 MRT lines.

1. Bishan Mrt Station (2330.03s)
2. Botanic Gardens MRT Station (2368.92s)
3. Newton MRT Station (2379.05s)

Here are the 3 nodes who has the largest general connectivity (time) value. All 3 nodes are in far-flung places.

1. Larkin Ter, located in Johor Bahru (6761.58s)
2. Halliburton, located in Tuas View (6127.78s)
3. See Hup Seng, located in Tuas View (6058.23s)

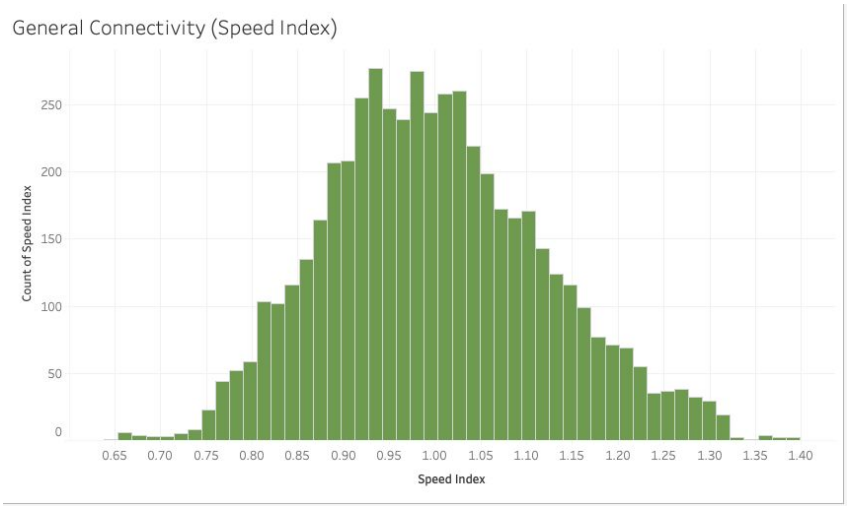
For MRT Stations, the general connectivity by time is 2866.43s, this is about 600 seconds (or 10 minutes) faster than the average node, which is 3489.68s.

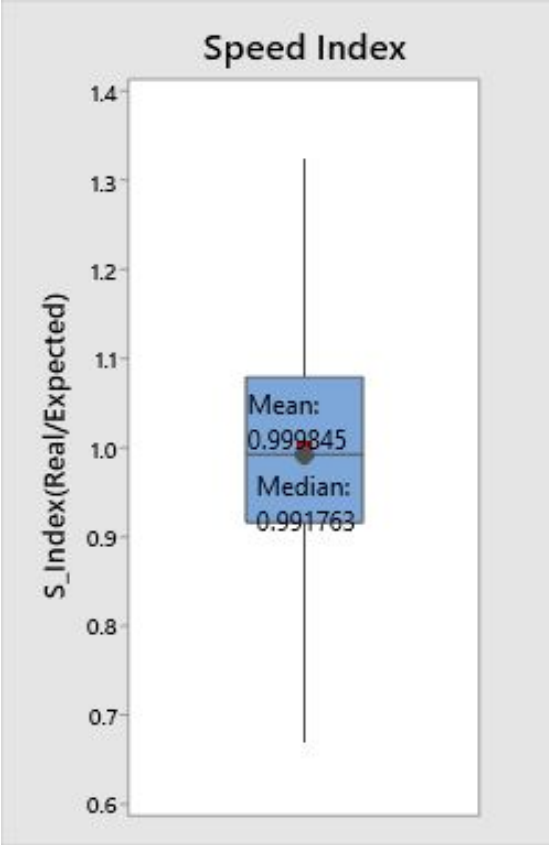
$$\frac{2866.43}{3489.68} \times 100\% = 82.1\%$$

The average MRT Station is about 17.9% faster than the average node.

General Connectivity by Speed Index

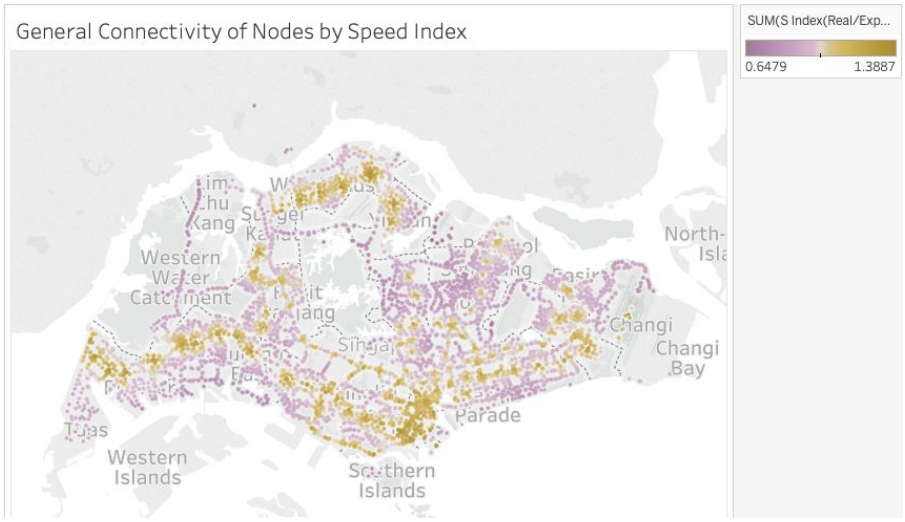
The general distribution for the general connectivity of all the Nodes by Speed Index is as follows:



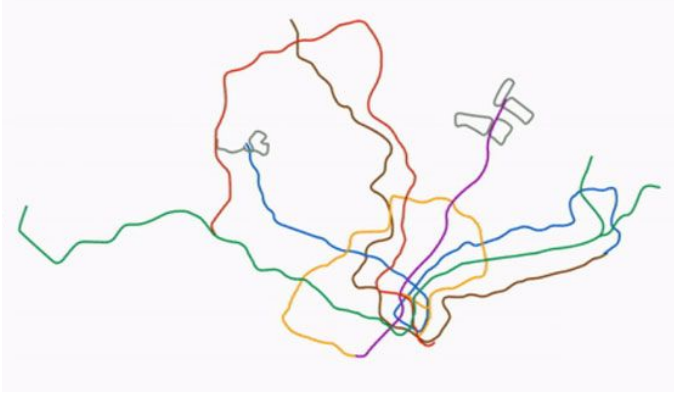


The following are some statistics of general connectivity by speed index. There are 5180 total number of nodes ranging from 1.3877 to 0.6479. The range is 0.7398. The mean is 0.9998 and the median is 0.9918. Naturally, the mean and median should be close to 1 since they are derived from a best-fit line. The first and third quartile is 1.0788 and 0.9151 with an interquartile range of 0.1637. There is a standard deviation of 0.1204.

Below is a map of all the nodes in Singapore arranged by their General Connectivities by Speed Index:



The yellow dots representing good connectivity values may seem quite spread out and at times random. However, if we display an image of the MRT lines of Singapore and compare it with the map above, we can notice some similarities.



*Brown line represents the yet to be completed Thomson-East Coast Line, thus, please ignore it for now

As can be seen from the 2 maps, General Connectivity through speed has a relation with MRT lines. This probably proves that areas with MRT Lines will be more effectively served and connected by public transport as MRT travels faster compared to buses.

Here are the 3 nodes who has the best general connectivity (subzone) value. All 3 nodes are MRT stations, which is similar to the general connectivity by time results.

1. Outram MRT Station (1.389)
2. Tanjong Pagar MRT Station (1.385)
3. Sembawang MRT Station (1.377)

Here are the 3 nodes who has the worst general connectivity (subzone) value.

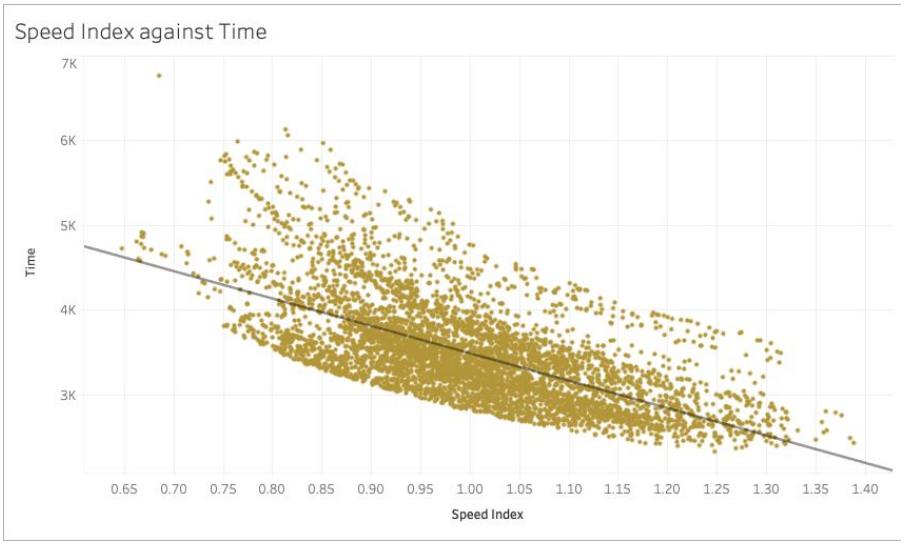
1. Bef RSAF Roundabout (0.648)
2. Bef West Camp Rd (0.663)
3. Air Force Sch (0.665)

For MRT Stations, the general connectivity by speed index is 1.18, while the average node is 1.00.

$$\frac{1.18}{1.00} \times 100\% = 118\%$$

The average MRT Station is about 18% more efficiently served by the public transport system than the average node.

Relation between Time and Speed Index



As observed from the graph, there is a clear relation between general connectivity by Time and general connectivity by Speed Index. The p-Value is a value used to calculate the relation between 2 axes in a linear graph, of which the lower the value, the higher the chance of there being a relationship. The p-Value for the relation between Time and Speed Index is less than 2.2×10^{-16}. There is also a regression line fit of 41.83%, indicating strong relations between the 2 values.

For general connectivity by time, MRT is shown to be 17.9% faster than the average node while for general connectivity by speed index, they are 18% faster. These values are very similar.

Through this we can conclude that the efficiency in which the transport system services a node does indeed affect the connectivity of a node.

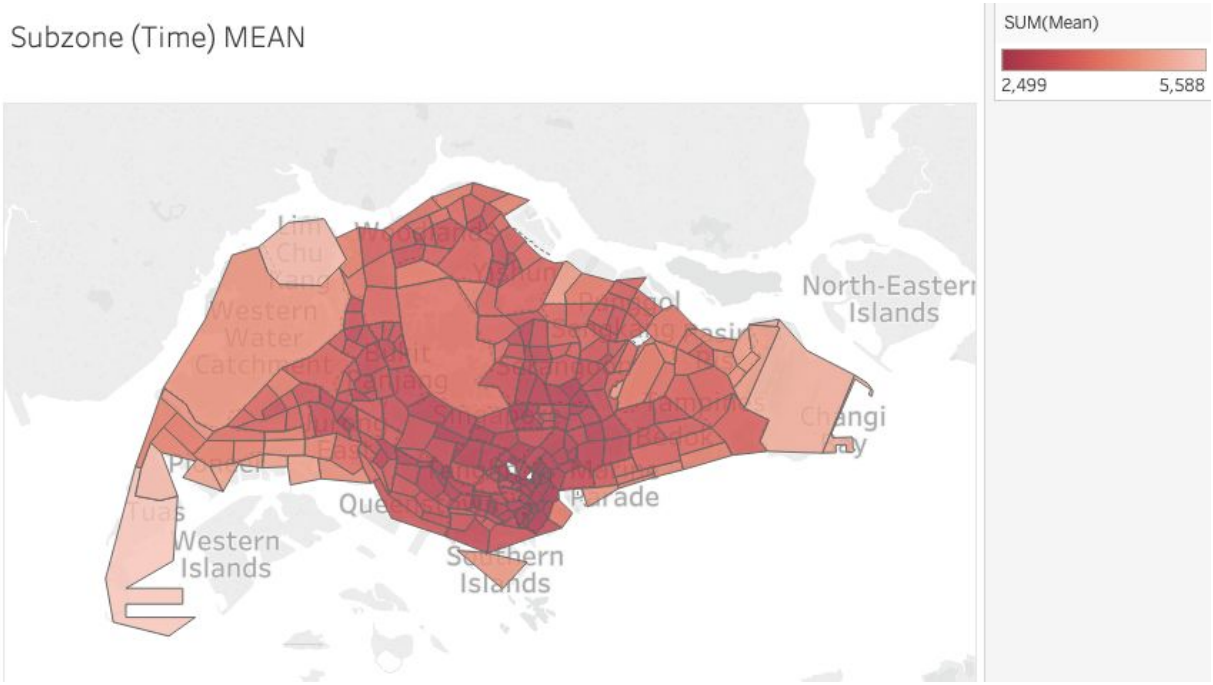
Research Question 2

What are the general connectivities of every subzone in Singapore, and what trends can be found in their connectivities?

The datasets can be found in the Appendix.

General Connectivity of Subzones by Time

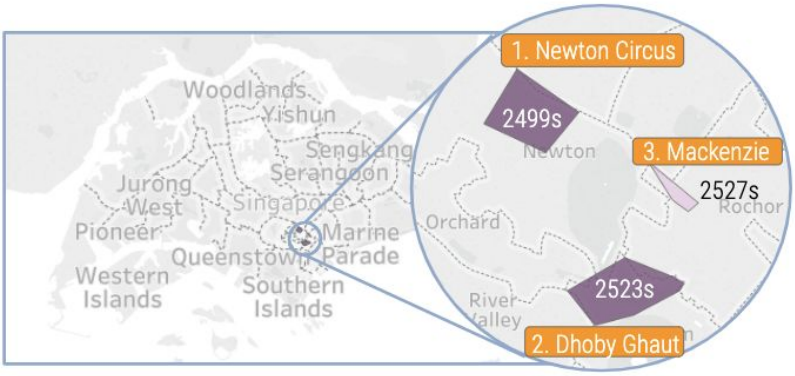
The map for general connectivity of subzones by time is shown below:



As can be seen from the map, most of the well-connected subzones are located centrally. The top 3 subzones are all located at the central business district. They are:

1. Newton Circus (2499s)
2. Mackenzie (2527s)
3. Dhoby Ghaut (2523s)

Map for easy reference:



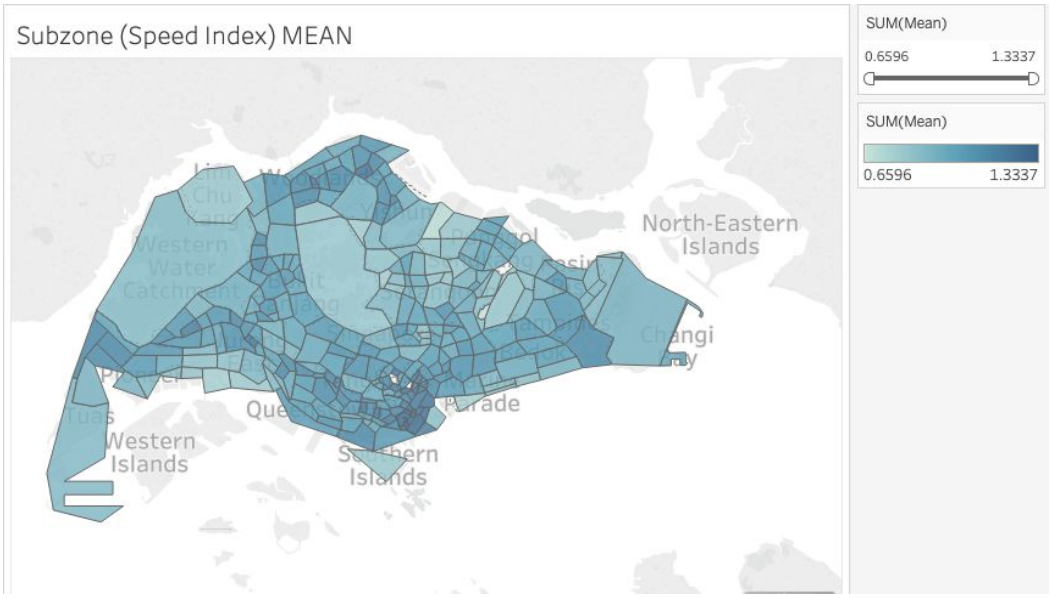
On the other hand, the worst 3 subzones are located far in the west at the borders of Singapore, they are:

1. Tuas View Extension (5588s)
2. Tuas View (5303s)
3. Lim Chu Kang (5279s)

Map for easy reference:

General Connectivity of Subzones by Speed Index

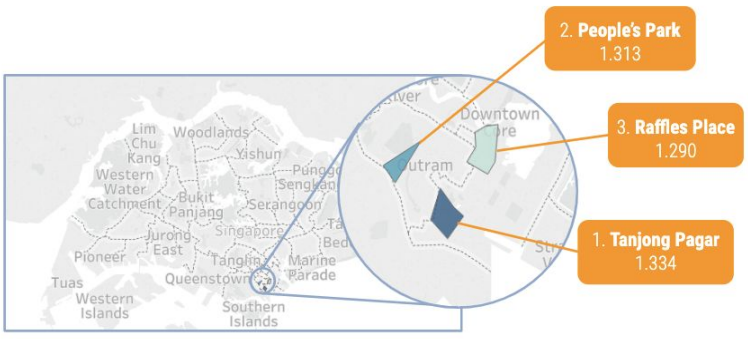
The map for general connectivity of subzones by speed index is shown below:



Similar to the results of general connectivity of subzones by time, all the subzones with the highest speed index values are located in the center. Here are the top 3 subzones, they are located in close proximity to one another. They are:

1. Tanjong pagar (1.334)
2. People's Park (1.313)
3. Raffles Place (1.290)

Map for easy reference:

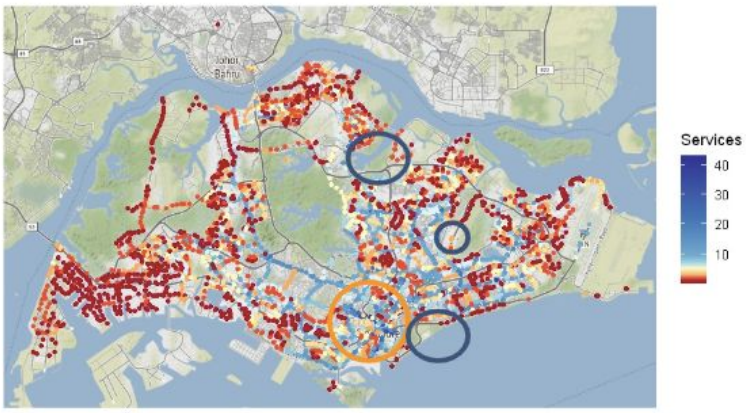


Here are the worst 3 subzones. Weirdly, however, they are located rather centrally and not at the border of Singapore we would expect them to be at. They are:

1. Airport Road (0.659)
2. Seletar Aerospace Park (0.695)
3. Marina East (0.747)

Map for easy reference:

The explanation can be found when compared to a map showing the number of bus services per bus stop. The above 3 subzones are found in places with few bus services, bus stops and little to no MRT and LRT stations. Below is a map showing the number of bus services servicing a node. Blue circles show the areas where the worst 3 subzones are located. Orange circles show areas where the best 3 subzones are located (shown in one circle due to proximity).



Research Question 3

Limitations

Firstly, some assumptions had to be made regarding the pathfinding programme. We set the walking speed to 1.11m/s. Even though this was the best estimate we got, we cannot be sure that it is entirely accurate or representative of everyone. Other estimations include a fixed addition of 20 seconds every time a bus stops at a bus stop as well as the fixed waiting between entering a MRT/LRT station and boarding a train, which is at 150s for peak and 300s for non-peak. These fixed values are estimates but the results should not differ too far as they are applied for every node and only makes up for a fraction of the total time spent on a journey.

Secondly, taxis are not taken into consideration. Taxis are classified as part of the public transport system. However, it is not included as we felt that consumers who take taxis generally do not take other forms of transport, and vice versa.

Thirdly, there is a lack of time to explore and find more correlations with the data sets.

Lastly, there may be a limited accuracy of data sources as we only picked 8 different times of the day to measure the average general connectivity by time. While these 8 times were spread out as evenly as possible to take into account peak and non-peak hours, weekdays and weekends as well as morning, afternoon and evenings, there is no guarantee that it is the accurate average value.

Extensions

Firstly, instead of measuring connectivity from nodes, we can extend this project to measure connectivity from regular coordinates (i.e. anywhere in Singapore) instead. This can give us more accurate results, especially pertaining to connectivities of subzones.

A pathfinding application can be developed to allow users to find the shortest path to their desired destination from their current location. This is possible as we already have the data for the shortest path between any 2 nodes.

A website with all our data obtained could be set up to allow others to easily access and find our data for other research.

Lastly, we can also measure the connectivity of a subzone by finding the average connectivity value of all the nodes in a subzone to every other node in that particular subzone. This can help us determine how connected a subzone is within itself.

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

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

Thomson, R. C., & Richardson, D. E. (1995, September). A graph theory approach to road network generalisation. In *Proceeding of the 17th international cartographic conference*(pp. 1871-1880).

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

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

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

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

Modiano, E. (n.d.). Lectures 18: Routing in Data Networks. In *6.263/16.37 DATA NETWORKS*. [PDF File] Retrieved from http://web.mit.edu/modiano/www/6.263/lec18.pdf

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

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

Data.gov.sg. (n.d.). Retrieved July 28, 2019, from https://data.gov.sg/

Modern Data Catalog for Analysis & Teamwork. (n.d.). Retrieved July 28, 2019, from https://data.world/

Geo-location APIs | Google Maps Platform | Google Cloud. (n.d.). Retrieved July 28, 2019, from https://cloud.google.com/maps-platform/

Appendix

Program used to calculate fastest routes through Dijkstra's Algorithm (C++)

```
#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; //current bus stop, current time spent, prev service
	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;
			}
			//cases
			//if (prev service train, next service train) CANNOT
			if ((s=="TRAIN")&&(i.second.second=="TRAIN")) continue;
			//if (next service walk) add no time
			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));
                                }
                        }
                        //if (next service train) add train wait time
                        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));
                                }
                        }
                        //if (prev service not same bus, next service is bus) add bus wait
time, else add no time
                        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]; //bus stop number, time,
service
       map<string, int> wtimes;
       int REF[10000]={};
       string FILE;
       int TIME;
       cin >> FILE >> TIME >> TWAIT >> direc;//>> BWAIT; //file to use, time of day
(1, 2, 3 or 4), train waiting time (120 or 270), time bus spends at a bus stop
       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);
       }
       //for (int i=0;i<3;i++) {
       //     dijkstra(i, k, REF, v, wtimes);
       //}
       cout.rdbuf(coutbuf);
       cout << "DONE";
}
```

Program used to find the subzone each node is in (C++)

```
#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];

int 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;
	}
}

bool testSubzone(){
	for(vcdit=cszgeoms.begin();vcdit<cszgeoms.end();vcdit++){
		cgeom=*vcdit;
		if(testGeom()) return 1;
	}
}

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.nam
        e.c_str(),cbs.oglat.c_str(),cbs.oglon.c_str(),cbs.roadname.c_str(),subzone[cbs.
        index].c_str(),town[cbs.index].c_str());
        }
}
```