An Analysis of Subway Networks using Graph Theory and Graph Generation with GraphRNN

Kuhan Jeyapragasan Gita Krishna Yash Maniyar
Department of Computer Science
Stanford University
{kuhanj, gitakris, ymaniyar}@stanford.edu

December 11, 2019

1 Introduction

This project analyzes and evaluates various global subway and metro systems in large metropolitan cities, in an attempt to make recommendations about how best to design subway systems in developing cities. Our project will consist of two major parts. The first part will be an analytical comparison and evaluation of established metro systems, by looking at the overall efficacy and quality of these systems through various graph qualities and metrics. The second part of this project involves using graph generation to make predictions about overall structure of subway systems in order to create an optimal transportation network, which could potentially be used as a skeleton or starting point when creating subway networks in developing cities.

2 Review of Relevant Prior Work

2.1 Networked analysis of the Shanghai subway network, in China - Zhang, Xu, Hong, Wang, Fei

In this paper the Shanghai subway network is analyzed using networked analysis (in particular, the connectivity, reliability and robustness of the network), and its topological characteristics are investigated according to complex network theory (invested parameters include degree distribution, network efficiency, node betweenness and edge betweenness). This paper analyzes various network measures using both weighted and unweighted graphs, which is something we also explored. By default, the edges in our metro system graphs were unweighted, and we incorporated different weights to represent whether or not two stations were in the same line.

2.2 Network Analysis of the Tel Aviv Mass Transit Plan - Sharav, Bekhor, Shiftan

This paper analyzes the transportation system in Tel Aviv, Israel, using graph theory and other network measures. It analyzes network performances using measures such as connectivity, treeness, circuity, accessibility, and entropy. The main technical content of this paper is in the extraction of these features about the Tel Aviv transportation system. This paper relates to the topics presented in the course as it pertains to general graph structure theory and analysis of networks. We incorporated various network metrics used in this paper.

2.3 A Network Analysis of World's Metro Systems - Wu, Tse, Dong, Ho, Lau

This paper studies network properties of metro systems in five major cities, and focuses on two metrics (average station density and station load). These two metrics are derived using network parameters that affect performance such as degree distribution and network efficiency. Certain technical graph theory ideas discussed include general topological properties of subway networks, characteristic path lengths, clustering and transitivity, and structural efficiency.

3 Model/Algorithm/Method

3.1 Data

For this project, we compiled subway/metro network datasets for the following cities: Vienna, Tokyo, London, Washington D.C, Chicago, Madrid, Beijing, New Delhi, and Santiago. Since we wanted data pertaining to the line structure in the subway systems, we had to manually transcribe the networks and then generate graphs by reading in the text files we had created. This involved assigning each station a unique station ID, and creating lists representing different lines in the network consisting of station IDs.

Most of the graphs that can be generated from these datasets simply include nodes representing stations, and edges connected directly connected stations. This is sufficient to do some analysis, but we wanted to add information about what lines each station was part of, which is why we manually transcribed line information. First, we added a weight of 1 to every edge between stations that are adjacent on train lines. To include connectivity between lines but penalize the cost of switching between lines (transition time, waiting time, inertia/unpleasant nature of swapping), we initially thought to instantiate every connection with weight 1, and add a discount factor (of 0.75 for example) between all nodes in the same line, we realize that this formulation did not take into account the fact that switching lines should incur some cost, and that with 0 costs for transitions, our formulation would not penalize switches.

Ultimately we decided to instantiate connector nodes multiple times (once for each line the connector station serves), and add a weight of 2.01 between each of these instances of the connector node. Doing this makes two stations along the same line more closely connected (lower cost to travel between two nodes than two stations that are an equal number of hops apart, but on different lines). The weighting of 2.01 lets us track the number of switches required for any trip (by subtracting the total cost from the integer floor function of the cost). This approach is not without its own problems - the encoding of one station as multiple instantiations with different node labels increases the total number of nodes, average degree, and complicates calculations for average and max numbers of transfers, but we felt that these compromises were reasonable given that transferring lines is similar in time to traveling between stations.

Figure 1 shows graphical depictions of the networks that we created.

3.2 Graph Generation

After creating our initial graphs for our nine cities, we used these graphs for training to generate model subway graphs using two models - the graph generative model GraphRNN, developed by You et. al, and the alternate multi-variate Bernoulli model, which uses a different model for edge creation - namely modelling the edge creation function as a single layer multilayer perceptron (MLP).

3.2.1 GraphRNN

GraphRNN is a deep autoregressive model which addresses the challenges of modeling complex distributions over graphs, and efficiently sampling from these distributions. Specifically, these issues include the high-dimensional, non-unique nature of graphs, and the complex, non-local dependencies that exist between edges in a given graph.

GraphRNN learns to generate graphs: it is given a representative training set of graphs, and translates the graph generation process into a sequence of node and edge formations, conditioned on the sequence/graph generated so far. You et. al. introduce a Breadth First Search node ordering scheme to reduce the complexity of learning over all possible node sequences to a tractable search space, and reduce the model complexity via weight sharing with recurrent neural networks. The GraphRNN approach is described in Figure 2 - the graph generation is split into two processes - one that generates a sequence of nodes (Graph-level RNN), and another that generates a sequence of edges (edge-level RNN).

3.2.2 Multi-variate Bernoulli (MLP)

A variant of the GraphRNN model that we experimented with is GraphRNN with Multivariate Bernoulli. This also implements that graph-level RNN as a Gated Recurrent Unit, but the edge-level model varies. Here, the edge-level model is implemented as a single layer multilayer perceptron (MLP) with a sigmoid activation function. The output of this MLP is a vector where each element $\theta_{i,j}$ is the probability of edge (i, j). We sample edges in S_i^π independently with a multivariate Bernoulli distribution parameterized by θ_i.

Figure 1: Graph representations of subway networks

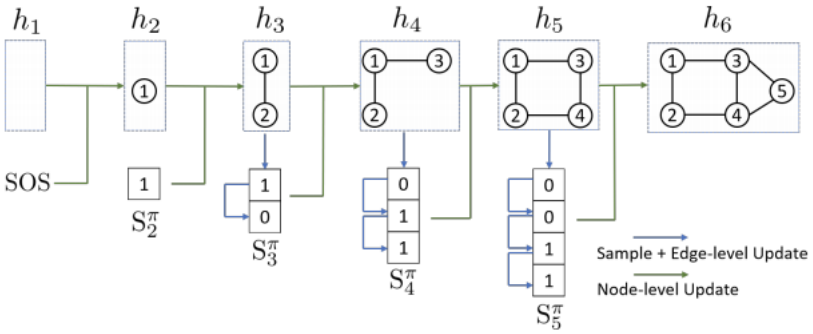


Figure 2: GraphRNN procedure

4 Mathematical Background

We used a normalization factor for our normalized average distance between nodes metric (since larger subway systems would naturally have larger average distances between nodes simply as a function of their size). To normalize for this, we looked at the example of average distance between nodes on just one line.

The total number of connections (without double counting) is given by $n^2 - n$, where n is the number of stations or nodes.
The sum of distances can be represented by the sum: $(n-1)*1 + (n-2)*2 + ... + 1*(n-1)$, or $\sum_{i=1}^{n-1}(n-i)(i)$.

The average distance comes to $\frac{\sum_{i=1}^{n-1}(n-i)(i)}{n^2-n}$.
Expanding this out we get $\frac{n*(n^2-n)-n(n+1)(2n+1)/6}{n^2-n}$. Taking the limit as n approaches infinity, we are left with $n - n/3$ in the numerator, leading to a normalization factor of $2n/3$.

4.1 Louvain Algorithm for Community Detection

We used the Louvain algorithm for community detection, which is a greedy algorithm that runs in $O(nlogn)$ time. This algorithm involves optimizing modularity via local changes in communities, and then aggregating communities into singular nodes to form a new network. These two steps are repeated until convergence occurs.

We can formally define the modularity gain by moving node i to community C as follows:

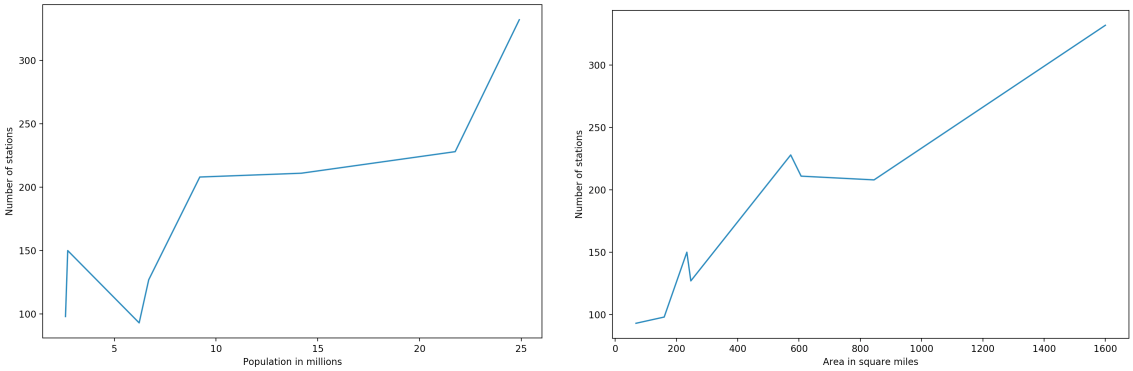
$$\Delta(Q)(i \to C) = [\frac{\Sigma_{in} + k_{i,in}}{2m} - (\frac{\Sigma_{tot} + k_i}{2m})^2] - [\frac{\Sigma_{in}}{2m} - (\frac{\Sigma_{tot}}{2m})^2 - (\frac{k_i}{2m})^2]$$

5 Findings

5.1 City Statistics

5.1.1 Population and Area

We did an analysis of the metropolitan population of the nine cities in question and how that relates to other statistics about the subway network in general. We saw that the number of stations tended to increase with the population of the city, which makes intuitive sense. We also saw that the more populated cities tended to have more "people per station", and the less populated cities had fewer "people per station". Based on our calculations, we see that a city adds a station for every 118,000 increase in population. For the analysis of city area, we see that a city tends to add a station for every increase in area of 6.5 square miles. Figure 3 is a graphical representation of this trend.



Population versus number of stations

Area versus number of stations

Figure 3: Cities statistics vs. population

5.2 Subway Statistics - Real Cities

Table 1: Subway statistics

City	Stations (w. duplicates)	Diameter	Avg. Degree	Normalized Distance	Avg. Transfers	Max. Transfers
Santiago	127 (144)	44.03	2.138888889	0.1533191069	1.208333333	5
Vienna	98 (109)	29.02	2.128440367	0.1631584698	0.8357882333	4
London	211 (303)	49.02	2.778877888	0.0787771319	1.081419033	4
Madrid	276 (328)	70.06	2.304878049	0.09625462288	2.360332391	6
Beijing	332 (390)	65.06	2.2	0.07799880494	2.217449047	8
New Delhi	228 (248)	65.04	2.096774194	0.1299562191	1.493854058	4
DC	93 (152)	31.04	2.986842105	0.124041918	0.8905817175	4
Chicago	150 (259)	46.02	3.281853282	0.09052256251	0.4513051386	3
Tokyo	208 (298)	38.03	2.712802768	12.63542271	1.296847499	4

5.2.1 Clusters

The Louvain Algorithm was able to cluster our subway networks into communities since the algorithm works on weighted graphs as well as unweighted graphs. We were unsurprised to see that the algorithm groups nodes roughly into communities based on their line, which speaks to inherent characteristics about how the stations are split between lines and whether or not that is optimal.

We performed an analysis of similarity between the actual line groupings of the graphs and the aggregated community partitioning from the Louvain algorithm. We saw that on average, the Jaccard similarity between each of the top half of the partitioned communities and the corresponding line was 0.42. We saw that the average highest Jaccard similarity was 0.64, and that the minimum was 0.20. We calculated the Jaccard similarity as follows:

$$J(A, B) = \frac{A \cap B}{A \cup B}$$

In Figure 4, we can see an example of Beijing's network after using Louvain Community Detection.

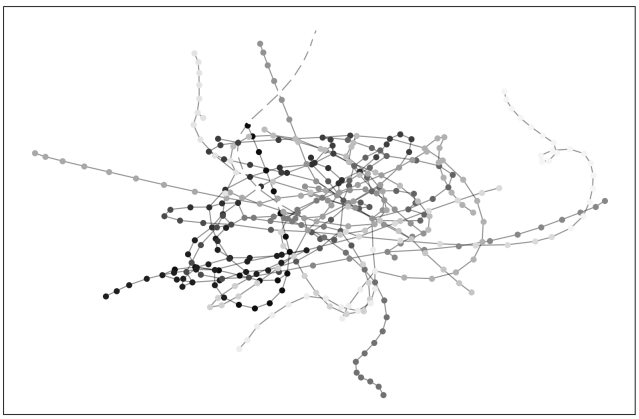


Figure 4: Beijing with Louvain Community Detection

5.2.2 Average Degree

The above statistics are closely related to the average degree of nodes - for subway systems with many stations serving more than one line, or connector 'hubs' serving several lines, the number of nodes with duplicates was significantly larger than the number without duplicates, and this was reflected in the average degree of nodes as well. Cities without many connector stations like New Delhi had a degree very close to 2 (and a similar numbers of nodes with and without connectors), while systems with many connections like Chicago had an average degree greater than 3 (and a large discrepancy between number of nodes with and without duplicates).

5.2.3 Diameter

Diameter describes the maximum number of edges required to get from one point on the subway to another over all possible trips. This statistic looks at the longest path (including transfers with weight of 2.01, and regular trips between stations with weight of 1).

5.2.4 Normalized Average Distance between Nodes

Initially we computed the cost of traveling between any two cities and took the average to determine this value. We realized as defined this definition is meaningless (since larger subway networks would have a higher average cost simply by virtue of having more). This is why we introduced the normalization facttor described above.

5.2.5 Average Number of Transfers per Trip

One metric often used to evaluate the design of subway systems is the number of transfers to get from one place to another, since transfers are not ideal for a number of reasons (increasing waiting times, disrupting continuity, energy required to leave a train, commute to the new platform, and wait for the next train.

5.2.6 Max Number of Transfers

Another related metric is the max number of switches needed to get from any station to another (assuming you take the shortest path). Statistics for the metrics described above can be found in Table 1.

5.2.7 Betweenness Centrality

Betweenness centrality of a node v (denoted $c_B(v)$) is, roughly, the measure of how frequently v is visited when traveling from any node in the graph to any other node in the graph (using the shortest possible path). It is calculated using the following formula:

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

where V is the set of all nodes in the graph, $\sigma(s,t)$ is the total number of shortest paths between nodes s and t (in our subway networks, this will usually be no more than 1), and $\sigma(s,t|v)$ is the total number of shortest paths between s and t that go through v (in a subway network, this value will usually be either 0 or 1). Figure 5 shows histograms counting the number of stations in each city with a given betweenness centrality. From these histograms, we see that almost all of the cities in our analysis have an extremely high number of stations with a betweenness centrality of almost zero. This means that these stations are not very often part of the shortest path a person could take from one station to any other station, or, in simpler words, these stations are not very “central". Figure 1 shows us that these cities have larger networks (in number of nodes) with lines that trail away from the central cluster, representing lines that reach out into suburbs of the city. The stations on a trailing line are probably not part of many shortest paths in the graph, other than paths that have one end on that line. This contributes to the highly skewed distribution we see in these cities' histograms in Figure 5.
Vienna, Santiago, and New Delhi stand out because they do not quite follow this pattern. Their distributions peak around 0.1, and are much less skewed. This means that, on average, stations in these networks are more central. Notably, these networks are smaller than the rest, and are more clustered in the center – there seem to be fewer lines that trail off away from the center, and those that do are shorter than their counterparts in the other cities.
Finally, the city with the station with the highest degree of betweenness centrality is Chicago, where one station has a centrality of just over 0.6, which is more central than any station in Vienna, Santiago, or New Delhi. More than half of all shortest paths go through this station. Just by looking at Chicago's centrality distribution, we can infer that it probably has many trailing lines shooting out from a small, central area. Chicago's graph, shown in Figure 1, confirms this.

5.3 Subway Statistics - Generated Graphs

We generated graphs using two variant graph generation methods with differing edge creation models - classic GraphRNN (RNN) and a multi-variate Bernoulli using a single multi-layer perceptron (MLP). Using our nine graphs as training data, we generated four batches of 1600 graphs, to test both the differences between models, and the impact of added training iterations on the quality of the graphs produced. The qualities of existing subway graphs we were looking to reproduce were:

1) Very low clustering coefficient (Below 0.1) - the vast majority of nodes would not be connected to neighbors' neighbors (other than connector nodes serving multiple lines) since this is inefficient in a subway network.

2) Average degree between 2 and 3 (lower than 2 would mean very few connector nodes but lots of end nodes/many lines which is infeasible for a subway, and a very high average degree would mean a waste of infrastructure

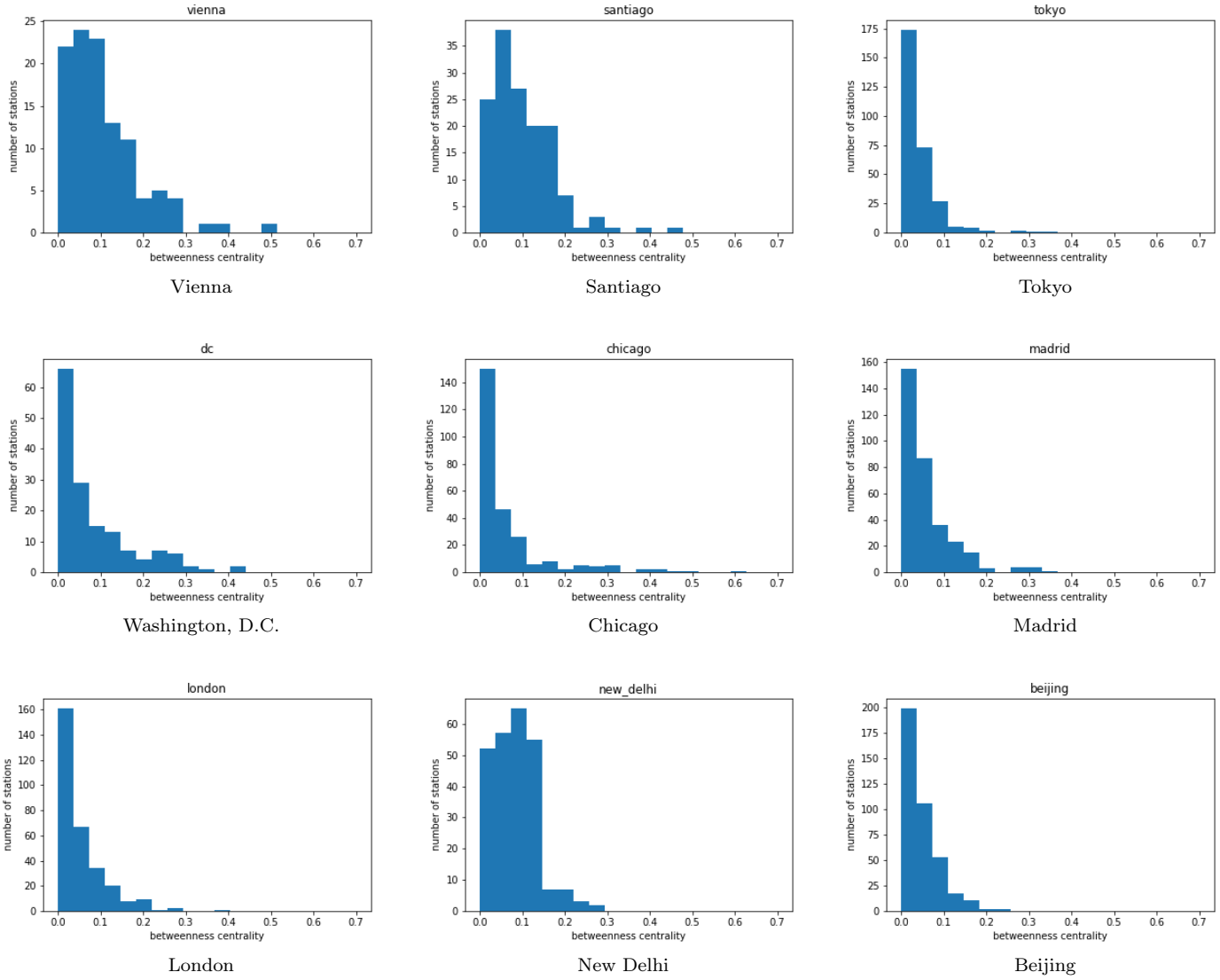


Figure 5: Betweenness centrality histograms for each subway network

creating paths that aren't necessary)

3) Low diameter and average path length (normalized to the number of stations/nodes). This means that connections are efficiently placed such that no trip is too long (doesn't require traversing a significant portion of the graph). In practice this meant a normalized distance of around 0.1 or less.

4) Low number of average and max. transfers/switches required to travel between stations.
Table 2 shows our results, comparing our generated graphs against the min, median, mean, and max values for network properties we analyzed for our nine city subway graphs. To simulate lines from our generated graphs, we assigned lines based on Community identification using the Louvain algorithm, which while imperfect, allowed us to calculate the same metrics used to evaluate the strength of existing systems (like average and max number of line switches needed for a trip between two stations). The imperfect nature of this line assignment led to the persistent issue of the Louvain algorithm repeatedly assigning fewer communities than the number of lines we'd expect for a subway with the number of nodes in the generated graphs, as well as groupings that did not necessarily correspond with the lines/groupings we'd see in real subways. This is what caused the average and max transfer statistics to be significantly higher than the median/mean values for real subways.

Graph	Stations	Diameter	Avg Degree	Clustering Coefficient	Normalized Avg Path Length	Avg. Transfers	Max Transfers
Min (Real)	109	29.02	2.096774194	0	0.06360112101	0.4513051386	3
Median (Real)	259	46.02	2.304878049	0.04596399535	0.09625462288	1.208333333	4
Mean (Real)	247.8888889	48.59111111	2.51437306	0.1375931616	0.1086255508	1.315101161	4.666666667
Max (Real)	390	70.06	3.281853282	0.429333751	0.1631584698	2.360332391	8
MLP_1600_1	298	41.05	3.194630872	0.1428537533	0.07110225475	2.072023783	5
MLP_1600_2	186	43.05	3.99	0.2768801987	0.098	1.923054688	5
MLP_3000_1	150	28.05	2.253333333	0	0.1135028978	2.175644444	5
MLP_3000_2	102	26.04	2.133410227	0	0.1745425402	2.133410227	6
RNN_2200_1	137	128.1	2.087591241	0.05766423358	0.4827998023	3.774734935	10
RNN_2200_2	87	39.03	2.229885057	0.05593869732	0.2065922707	1.749504558	3
RNN_3000_1	240	65.06	2.1	0.01083333333	0.1380946528	2.556111111	6
RNN_3000_2	212	57.06	2.08490566	0.00786163522	0.142726735	2.430580278	6

5.3.1 Multi-variate Bernoulli Single Multi-Layer Perceptron Generated Graphs

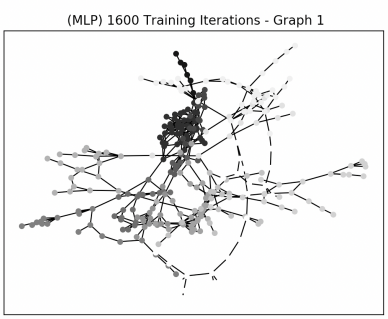
We see (both visually and statistically) that the MLP graphs with 1600 training iterations are the weakest overall - the graphs have highly clustered components and few of the 'tails/chains' characteristic of subway graphs. The MLP graphs with 3000 iterations are more realistic, but there is a fair bit of one/two/three node branches from long chains which we do not see in subway graphs. The metrics the MLP graphs performed well on are diameter (relative to the number of stations), normalized average path length, and number of average/max transfers. The MLP graphs did particularly poorly on clustering coefficient (too high for 1600 iteration training, and too low, literally 0 for 3000 iteration training), and average degree (for 1600 but not 3000 training iterations). The improvement between 1600 and 3000 iterations of training is quite clear for the MLP generated graphs.

5.3.2 GraphRNN Generated Graphs

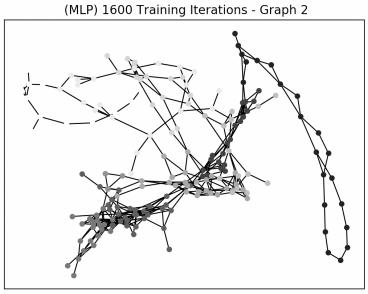
The graphs generated using GraphRNN were much more successful overall than the MLP graphs (as can be seen visually with the clean chain-like structures without offshoots or branching as was the case with the MLP graphs), but had some critical errors as well. For example, with the first sample graph with 2200 training iterations, the longest trip required traversal of 127 of 137 stations, indicating inefficient placement of connector stations in the graph. More generally, the diameter/max trip length was a problem for the RNN graphs, with diameter/station ratios being much higher than for real and MLP graphs. RNN also performed particularly poorly on the normalized average path length (much higher values than the real graphs, again indicative of insufficient connector nodes in the graph). The metrics where RNN performed well include average degree (low, close to 2), low clustering coefficient, and the number of transfers (average and max). We again see the added benefits of extra training iterations, as the 3000 iteration graphs performed similarly or better on all metrics than the graphs generated with 2200 training iterations.

5.4 Betweenness Centrality of Generated Graphs

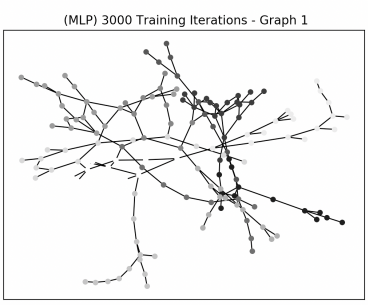
We compared betweenness centrality of the generated graphs to the betweenness centrality of the original graphs. Based on our randomly sampled graphs, the generated graphs follow approximately the same distribution regarding betweenness centrality. The betweenness centrality distribution is a little more skewed to the left, meaning that a



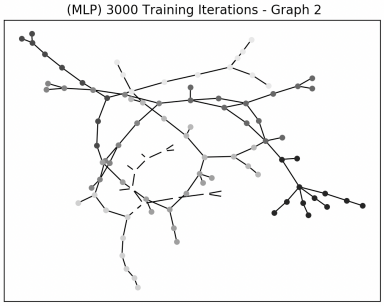
MLP - 1600 iterations Sample 1



MLP - 1600 iterations Sample 2

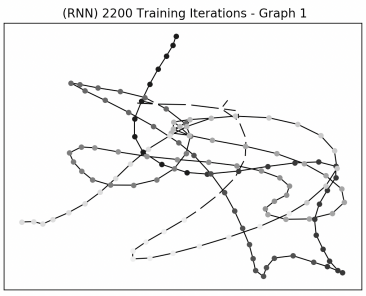


MLP - 3000 iterations Sample 1

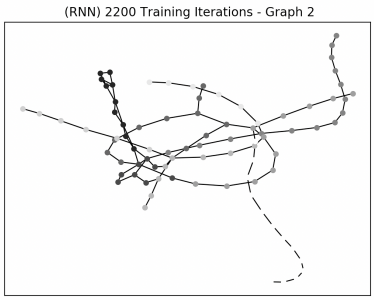


MLP - 3000 iterations Sample 2

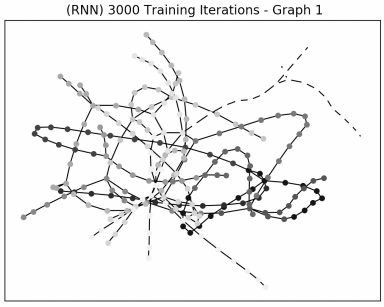
Figure 6: MLP generated graphs at various iterations, clustered into communities by Louvain



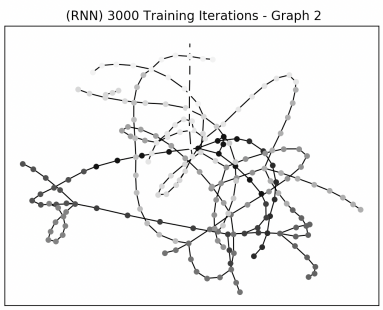
GraphRNN - 2200 iterations Sample 1



GraphRNN - 2200 iterations Sample 2



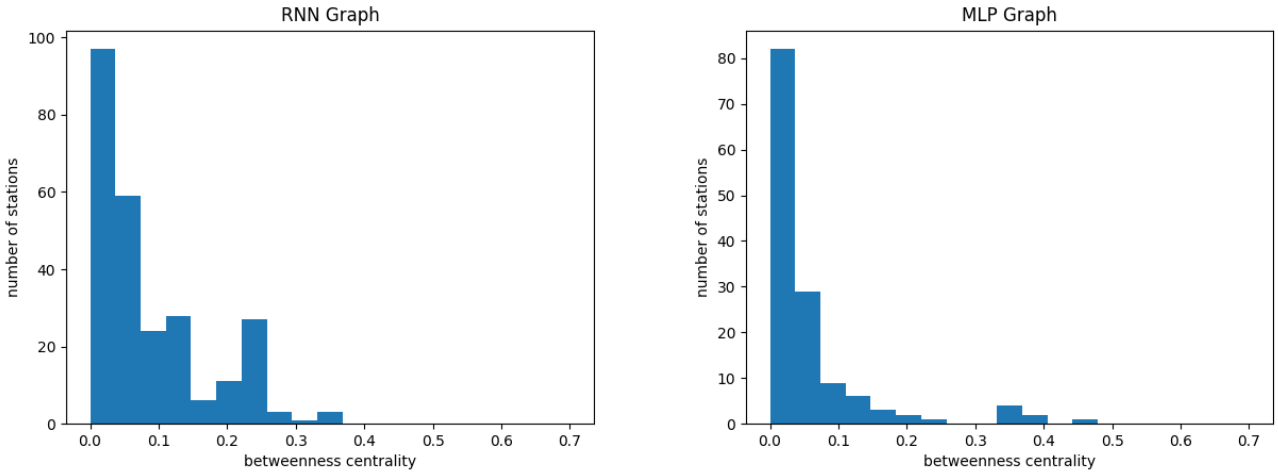
GraphRNN - 3000 iterations Sample 1



GraphRNN - 3000 iterations Sample 2

Figure 7: GraphRNN generated graphs at various iterations, clustered into communities by Louvain

larger portion of the nodes have a betweenness centrality of 0 or close to 0, which means that that node is not often part of the shorted path from any node in the graph to any other node in the graph using the shortest possible path. This indicates that the overall structure of the generated graphs might be slightly less centrally clustered than the actual subway networks, with more outreaching trails in the generated graphs.



GraphRNN RNN - 3000 iterations Sample 1

GraphRNN MLP - 3000 iterations Sample 1

Figure 8: Betweenness centrality histograms of generated graphs

6 Conclusion

With the graph theory-based network analysis, we were able to parse out several interesting statistics about subway networks in developed cities, and discern and analyze trends in statistics based on various features of the city. For example, positive correlation between the population and area of the city and the number of stations in the city was an interesting and relevant find. Additionally, looking at metrics such as the average degree, the diameter, the average number of transfers per trip, the normalized average distance between nodes, and betweenness centrality gave us further insight into relevant ranges for these metrics and how to structure graphs such that the metrics fit within normal ranges for subway networks.

Next, we learned from Louvain clustering about general community structure that is derived from the line structure of the subway networks. In the real graphs, these communities tend to be segments of lines, or clusters of connected stations near the center of the city. Practically, these clusters could be used to set up maintenance centers for the subway network; after all, we would want to minimize the number of maintenance hubs to optimize on cost, but at the same time, these centers should be as close as possible to the stations they service. Our goal was to be able to generate new subway network skeletons for developing cities, so by finding optimal clusters in our generated graphs, we can propose locations for metro maintenance centers as well.

Finally, in terms of average degree, clustering patterns, betweenness centrality histograms, and diameter, the generated graphs are very similar to the real graphs, which is an indication that the generated graphs may be valid skeletons on which to base the structures of new subway networks. The graphs generated using GraphRNN were much more successful overall than the MLP graphs, as they had clean chain-like structures without short (1-5 node) offshoots or branching, which was a common phenomenon in the MLP graphs. However with the RNN graphs, we see a central, well-connected cluster of stations from which lines of stations branch out, which matches real subways more closely. The GraphRNN generated graphs are interesting in that they tend to have more loops than the MLP generated graphs, as well as the real graphs. This may be an interesting new design to put into use, as it improves connectivity and reduces average path length, making the network more efficient.

7 Works Cited

Sharav, N., Bekhor, S. Shiftan, Y. Urban Rail Transit (2018) 4: 23. https://doi.org/10.1007/s40864-018-0075-7

Wu, X., Tse, C., Dong, H., Ho, I., Lau, F. (2016). A Network Analysis of World's Metro Systems. 2016 International Symposium on Nonlinear Theory and Its Applications 606-609.
www.ieice.org/nolta/symposium/archive/2016/articles/1065.pdf

You, J., Ying, R., Ren, X., Hamilton, W., Leskovec, J. (2018). GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models.

Zhang, J., Xu, X., Hong, L., Wang, S., Fei, Q. (2011). Networked analysis of the Shanghai subway network, in China. Physica A: Statistical Mechanics and Its Applications, 390(23-24), 4562–4570. doi: 10.1016/j.physa.2011.06.022