Network Analysis of the Paris and Tokyo Subway Systems

by

Travis Schauer

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in the

Mathematics

Program

YOUNGSTOWN STATE UNIVERSITY

April, 2023

Network Analysis of the Paris and Tokyo Subway Systems

Travis Schauer

I hereby release this thesis to the public. I understand that this thesis will be made available from the OhioLINK ETD Center and the Maag Library Circulation Desk for public access. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

Travis Schauer, Student Date

Approvals:

Dr. Alexis Byers, Thesis Advisor Date

Dr. Alicia Prieto, Committee Member Date

Dr. Hayden Julius, Committee Member Date

Dr. Salvatore A. Sanders, Dean of Graduate Studies Date

Abstract

This thesis applies network analysis to the subway systems of Paris and Tokyo, with the goal of improving subway efficiency and sustainability through insights gained from network science. The paper covers the mathematical preliminaries of graph theory and various topics within network science, focusing particularly on centrality measures. Using networks based on the Paris and Tokyo subway systems, the paper finds that the chromatic number of both networks is 3. Comparisons are made using various centrality scores and descriptive metrics, the practical implications of the findings are discussed. The results suggest that the insights gained from network analysis can be used to enhance the efficiency and sustainability of subway systems, offering practical applications for public transportation management.

Table Of Contents

Chapter 1

Background

1.1 Introduction

Graph theory has notable connections to transportation problems. A famous example dates back to the 18th century with the *Königsberg Bridge Problem*, in which there were seven bridges that connected four land areas divided by the River Pregel in Königsberg, Prussia. [4] The citizens of Königsberg created a game in which they attempted to cross each of the seven bridges exactly once. No one was able to find such a route, and it remained unsolved until it was brought to the attention of the mathematician Leonhard Euler. In August of 1735, Euler published a paper proving it to be impossible to cross over each bridge exactly once. Figure 1.1 illustrates the Königsberg Bridge Problem. It was with this problem, a transportation problem, that the field of graph theory began. [17] Since then, graph theory has been constructed around its many applications, especially in the contemporary as the language of modern network science. One such network problem concerns the design and efficiency of subway systems in large cities.

Japan opened its first subway line open in Tokyo in 1927. The first expansion occurred in 1933, with rapid development in the 1950s and 1960s. [2] The Tokyo Metro has since developed into a system that moves 6.84 million passengers in a single day. [23] The Tokyo subway system consists of two subway systems: a private company, called - the Tokyo Metro, and the public Tokyo Metropolitan Bureau of Transportation (Toei Subway). These lines and stations combine to form the seventh longest subway system in the world with a yearly ridership of 3.334 billion in 2013. [21][20] The Tokyo subway is often cited as the busiest subway system in the world and serves as an excellent example of modern public transit, connecting more than 35 million citizens to Japan's capital. [20]

The Paris Metro opened with its first line in July of 1900, increasing to ten lines by 1920. Three more lines

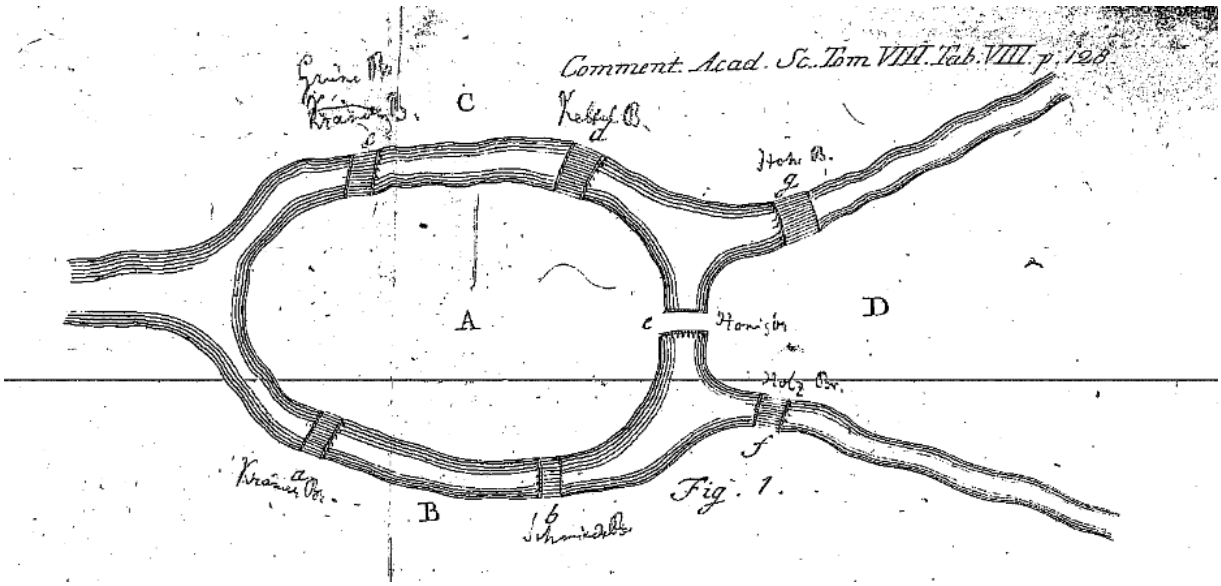


Figure 1.1: The problem as drawn in Euler's original publication. [7]

were completed by the 1930s, and since then, the Paris Metro has developed into the second busiest metro in Europe with a daily ridership of approximately 5 million. [19][14] Like the Tokyo subway system, the Paris Metro serves as an archetypal example of modern public transit.

In this paper, we will examine both the Tokyo subway and the Paris subway using graph theory. Given the expansive size of the two systems, we utilize techniques of network science. The following section discusses the subtle differences between a graph and a network. It will also explore the research that has been conducted with subway systems and their relation to graph theory.

1.2 Subway Systems in Network Science

There are a handful of ways to define both a graph and a network. More rigorous definitions will be provided in Section 2.1, but for now, a **graph** can be thought of as a set of points connected together by a collection of edges - see Figure 1.2. A graph can be "expanded" to a **network**; that is, a network is a massive collection of points and edges, often too large to draw by hand. Mathematically, a graph and a network are the same and so the terms can be interchanged. Generally, the term *graph* is used in a pure math setting while the term *network* is used in an applied math setting. In this paper, for either term, we will be considering only simple graphs (or networks); that is, graphs that do not have any loops (a point connected to itself) or multiple edges between the same two points.

While subway systems may not be the first type of system to come to the mind of network scientists, there has been some significant research conducted. In 2012, Sybil Derrible published "Network Centrality

of Metro Systems," in which she analyzed 32 subway systems including Tokyo and Paris. The first goal was to analyze each example with *network centrality* to determine the most central stations, laying the foundation for designing a system that will help distribute the flow of passengers. Similarly, Derrible analyzed the *betweenness centrality* of each station, but not all systems. The analysis only considered stations that were not end lines. [5]

In 2018, Mohieddin Jafari and Sayed Mohammad Fakhar published "Network Centrality Analysis of Tehran Urban and Suburban Railway System." This paper examines the *degree centrality, closeness centrality, betweenness centrality*, and *eccentricity centrality* of the Tehran Urban and Suburban Railway System (TUSRS). The most novel part of this paper is the data visualization techniques used. The network is presented with two concentric circles, sorted clockwise according to closeness centrality. The size of each node represents betweenness centrality and the color represents degree centrality. This network is shown in Figure 1.2. The paper also presents a matrix plot of four centrality measures for each of the eight lines in the TUSRS. [11]

"A Network Analysis of World's Metro Systems" by Xingtang Wu, Chi K. Tse, Hairong Dong, Ivan W. H. Ho and Francis C. M. Lau was published in 2016. In this paper, various network properties of Beijing, Hong King, London, Paris, and Tokyo are analyzed. This paper proposes multiple metrics: *average station density*, *average station coverage*, and *average station load*. The network properties, mainly network efficiency, are compared in respect to these proposed metrics. A definition of *network efficiency* specifically for subway systems is also proposed; see [Section 2.3, [10]].

In 2019, "An Analysis of Subway Networks using Graph Theory and Graph Generation with GraphRNN" was published by Kuhan Jeyapragasan, Gita Krishna, and Yash Maniyar. This paper describes two separate but related projects. They first compared Vienna, Tokyo, London, Washington D.C., Chicago, Madrid, Beijing, New Delhi, and Santiago using multiple metrics. The metrics the paper uses are similar to what has already been discussed, but with a couple of interesting additions. The paper discusses two metrics related to transfers, and more interestingly, a metric related to *communities*. The second project uses GraphRNN to make predictions about the overall structure of subway systems. GraphRNN uses an auto-regressive model that takes in various networks (or graphs) as a training set. With respect to the nodes, the program generates the least complex edge formations for a given network. The goal of this analysis is to create an optimal starting point for creating subway systems in new cities. [13]

This paper aims to contribute to the field of research that examines subway systems by comparing two such systems using established methods. The study begins with a discussion of the necessary mathematical concepts, followed by a description of the data collection process for creating the Tokyo and Paris networks, and concludes with an examination of the two networks.

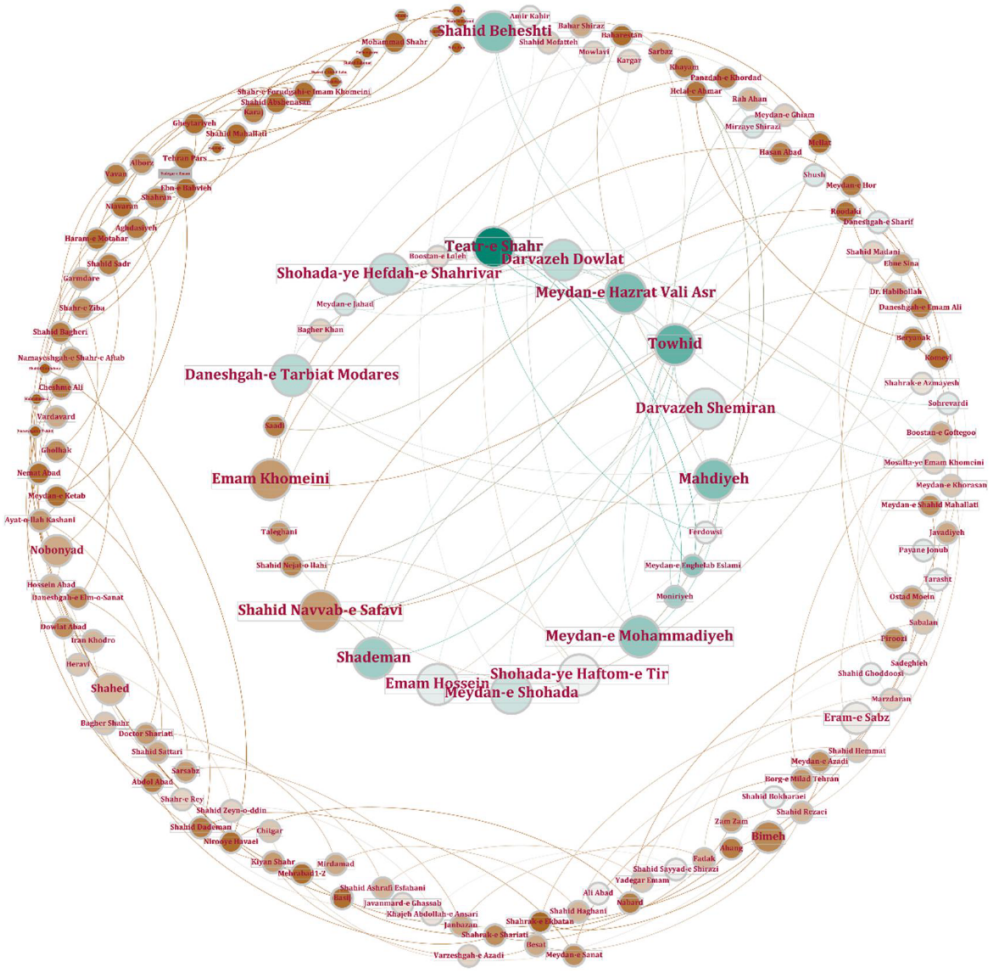


Figure 1.2: A visualization of the TUSRS network. [11]

Chapter 2

Mathematical Preliminaries

2.1 Graph Theory

This section will introduce the necessary definitions and theorems from graph theory to discuss network science. The first goal is to understand the fundamental concept of a graph. In this chapter, if no citation is given, assume that *A First Course in Graph Theory* by Gary Chartrand and Ping Zhang was used. [4]

Definition 2.1.1. A **graph** G consists of finite nonempty sets V of objects called *vertices* and a set E of 2-element subsets of V called *edges*.

The set $V(G)$ is called the vertex set of G and $E(G)$ is called the edge set of G.

Definition 2.1.2. The number of vertices in G, denoted $|V(G)|$, is called the **order** of G.

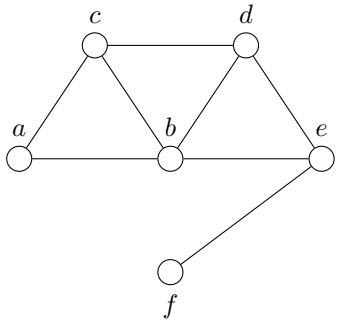
Definition 2.1.3. The number of edges in G, denoted $|E(G)|$, is called the **size** of G.

Definition 2.1.4. Suppose that uv is an edge of a graph G, then u and v are said to be **adjacent** in G.

Definition 2.1.5. Suppose that e is the edge between vertices uv of a graph G, then u and v are referred to as **neighbors** and the vertex u and the edge e are said to be **incident** with each other.

Definition 2.1.6. The **degree of a vertex** v in a graph G is the number of edges incident with v and is denoted by $\deg_G v$ or just $\deg v$ if it is clear what graph is being discussed.

Example 2.1.7. Consider the following graph G:



Then we have the following:

$$V(G) = \{a, b, c, d, e, f\} \text{ and } |V(G)| = 6$$
$$E(G) = \{ac, ab, bc, bd, be, cd, de, ef\} \text{ and } |E(G)| = 8$$
$$\deg(b) = 4$$

After gaining a basic understanding of graphs, we can explore more advanced concepts. One approach to describing graphs is by encoding their information in a matrix with real-valued entries, using techniques from linear algebra and matrix theory to reveal their properties. The systematic examination of graph theory from a linear algebraic perspective is known as *Spectral Graph Theory*.

Definition 2.1.8. Let G be a nontrivial graph of order n with $V(G) = \{v_1, ..., v_n\}$. Then the **adjacency matrix of** G is an $n \times n$ matrix with the entry in the i^{th} row and j^{th} column defined by:

$$a_{ij} = \begin{cases} 0 & \text{if } v_i v_j \notin E(G) \\ 1 & \text{if } v_i v_j \in E(G) \end{cases} \qquad [3]$$

Definition 2.1.9. If A is an adjacency matrix of a graph G and D is the diagonal matrix of G (where the diagonal entries $d_{ii} = \deg(i)$ and 0's elsewhere), then the **Laplacian matrix**, L, is defined as $L = D - A$. [3]

Definition 2.1.10. A graph G is called **regular** if every vertex has the same degree.

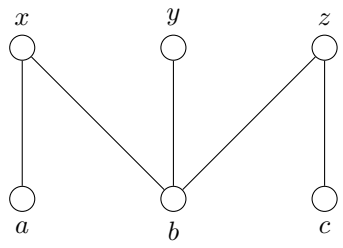
Definition 2.1.11. Let u and v be vertices of some graph G. A $u - v$ **walk** in G is a sequence of vertices in G beginning with u and ending at v.

Definition 2.1.12. Let u and v be vertices of some graph G. A $u - v$ **path** is a $u - v$ walk in G such that no vertices are repeated.

Definition 2.1.13. If a graph G contains a path between every vertex, then G is said to be **connected**.

Definition 2.1.14. A graph G is a **bipartite graph** if $V(G)$ can be partitioned into two subsets U and W such that every edge in G joins a vertex of U and a vertex of W.

Example 2.1.15. Consider the following graph G:



We can see that G is a bipartite graph where $U = \{x, y, z\}$ and $W = \{a, b, c\}$.

Theorem 2.1.16. Let G be a graph of order n. The following are equivalent:

(a) G is bipartite.

(b) If A is the adjacency matrix of G and μ is a nonzero eigenvalue of A, then $-\mu$ is also an eigenvalue of A.

(c) The characteristic polynomial of G, denoted $\phi_G(\lambda)$, is a polynomial in λ^2.

(d) If $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ is the set of eigenvalues of A, then $\sum_{i=1}^{n} \lambda_i^{2t-1} = 0$ for all positive integers t. [3]

Usually, it is easy to determine if a graph is bipartite by visual inspection. If the graph gets large, it can still be done, but is more challenging. A property that is not so easy to determine by inspection is diameter, especially when the graph is large. Fortunately, upper bounds for diameter can also be obtained by studying the adjacency matrix

Definition 2.1.17. A path is called a **triad** if it forms a cycle of length three in a graph. [15]
Note: We define **length** as the number of edges in a path

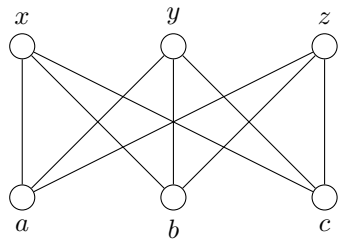
Definition 2.1.18. Let G be a connected graph of order n and let u and v be two vertices of G. The **distance** between u and v is the smallest length of any $u - v$ path in G and is denoted by $d(u, v)$.

Definition 2.1.19. The greatest distance between any two vertices of a connected graph G is called the **diameter** of G and is denoted by $\text{diam}(G)$.

Theorem 2.1.20. The diameter of a connected graph G is less than the number of distinct eigenvalues. [3]

The next example will delve into walks, paths, and diameter. It is important to note that the chosen graphs in Examples 2.1.15 and 2.1.21 meet the conditions of Theorem 2.1.16. This theorem involves a lot of linear algebra, the aim of this section is to illustrate concepts in graph theory. Therefore, we will not delve into the details of the theorem.

Example 2.1.21. Consider the following *complete bipartite* graph, $K_{3,3}$:



We would like to determine the diameter of this graph. First, let us look at a $x - c$ walk, such as the sequence $w = (x, b, y, a, z, c)$. Not only is this a walk, but since no vertices are repeated, w is a path. Now, there are many $x - c$ paths that we can select with this graph, but there is a unique smallest path between x and c of length 1; namely, the path $u = (x, c)$. Thus the distance between x and c is $d(x, c) = 1$. However, consider the distance between nodes x and y. While there are multiple shortest paths (of the same length), one such path is $k = (x, a, y)$ and so $d(x, y) = 2$. By inspection, it seems that the distance between every pair of nodes is either 1 or 2. We might then infer that $\text{diam}(K_{3,3}) = 2$. Let us verify this using Theorem 2.1.20.

The adjacency matrix for the graph above is presented below. The rows and columns are labeled with the same names as the nodes, namely x, y, z, a, b, and c, in left-to-right and top-to-bottom order, respectively.

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Now, we will use `https://matrixcalc.org/` to calculate the following eigenvectors and eigenvalues:

$$v_1 = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } \lambda_1 = 0 \qquad v_2 = \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } \lambda_2 = 0$$

$$v_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \end{bmatrix} \text{ and } \lambda_3 = 0 \qquad v_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} \text{ and } \lambda_4 = 0$$

$$v_5 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \text{ and } \lambda_5 = 3 \qquad v_6 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \text{ and } \lambda_6 = -3$$

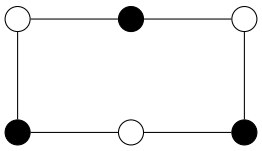
The number of distinct eigenvalues is 2 and so, by Theorem 2.1.20, diam$(K_{3,3}) \leq 2$. In the work above, we found two vertices of distance 2, and so we can conclude that diam$(K_{3,3}) = 2$.

The *Total Coloring Conjecture* is a popular unsolved problem in mathematics. While the adjacency matrix might not be able to help us with total colorings (with current knowledge), it can certainly help us put an upper bound for proper colorings.

Definition 2.1.22. A **proper coloring** (or coloring) of a graph G is an assignment of colors to the vertices of G, one color to each vertex, such that adjacent vertices are colored differently.

Definition 2.1.23. The smallest number of colors in any coloring of a graph G is called the **chromatic number** and is denoted by $\chi(G)$.

Example 2.1.24. Consider the following coloring of a graph G:



Since every vertex needs a different color than its neighbor, then we need at least two colors to color G. Therefore, we know that we are looking at the smallest number of colors for a coloring of G. Hence, $\chi(G) = 2$.

Let $\lambda_{\max}(G)$ denote the largest eigenvalue of the adjacency matrix of G. This eigenvalue places an upper bound on the chromatic number of a graph. The problem of determining the chromatic number of a graph is computationally difficult. However, Theorem 2.1.25 offers a helpful computation to narrow down our options for the chromatic number.

Theorem 2.1.25. For every graph G, $\chi(G) \leq 1 + \lambda_{\max}(G)$. [3]

2.2 Network Science

As discussed in the introduction, the terms *graph* and *network* are often used interchangeably in the literature. In this paper, we define a **network** as a graph that is too large to be drawn and analyzed by hand. It is worth noting that all the concepts presented in the previous section also apply to networks. For the remainder of this paper, we will use the word **network** instead of graph and **node** instead of vertex, while edges will remain the same.

The purpose of this section is to explore network science concepts that go beyond what is typically covered in graph theory resources. These topics will be the primary focus of our analysis later on, and thus, we will not provide many examples. We will begin this section by discussing a particular type of network that frequently appears.

Definition 2.2.1. A **directed network** is a network in which each edge has a direction, pointing from one node to another. [15]

This will be working with *undirected* networks, and so, this section will focus specifically on analysis for such networks.

Definition 2.2.2. Consider an undirected network G of size m and order n. The **average degree** of G is given by:

$$c = \frac{2m}{n}. \qquad [15]$$

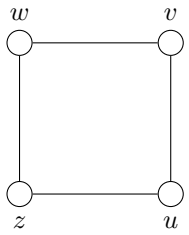
Examining the average degree is perhaps the simplest metric that can be analyzed relatively quickly from a network. While simple, this metric tends to be quite useful. We will see this idea come up a couple times. The average degree calculation allows us to compare the number edges to the number of nodes.

Definition 2.2.3. The **clustering coefficient** is the ratio of triads to the paths of length two in a network, given by:

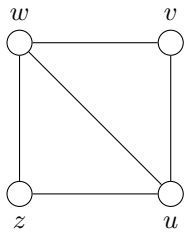
$$C = \frac{\text{(number of triads)}}{\text{(number of paths of length two)}}. \qquad [15]$$

The clustering coefficient is a metric used to analyze *transitivity* in networks, based on the transitive property of relations where two nodes sharing an edge is the relation. For instance, if node v is connected to node u, and if node u is connected to node w, then we can say that v is connected to w. [15] A value of $C = 1$ means that the network is composed of cliques where all nodes are connected to each other, and has perfect transitivity. On the other hand, a value of $C = 0$ means that there are no triads. [15]

Example 2.2.4. To help understand the clustering coefficient, first we will consider the following network N:



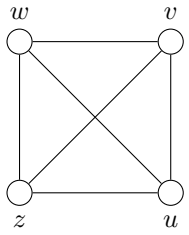
We want to calculate the clustering coefficient we can see that there are 0 triads and 4 paths of length two, those are wvu, vuz, uzw, and zwv. [1] Thus, we have $C = \frac{0}{4} = 0$. Now let us change the network slightly and call it K:



Now we can see that our paths of length two are $wvu, wuz, wuv, wzu, vwz, vuz, uwz$, and uwv. Out of those paths, only vwz and vuz do *not* form a triad. Thus, we can calculate $C = \frac{6}{8} = 0.75$.

[1] When programming an algorithm, it is common practice to count these paths twice. That is, treat wvu and uvw as distinct paths. Counting the paths once does not change the final calculation since the additional power of 2 gets cancelled in the fraction. [15]

Let us consider one more network, P:



For this problem, we now have the paths $uvw, uzw, zuv, zwv, zwu, zuw, vuw, zwv, zvw, zvu, vzw,$ and vzu. However, all these paths are also triads, and so, $C = \frac{12}{12} = 1$. Thus, P is composed of cliques, and in this case, is a complete graph.

Definition 2.2.5. Consider a network G. Let suppose that u and v are two arbitrary vertices in G, then the **average path length** is given by:

$$L = \frac{1}{n(n-1)} \sum_{u \neq v} d(u, v). \qquad [6]$$

Average path length is useful in analyzing the overall connectivity of a network. If the average path length is smaller, then the distance between any two nodes is smaller and so the network is more tightly connected. [6] This metric will be used to help compare multiple networks.

Definition 2.2.6. Let T be the total physical distance of a metro system (i.e., in kilometers) and n be the order of a network, N, then the **average distance between nodes** is given by:

$$\overline{D(N)} = \frac{T}{n}.$$

In this paper, we aim to explore the mathematical analysis of networks that are based on real-world objects. However, it's worth noting that our mathematical analysis severs the connection between the networks and their physical counterparts. To make sense of our network in the physical world, we will use the definition we've established. While there are other similar connections that we can make, this calculation works well with the reliable data that is available.

2.2.1 Centrality

When analyzing a network, one often answer the question, "What is the most important node?" or, rather, "What is the most *central* node?" Of course, the definition of "important" may change based on the objective of the analysis, and there are many types of centrality used in practice. In this section, we discuss some the most popular centrality measures in network science that are used in Chapter 4 to analyze metro systems.

The simplest type of centrality is known as **degree centrality**. It ranks nodes based on their degree, which works for both directed and undirected networks. In directed networks, both *in-degree* and *out-degree* are counted. This simple metric can be enlightening, particularly for social networks, as it easily reveals nodes with high influence and those with access to a lot of information.

As discussed earlier, networks lose the idea of physical closeness, but the shortest paths of a network can be used to discuss the closeness of nodes within a network. This is the concept behind **closeness centrality**. The goal is to determine, on average, how close each node is to the rest of the nodes in the network. It is generally found that closeness centrality and degree centrality are positively correlated. [15] The calculation is given as the inverse [2] of the mean shortest distance from a node i to every node in the network. That is, let d_{ij} be the shortest distance from node i to node j, then the closeness centrality C_i is given by:

$$C_i = \frac{n}{\sum_{j=1}^{n} d_{ij}}. \qquad [15]$$

While most centrality measures take the "central" part literally, **betweeness centrality** aims to rank nodes based on where they lie on a path between other nodes. That is, to determine which nodes in the network gain influence by the information that is passed through them. For a node i of an undirected network, we will define n_{st}^i to be the number of shortest paths from nodes s to t that pass through i and g_{st} as the total number of shortest paths from nodes s to t. Then the betweeness centrality score is given by:

$$x_i = \sum_{st} \frac{n_{st}^i}{g_{st}}. \qquad [15]$$

If n_{st}^i and g_{st} both happen to be zero, then we will let $\frac{n_{st}^i}{g_{st}} = 0$.

The most important type of centrality, and the one with the most variations, is **eigenvector centrality**. Eigenvector centrality aims to rank each node of a network based on the centrality scores of its neighbors. For example, you might have only one friend in the world, but if that friend is the president of the United States, then you yourself may be an important person. [15] Formally, eigenvector centrality awards a number of points proportional to the centrality scores of its neighbors. For this definition of eigenvector centrality, it is important to consider an undirected network, N. [3] Suppose N has n nodes; the eigenvector centrality of

[2] The inverse is considered so that nodes with a high centrality (in this case, the smallest mean shortest distance) are given a high score similar to how we view the other centrality measures. [15]

[3] Eigenvector centrality is a powerful metric, but it only works for undirected networks. A slight variation that allows for directed networks is *Katz Centrality*. This also has some flaws, and so *PageRank Centrality* and HITS can also be considered. [15]

node i can be calculated with the following formula:

$$x_i = \frac{1}{k} \sum_{j=1}^{n} A_{ij} x_j \tag{2.1}$$

where A is the adjacency matrix for the network, and k is a constant. This formula leads to the following eigenvector equation:

$$\mathbf{Ax} = k\mathbf{x} \tag{2.2}$$

where **x** is the leading eigenvector of the adjacency matrix. [15]

Since any adjacency matrix is a real-valued square matrix with positive entries, according to the *Perron-Forbenius Theorem*, there exists a unique maximal real eigenvalue with an eigenvector of strictly positive entries. [12] Therefore, Equation 2.2 illustrates one method of determining the eigenvector centrality scores of a network. That is, find the leading eigenvector and pull out the eigenvalue. This will produce a column vector such that the highest scoring nodes have a score of 1. However, determining the eigenvalues and eigenvectors of a matrix is computationally expensive (both by hand and by computer). Thus, Equation 2.1 is used to recursively calculate the score for each node, effectively creating the same column vector.

Chapter 3

Data Collection

We will now transition to discussing the networks created for this paper. It is important to contextualize the role of data collection, as it tends to be the most time-consuming element of network science research. For this project, the networks analyzed are based on the current layout for the Paris and Tokyo subway systems. While some research has been done on these systems, there does not seem to be any publicly available data sets that can be used to construct a network. This chapter will discuss not only the data collection process for these systems but also why data collection is essential to network science.

Data collection is an essential step in any network analysis. We need data in order to apply the various established techniques, which may seem apparent, but network science books rarely discuss the process. Valuable information can be determined from network analysis, but the results will only be as good as the data collected. It is important to ensure that the data is of high quality and reproducible. The data collection process will be clearly documented and a source to download the data is provided in the appendix.

In this paper, we will use both Gephi and the NetworkX package in Python to perform our analysis. Gephi requires a certain format for the data sets, and so, the data sets created in this paper are specifically tailored to Gephi. Gephi can work with limited information, but allows for the entry of any number of properties for an edge or node, for example, we might want easy access to the number of passengers a station encounters per day.

Traditionally, Gephi requires two CSV files: one with a list of nodes and one with a list of edges. The "nodes" file usually has two columns: "ID" and "Label." However, only the first column is needed for Gephi to operate. The "edges" file requires three columns: "Source", "Target", and "Type." The first two columns just consist of the node IDs indicating which nodes form an edge. The third column is used to indicate if the edge is directed or not. In our data set, we will add two more columns to the edge file: "SourceStation" and

"TargetStation" which consist of the names of the each station. This information is already captured in the "Label" column of the "nodes" file, but the reason for the inclusion of these two columns will be apparent soon.

While Gephi is an open source program, it is constantly being updated to keep up with evolving data analysis needs. Recently, Gephi has expanded its data import options beyond just CSV files. For our research, we utilized Excel Workbooks with separate sheets for both "nodes" and "edges," following the format described in the previous paragraph. This file type provided advantages during data collection, as Excel functions were readily available to facilitate the process.

After considering multiple methods, the collection method used for these data sets was to enter each piece of data manually. This allowed for easily reproducible and reliable data. The only issue with this method is the potential for human error, but if done carefully, the risk is minimal. When importing a data set to Gephi, it is possible to import only the "edges" file, and Gephi will automatically populate the nodes within its built-in data laboratory. Since the names of each station are given in our file, that data remains accessible within the software.

To collect data for this project, the following method was used. First, a workbook was created with a sheet named "edges" with the "SourceStation" and "TargetStation" columns. Then, using maps of the subway systems, the data for each connection was transcribed by hand. That is, the name of each station was entered in the respective column.

Next, a "nodes" sheet was created, and a VLOOKUP function was used to generate unique IDs for each station and to list the name of each station in the "Label" column. After this, the "Source", "Target", and "Type" columns were added to the "edges" sheet. Using an Excel function, IDs were populated for the first two columns, and the term "undirected" was added to each entry in the third column. Figure 3.1 and Figure 3.2 provide examples of these data sets. While initially time-consuming, the manual method of data collection allowed for easily reproducible and reliable data. The full data sets will be available in the appendix of the paper.

Source	Target	Type	SourceStation	TargetStation
0	1	undirected	shibuya	omote-sando
1	2	undirected	omote-sando	gaiemmae
2	3	undirected	gaiemmae	aoyama-itchome
3	4	undirected	aoyama-itchome	akasaka-mitsuke
4	5	undirected	akasaka-mitsuke	tameike-sanno
5	6	undirected	tameike-sanno	toranomon
6	7	undirected	toranomon	shimbashi
7	8	undirected	shimbashi	ginza
8	9	undirected	ginza	kyobashi

Figure 3.1: Example of the "edges" sheet in Excel for the Tokyo workbook.

Id	Label
0	shibuya
1	omote-sando
2	gaiemmae
3	aoyama-itchome
4	akasaka-mitsuke
5	tameike-sanno
6	toranomon
7	shimbashi
8	ginza
9	kyobashi

Figure 3.2: Example of the "nodes" sheet in Excel for the Tokyo workbook.

Collecting data for the Paris subway system was straightforward. As illustrated in Figure 3.3, each subway line is depicted as a horizontal line with each station placed at equal distances from one another. This allowed for swift data entry as it is easy to read each line from left to right.

Collecting data for Tokyo was slightly more challenging as there was no clear map of the subway lines like the one available for Paris. To record the connections, the most legible map that included both Tokyo Metro Lines and TOEI Lines was selected and used. This map can be seen as Figure 3.4, including the color and names of each line in the subway system. Some lines outside of the subway system were shown on the map, but they were ignored since the focus was on the subway. Each station and its connections were carefully entered into the workbook by following each line. For both Paris and Tokyo, a helpful consequence of this method was that if a station appeared on more than one line, it would appear again on the map for the other line, and these connections would be automatically added to the dataset.

Figure 3.3: The M1 line in Paris. Courtesy of `https://www.ratp.fr/en/plans-lignes/metro/1`

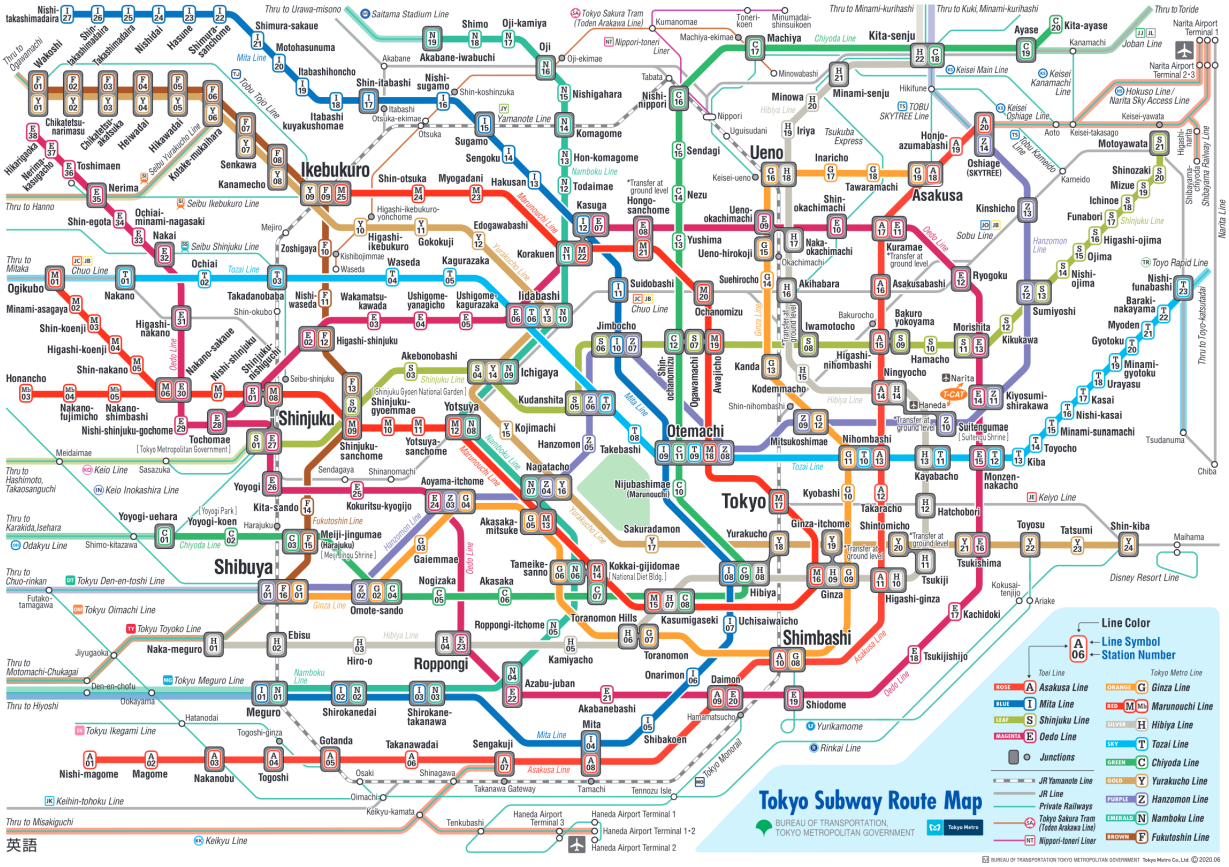


Figure 3.4: Inclusive map of the Tokyo Subway. Courtesy of `https://www.tokyometro.jp/en/subwaymap/`

Chapter 4

Network Analysis

Using network science to analyze a subway system gives the opportunity to gain valuable insights into the structure and function of complex transportation systems. Subway systems consist of stations (nodes) and routes (edges), making it possible to understand how the networks are organized and, with the right data, how they evolve over time. The analysis can also provide valuable insights on how to improve the efficiency and sustainability of the subway systems. In this chapter, we will discuss the analysis conducted on both the Paris and Tokyo subway system and then make any comparisons between the two networks.

The analysis will be conducted using both Gephi and NetworkX. While Gephi will primarily be used for visualizing the networks, some metrics will be extracted from the software. Although Gephi is a powerful tool for network science and can perform all the calculations explored in this chapter, NetworkX in Python will provide finer control and allow for easier data visualization when a chart needs to be constructed. The first section will take a deep dive into calculating metrics for the Paris network, explaining the code needed to obtain these metrics. The second section will discuss the same metrics for the Tokyo network, but without explaining the code since the methods are identical between the networks. Finally, the chapter will conclude with a comparison of the two networks.

4.1 Paris

After importing the workbook to Gephi, the network in Figure 4.1 is presented. We can see that the Paris network consists of 307 nodes with 368 edges. [1] We will reserve the main discussion of the more basic metrics for Section 4.3, but some metrics will sprinkle their way into the next sections. All of the Gephi files can be found in the appendix.

[1] NetworkX can be used to calculate both order and size, but Gephi will be preferred for the simple calculations.

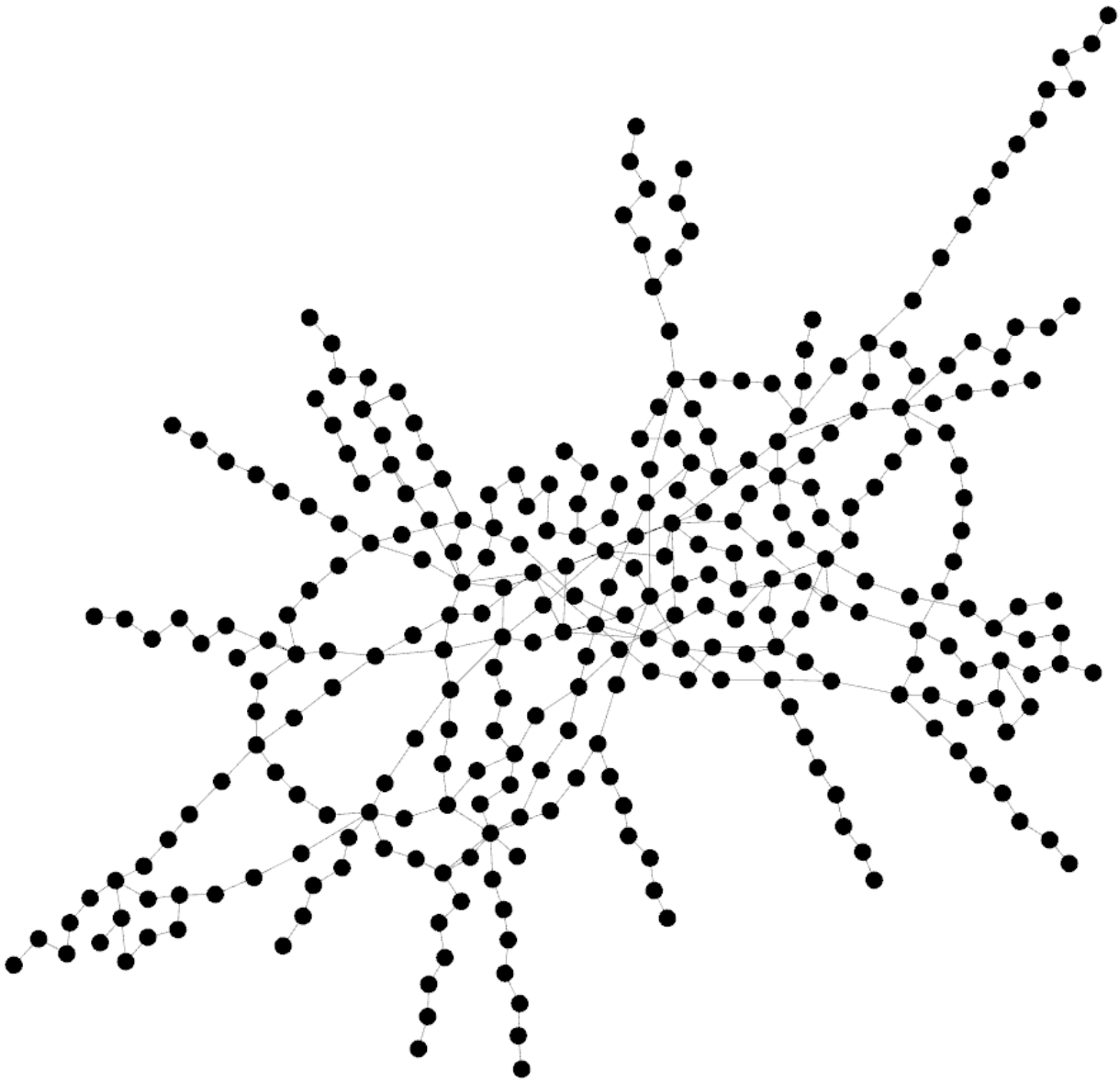


Figure 4.1: Paris network.

Now that a basic visualization of the network has been created, NetworkX will be used to gain a deeper understanding of the network. Before conducting any analysis, the data set must be imported. For the code, the following the packages will be used.

```
import networkx as nx
import numpy.linalg
import matplotlib.pyplot as plt
import numpy as np
import csv
```

Instead of using the Excel workbook generated before, the "edges" sheet from the workbook was exported as a CSV file. The following code creates a network object called `ParisNetwork`. With this object, various functions can be called from within the NetworkX package. Documentation can be found in the appendix.

```
# create network object from CSV
ParisNetwork = nx.empty_graph()
with open("Paris.csv", "r") as csvFile:
    csvReader = csv.reader(csvFile)
    next(csvReader, None)
    for row in csvReader:
        ParisNetwork.add_edge(row[0],row[1])
```

4.1.1 Centrality

Degree Centrality

One effective way to quickly grasp a network's structure is to explore its various centrality measures. Beginning with the simplest centrality score, degree centrality, can provide useful insights. Using Gephi, we find that the Paris network has an average degree of 2.397, which is relatively low but not unexpected. If we were to randomly select a sub-graph from the network, it is highly likely that this sub-graph would contain a path or cycle, indicating that there are few nodes with a degree larger than 2.

Due to the simplicity of degree centrality, we have freedom on how to display the data. Considering that our maximum degree for the Paris network is 8, it is easy to construct a bar chart showing the distribution of the degrees. There is no need to normalize our values. We can construct the bar chart with the following code.

```
# gather a list of all the degrees
degreeSequence = sorted((d for n, d in ParisNetwork.degree()), reverse=True)
# show as a bar chart
plt.bar(*np.unique(degreeSequence, return_counts=True), color='#0B645A')
plt.title("Distribution of Degrees in Paris Network")
plt.xlabel("Degree")
plt.ylabel("# of Nodes")
```

Figure 4.2 shows the bar chart produced from the code. As mentioned earlier, the chart illustrates that the majority of nodes have a degree of 2. There is also a significant number of nodes with a degree of 3 or 4, but after that, the degree accumulation drops off noticeably. In addition to the bar chart, Gephi can display the degree distribution on the network itself using a color gradient, which is shown in Figure 4.3.

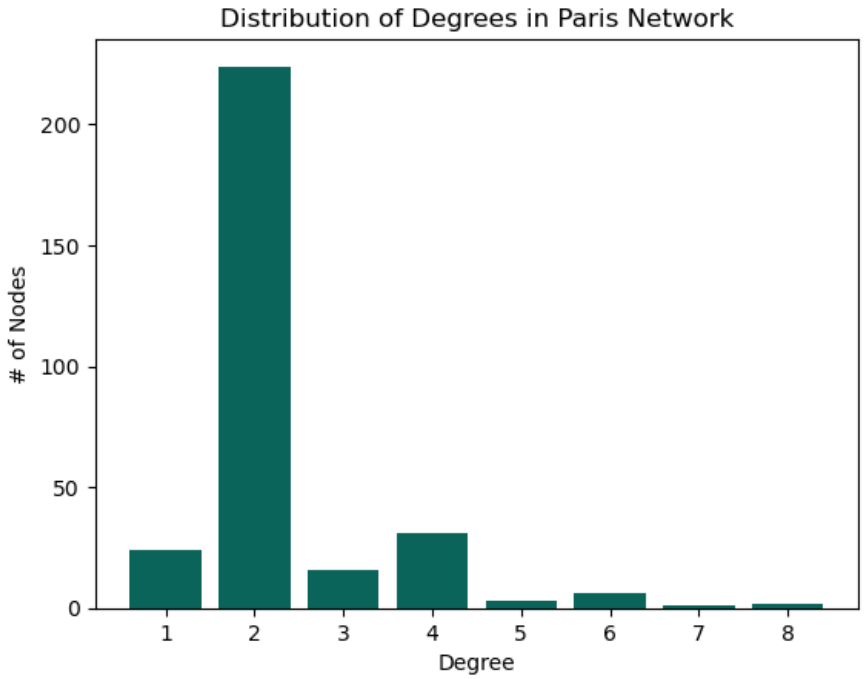


Figure 4.2: Examining this bar chart, we can see why our average degree is so low for the Paris Network.

Figure 4.3: The darker the color, the larger the degree for that node. The station with the largest degree is Châtelet which is located in medieval Paris.

Closeness Centrality

Recall that closeness centrality measures how close each node is to the rest of the nodes in a network on average. Before delving into the centrality scores, let's consider the average path length, as this will provide insight into the overall connectivity of the network. Using Gephi, we calculated the average path length of the Paris network as 11.796. Given that the network's diameter is 34, this implies that the network is quite spread out. Looking at Figure 4.1, we can see that there are many paths that end at a single node. Therefore, the large average path length should not come as a surprise.

To calculate the centrality scores, we will use a function built into NetworkX. This function returns a dictionary of centrality scores, which we will store as `closenessCentrality`. We will then extract the values from this dictionary and create a list of ordered pairs called `inside`. The histogram will be constructed from this list. The code for this process is shown below.

```
closenessCentrality = nx.closeness_centrality(ParisNetwork)
inside = [(f"{c:0.2f}",v) for v,c in closenessCentrality.items()]
#plot centrality
xs = [float(x) for x,y in inside]
plt.hist(xs, bins=10, color='#0B645A')
plt.xlim(0,0.15)
plt.title("Histogram of Closeness Centrality in Paris Network")
plt.xlabel("Centrality Score")
plt.ylabel("# of Nodes")
plt.show()
```

The result of the code is displayed in Figure 4.4. The distribution of the scores is remarkably similar to a normal distribution, which suggests a nice even spread of connectivity within the network. This observation is reinforced by analyzing Figure 4.5, where we can see that the highest scores are located in the center of the network, where we find dense clusters of nodes. The scores then gradually decrease as we move towards the network's outer edges.

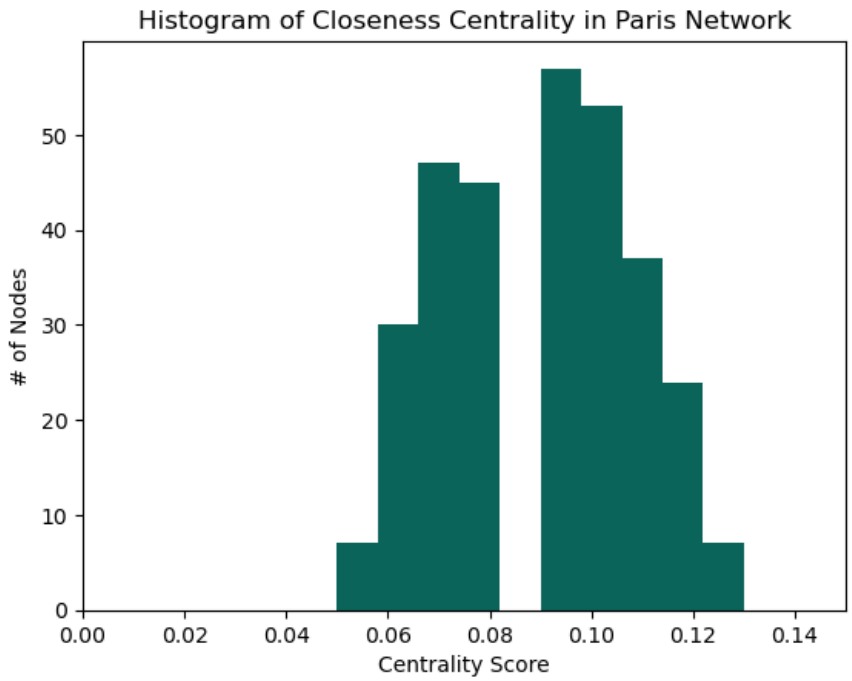


Figure 4.4: Examining this histogram, we can see a nice even distribution between the scores.

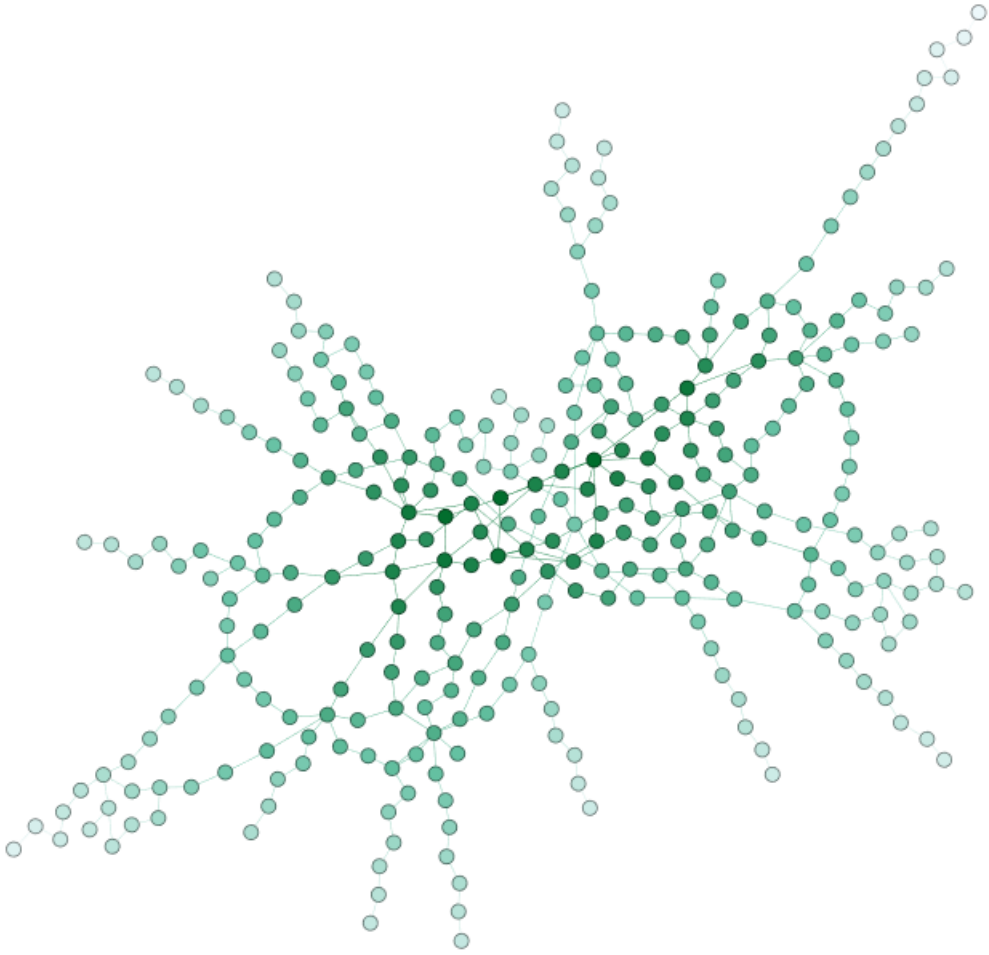


Figure 4.5: The darker the color, the better the centrality score for that node. The best scoring station is once again Châtelet, however, the scores for Pyramides and Madeleine differ by an insignificant amount.

Betweeness Centrality

Betweenness centrality, in some ways, is the most important form of centrality for analyzing a subway network. The goal of betweenness centrality is to determine which nodes act as bridges or "hubs" between different parts of the network, and therefore receive the most amount of information flowing through them. For a subway system, a high betweenness centrality score might indicate key transfer stations where passengers switch lines, or stations that serve as important connection points between different areas of the city. Additionally, the score can be useful in identifying which stations, if taken offline, might cause the largest disruption within the network.

The following code is similar to what was used for calculating closeness centrality, with a few differences in the function being called, its assignment, and the values for the histogram plot.

```
betweenessCentrality = nx.betweenness_centrality(ParisNetwork)
```

```
inside = [(f"{c:0.2f}",v) for v,c in betweenessCentrality.items()]
#plot centrality
xs = [float(x) for x,y in inside]
plt.hist(xs, bins=20, color='#0B645A')
plt.xlim(0,.35)
plt.title("Histogram of Betweeness Centrality in Paris Network")
plt.xlabel("Centrality Score")
plt.ylabel("# of Nodes")
plt.show()
```

Examining Figure 4.6 and Figure 4.7, we can see that the majority of the stations score low. However, there are a couple of stations that score significantly higher. One of these stations is Gare de Lyon. This station is cited as the third busiest station in all of France and serves as a connection to trains that leave the city. These trains head to various cities within Europe. [1] Since we have not seen this station in previous analysis, this shows how a station can become important due to its outside connections.

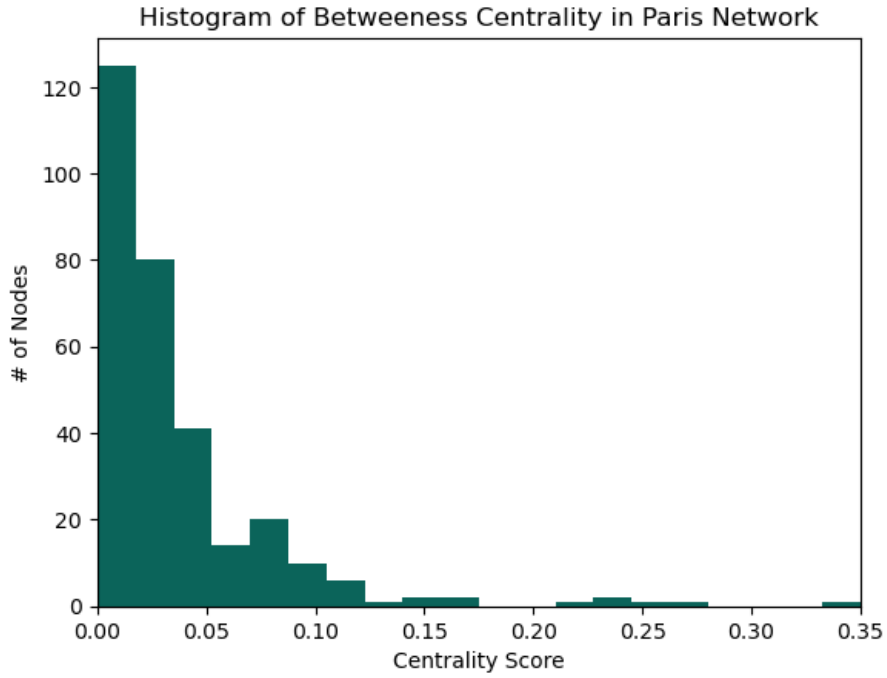


Figure 4.6: Examining this histogram, we can see that most stations are not important to the flow of information in the network, but there are a couple that are significant.

Figure 4.7: The darker the color, the better the centrality score for that node. The best scoring station is once again Châtelet, however, the scores for Gare de Lyon and Madeleine are close behind.

Eigenvector centrality

Eigenvector centrality is often considered the most important form of centrality in network analysis. Although there are multiple forms of eigenvector centrality depending on the type of network, the traditional definition of eigenvector centrality is the most appropriate for these networks. The primary goal of eigenvector centrality is to identify the most important nodes based on their associations with other important nodes. For a subway system, this can help identify not only well-connected stations, but also stations that are connected to other important stations in the system. This information can be useful to city planners in identifying stations that are likely to experience high passenger congestion. Like with betweeness centrality, the code follows closely with only a few changes.

```
eigenvectorCentrality = nx.eigenvector_centrality(ParisNetwork)
inside = [(f"{c:0.2f}",v) for v,c in eigenvectorCentrality.items()]
```

```
#graph centrality
xs = [float(x) for x,y in inside]
plt.hist(xs, bins=20, color='#0B645A')
plt.xlim(0,.35)
plt.title("Histogram of Eigenvector Centrality in Paris Network")
plt.xlabel("Centrality Score")
plt.ylabel("# of Nodes")
plt.show()
```

For our centrality scores, the same station achieved the highest score, that being Châtelet. However, for eigenvector centrality, Châtelet ranks third with Strasbourg–Saint-Denis just in front and République beating them both. This implies that eigenvector centrality is performing as expected by identifying the stations that are important based on their association. Digging into the edge file, République is shown as a massive interchange between five lines that happens to be connected to other interchanges. Figure 4.8 and Figure 4.9 provide visuals of the analysis.

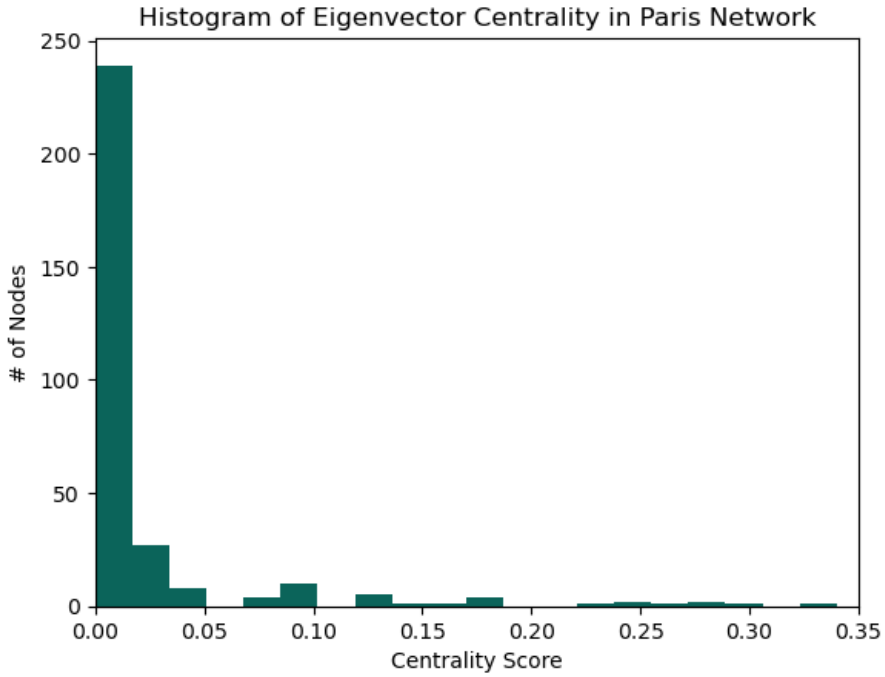


Figure 4.8: Examining this histogram, we can learn that the majority of the nodes in Paris achieve an eigenvector centrality score of near enough to zero.

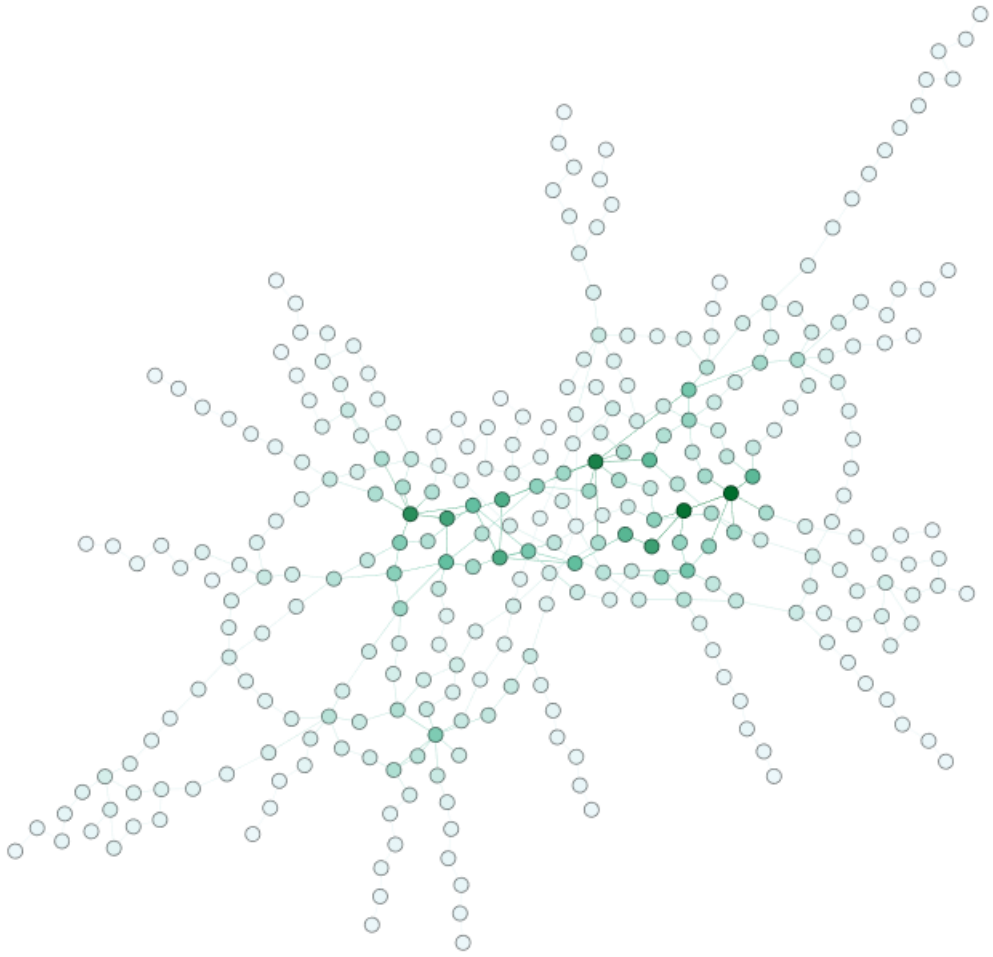


Figure 4.9: The darker the color, the better the centrality score for that node. For eigenvector centrality, République is the highest scoring station with Strasbourg–Saint-Denis as a close second, and Châtelet just behind.

4.1.2 Chromatic Number

Investigating the chromatic number of a network can provide multiple valuable insights. The most discussed application of proper colorings is scheduling. That is, if two objects cannot exist at the same time, an edge is added between them. Once the chromatic number is determined, a network can be separated into multiple sets of nodes that can all be active at the same time.

This certainly has direct applications to subway systems, as it could allow for more efficient scheduling of maintenance and repair work without causing disruptions to the entire network. Additionally, the chromatic number can provide insight into how complex the system is. If the chromatic number is high, then that indicates that there might be more transfers for passengers to make, increasing the difficulty of navigating the system.

Determining the chromatic number of a network is not an easy task. While there has been a lot of research done on the topic, there is not a one-size-fits-all answer. Using the spectrum of a graph, Theorem 2.1.25 gives an upper bound for the chromatic number. Using the *Greedy Coloring Algorithm*, we can attempt to determine the chromatic number. [8] If the supposed chromatic number falls nicely within the bounds given by Theorem 2.1.25, then the chromatic number can be safely determined.

Using NetworkX, it is possible to extract the Laplacian matrix of the network. After this, NumPy can be used to calculate the largest eigenvalue. This is easy to do with the following code.

```
L = nx.normalized_laplacian_matrix(ParisNetwork)
e = numpy.linalg.eigvals(L.A)
print("Largest Eigenvalue:",max(e))
```

Our resulting eigenvalue is 2.993397042602789, and so, by Theorem 2.1.25 we can conclude that our chromatic number, $\chi(P) \leq 1 + 2.993397042602789 \approx 3$ where P is the Paris network. Using the following code, the greedy coloring algorithm can be ran on the network.

```
chromaticNumber = nx.coloring.greedy_color(Graph, strategy="largest_first")
```

The function returns a dictionary that, for each node ID, assigns a color given by an integer. Investigating the dictionary yields that the greedy coloring algorithm was able to color the network with 3 colors. Using the following code, a new CSV was created with the just the node IDs and the integer representation of their color. After this, Excel was used to update the "nodes" file, and then the new information was imported to Gephi. Gephi was then able to output Figure 4.10, which is a coloring of the Paris network with just three colors. However, this is not enough to conclude that $\chi(P) = 3$.

According to an extension of *Brooks' Theorem*, if a network contains an odd cycle, then the chromatic number must be larger than 2. [4] The black box in Figure 4.10 highlights an odd cycle. Thus, we can conclude that $\chi(P) = 3$ since $2 < \chi(P) \leq 3$.

```
# create new csv with just node id and color
with open('parisColor.csv', 'w') as csvFile:
    csvWriter = csv.writer(csvFile)
    for key in chromaticNumber.keys():
        csvWriter.writerow([key, chromaticNumber[key]])
```

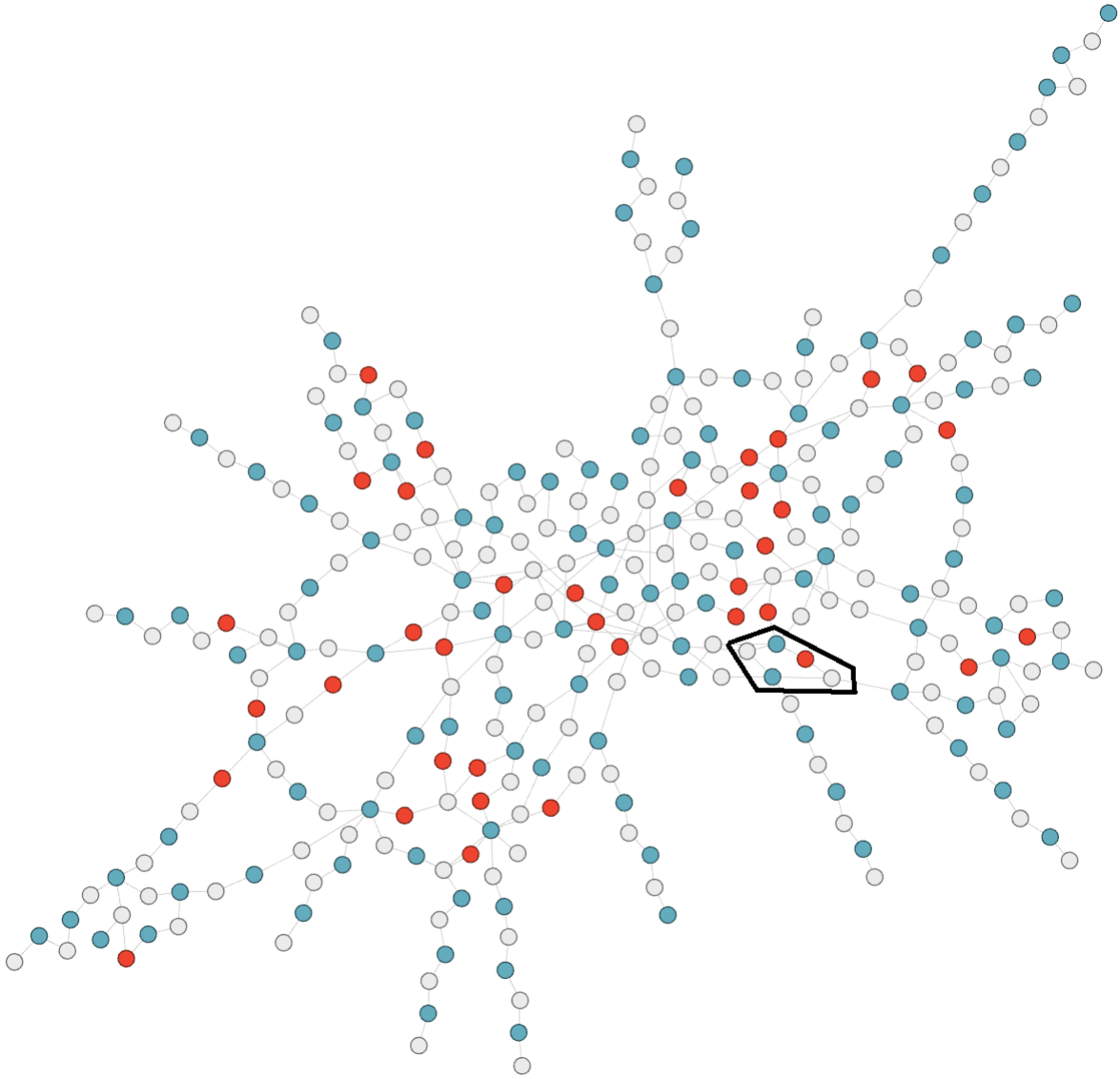


Figure 4.10: Paris network colored with three colors.

4.2 Tokyo

This section will explore the same analysis conducted to the Paris network, but with the data collected from the Tokyo subway system. The code and interpretations of the various metrics remain the same as the previous section. All the code for this section, and for the previous, can be found in the appendix. Using Gephi, a network 216 nodes and 280 edges was created. This network can be seen in Figure 4.11.

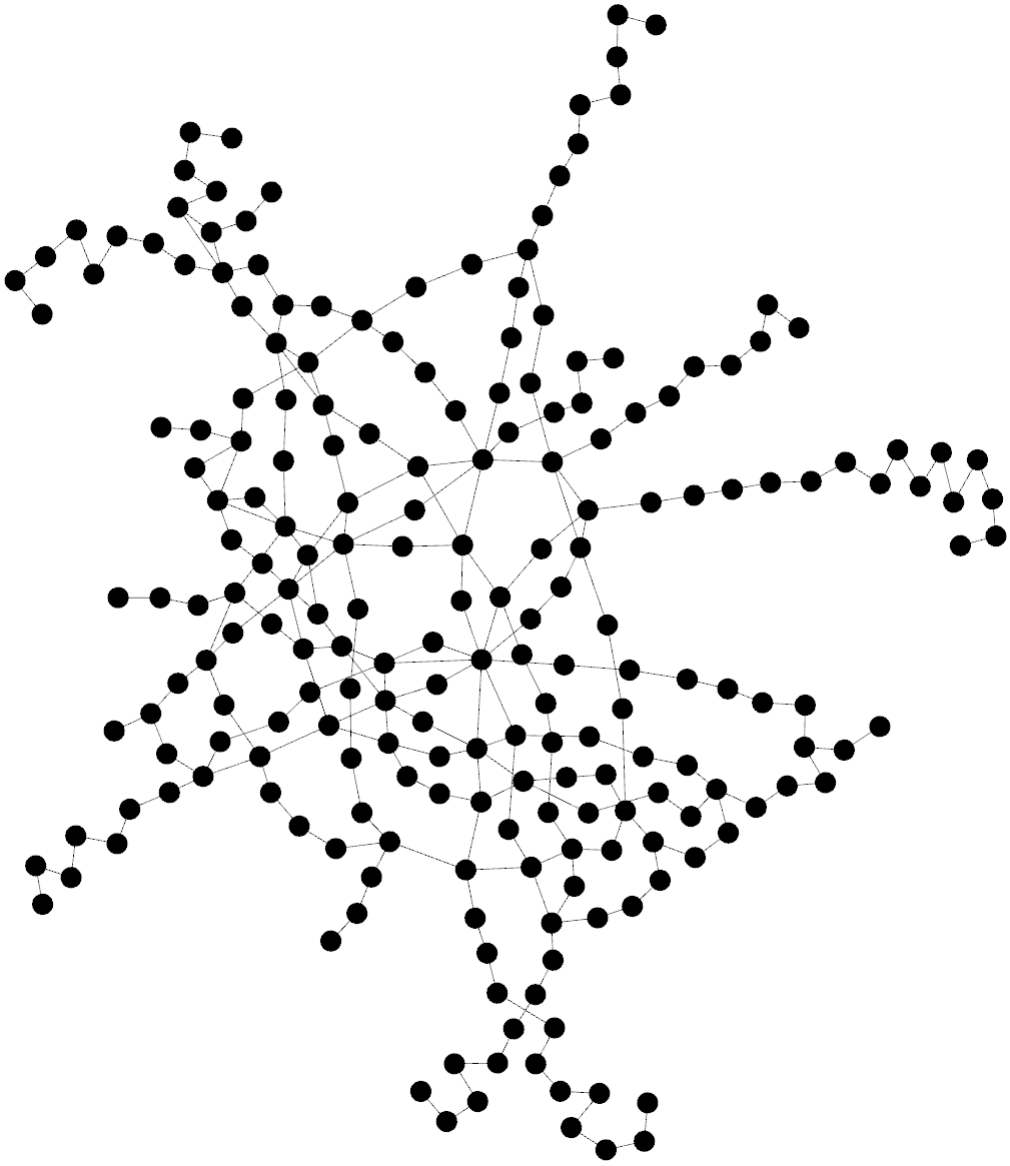


Figure 4.11: Tokyo network.

4.2.1 Centrality

Degree Centrality

The Tokyo network has an average degree of 2.593, and like before, this suggests that the network mostly consists of paths and cycles. Figure 4.12 shows the distribution of the degrees in the network. The majority of the nodes have a degree of 2, but the average is increased by the next significant concentration, which is nodes of degree 4. Figure 4.13 shows the network colored based on the degree number for each node. The station with the largest degree of 9 is Ōtemachi, which is cited as the largest station in Tokyo and serves both the Tokyo Metro and TOEI subway systems. [16]

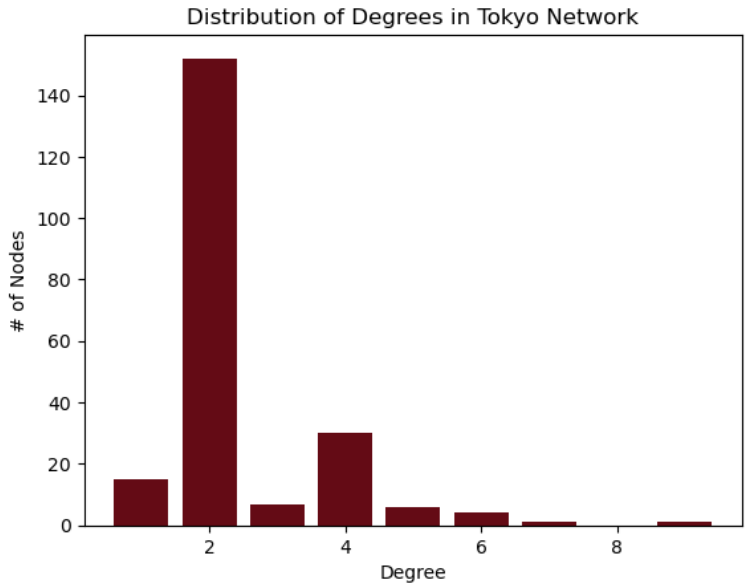


Figure 4.12: Examining this histogram, we can see that most stations have a degree of 2.

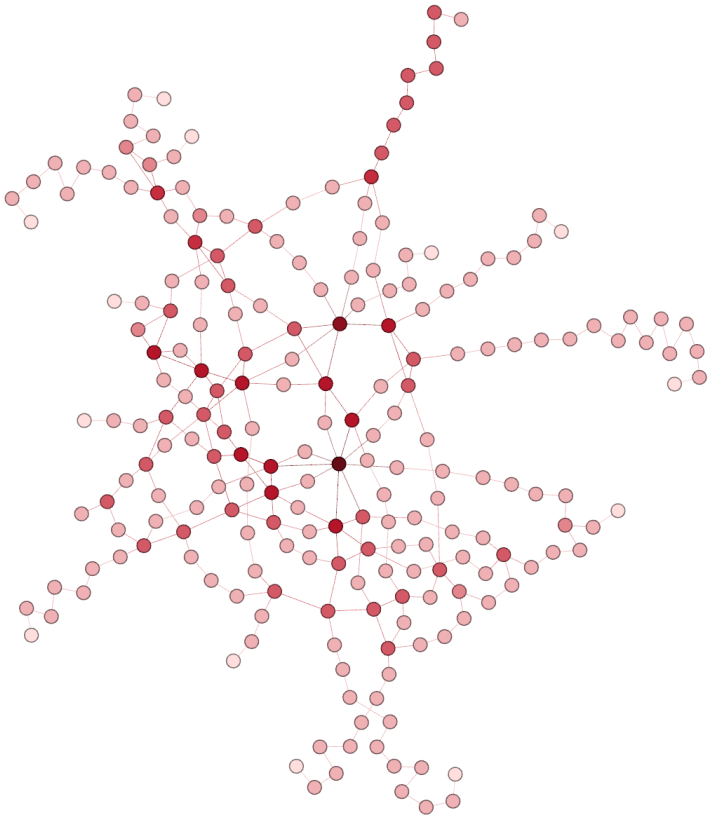


Figure 4.13: The darker the color, the better the centrality score for that node. The station with the largest degree is Ōtemachi with Iidabashi as a close second.

Closeness Centrality

Gephi calculates the average path length of the Tokyo network 10.336 with a diameter of 32. This implies that the Tokyo network is quite spread out, slightly more than the Paris network, but not by a significant amount. Figure 4.14 shows the distribution of the closeness centrality scores. The distribution here follows closely to a normal distribution, but is slight skewed to the left, implying that in general, stations are closer to each other than not. Figure 4.15 helps to visualize this on the network. Where we can see that most stations are closely connected, but the score drops as we follow paths towards the outer reaches of the network.

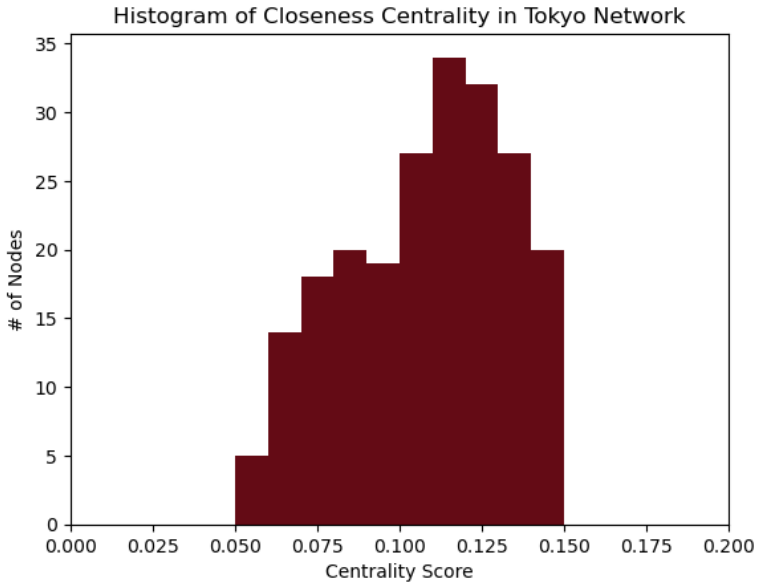


Figure 4.14: This left-skewed distribution is shifted slightly more right than what we see in Figure 4.4, suggesting that the Tokyo network is slightly closer overall.

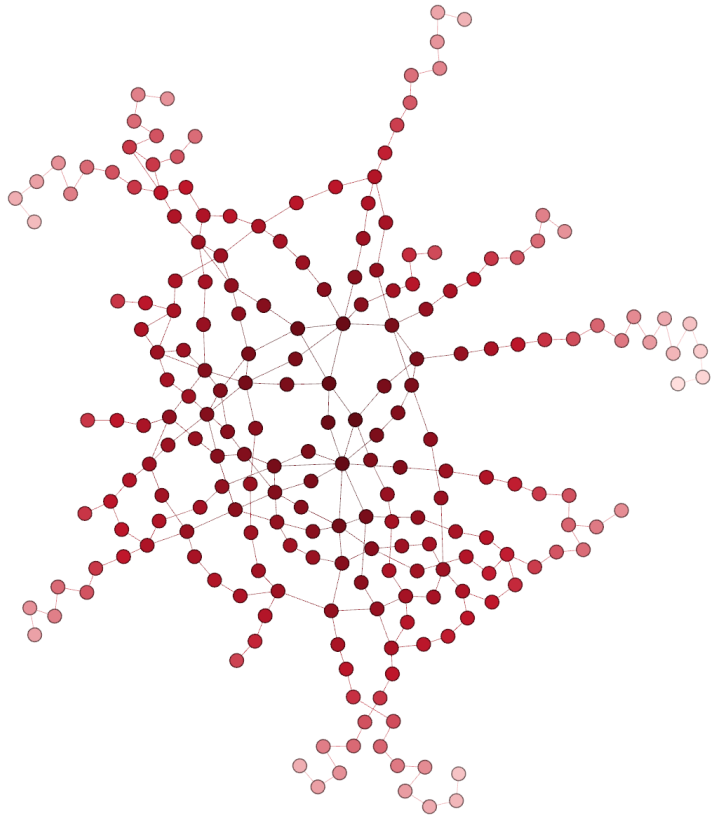


Figure 4.15: The darker the color, the better the centrality score for that node. Ōtemachi once again is the winner here, but Jimbōchō is second with Iidabashi in third.

Betweeness Centrality

Examining Figure 4.16 yields a heavily right-skewed distribution, suggesting that the majority of the stations in the network are not important for transferring information throughout the network. Unlike with what we found in the Paris network, the station with the largest betweeness centrality is once again Ōtemachi, which furthers solidify just how important this station is to the network. Figure 4.17 illustrates that Ōtemachi is physically in the center of the network, which helps see that the station is an important hub in the network.

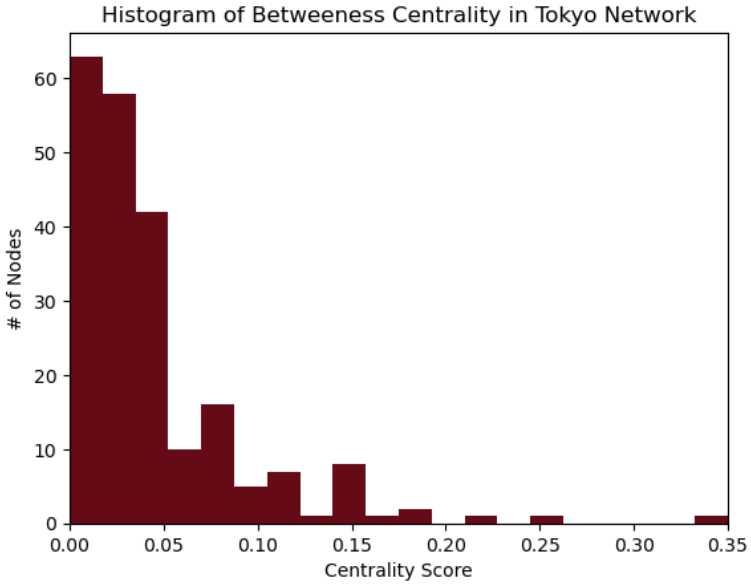


Figure 4.16: Comparing to Figure 4.6, both networks share a similar distribution with an identical range.

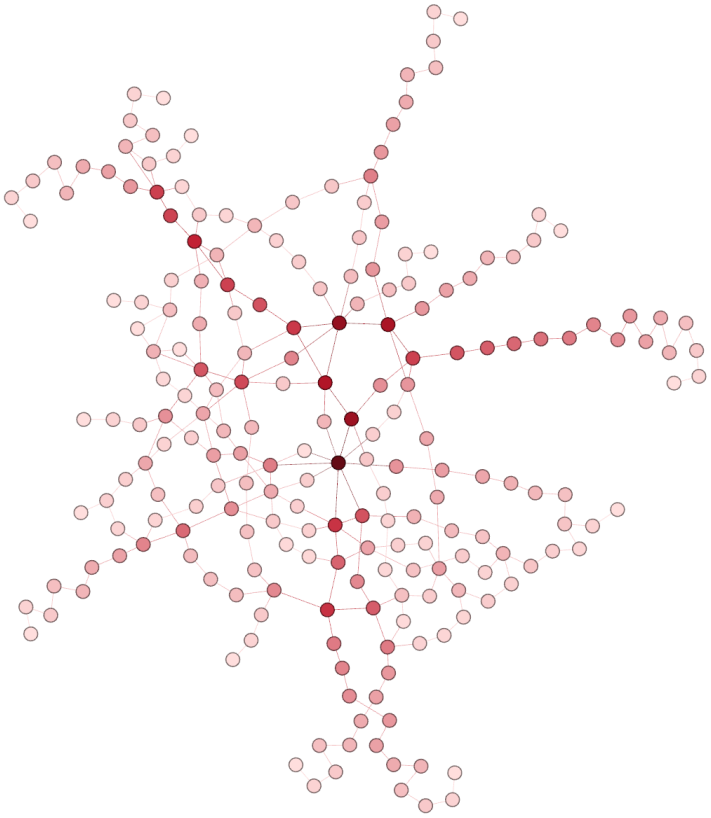


Figure 4.17: The darker the color, the better the centrality score for that node. The station with the largest score is once again Ōtemachi. Iidabashi and Jimbōchō follow in second and third respectively, but the gap between scores for Ōtemachi and Iidabashi is significant.

Eigenvector Centrality

Figure 4.18 shows the distribution of the scores for eigenvector centrality. The distribution is largely skewed to the right, however, there is a fairly large range with a large gap between the best scorer and the second best scorer. The highest scorer is once again Ōtemachi. Looking at Figure 4.19, it is shown just how influential Ōtemachi is to its neighbors. Second place goes to Jimbōchō, however, third place is awarded to a new competitor, Hibiya. Hibiya is cited as the 34^{th} busiest station in the Tokyo subways system. [9] This fact illustrates just how important it is for a station to be connected to a highly influential station when considering eigenvector centrality.

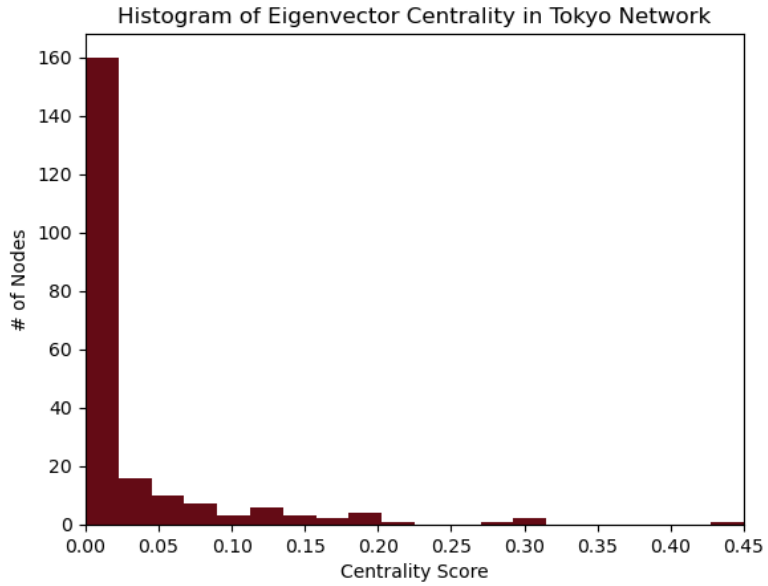


Figure 4.18: Comparing to Figure 4.8, both networks share a similar distribution, however, there is a larger ranger for the Tokyo network.

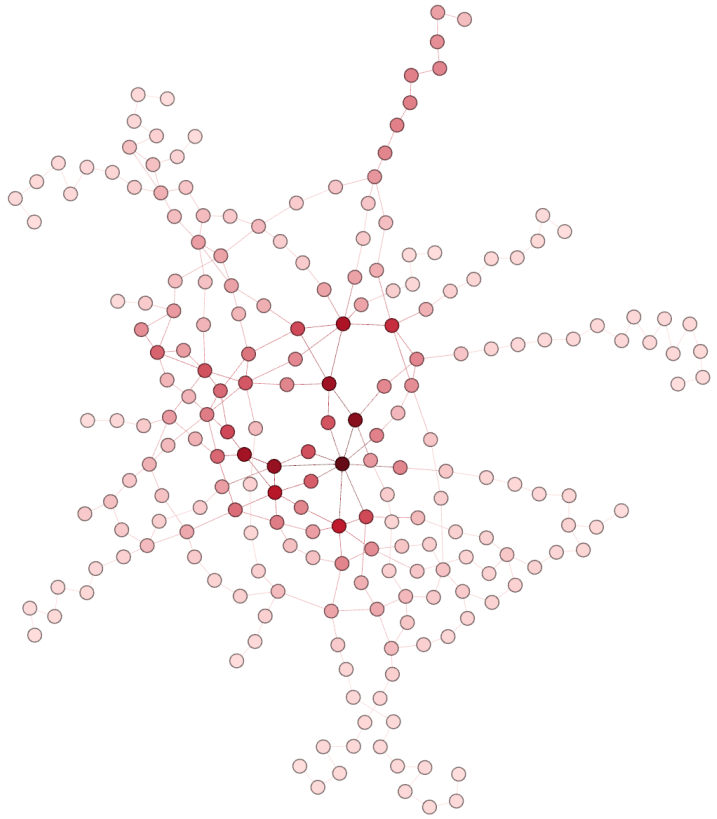


Figure 4.19: The darker the color, the better the centrality score for that node. The station with the largest score is once again Ōtemachi, with Jimbōchō and Hibiya in second and third respectively.

4.2.2 Chromatic Number

To determine the chromatic number of the Tokyo network, the same method was used as with the Paris network. Using Python, the largest eigenvalue of Tokyo's spectrum is calculated as $\lambda_{max}(T) = 1.9949031806190038$. Hence, by Theorem 2.1.25, $\chi(T) \leq \lambda_{max}(T) + 1 \approx 3$ where T is the Tokyo network. Using the greedy coloring algorithm, the network can be colored with three colors. Using Gephi, Figure 4.20 was produced which shows a coloring of the Tokyo network with just three colors.

Like with the Paris network, this is not enough to determine the chromatic number for the Tokyo network. Once again, we will make use of *Brooks' Theorem* by finding an odd cycle. The black box in Figure 4.20 highlights an odd cycle. Thus, we can conclude that $\chi(T) = 3$ since $2 < \chi(T) \leq 3$.

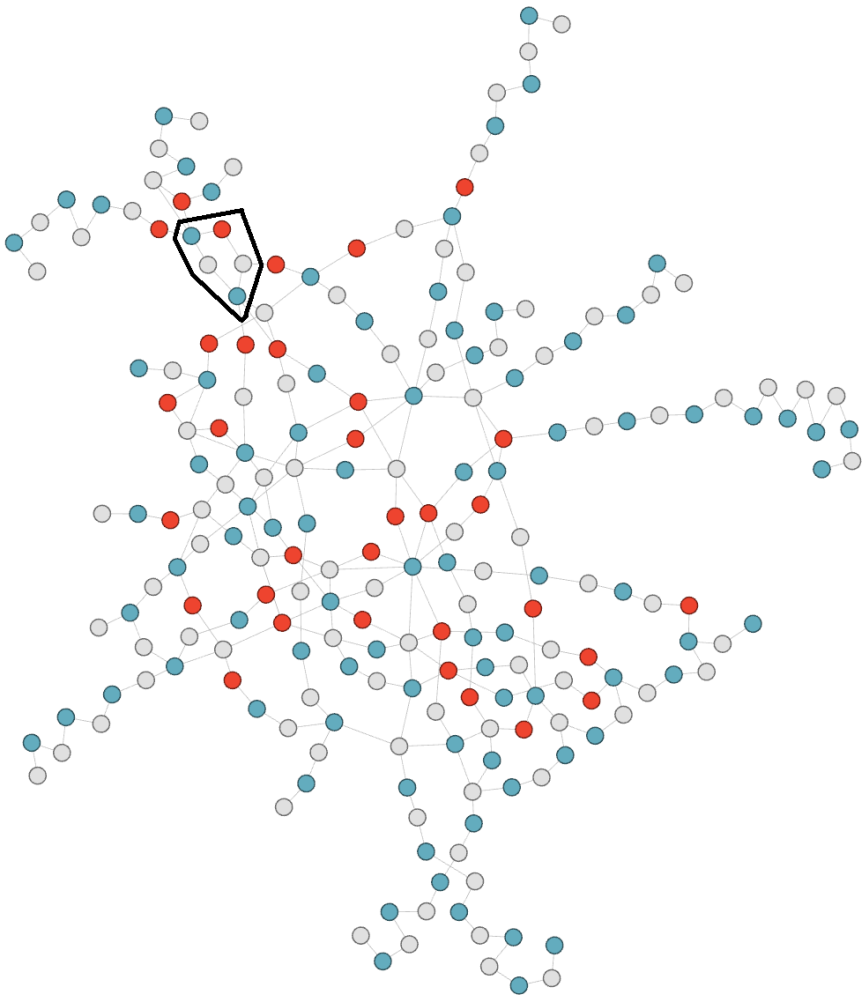


Figure 4.20: Tokyo colored with three colors.

4.3 Comparison

In this section, the paper concludes by closely comparing several metrics between the Paris and Tokyo subway networks. Since commentary on the various centrality scores was given throughout Section 4.2.1, it will be mostly omitted from this portion. A chart summarizing the metrics to be discussed is provided as Table 4.3. The only new metric introduced is the average distance between nodes, which will be discussed shortly. In each row of the table, the largest number is made bold to aid in comparisons.

	Paris	Tokyo
Order	**307**	216
Size	**368**	280
Diameter	**34**	32
Average Degree	2.397	**2.593**
Average Path Length	**11.8**	10.34
D(N) in kilometers	0.713	**1.4**
Average Clustering Coefficient	0.009	**0.034**

Table 4.1: Summary Comparison of the Paris and Tokyo networks.

Comparing the Paris and Tokyo subway networks reveals some interesting insights. First, the Paris network has a larger order and size than the Tokyo network, indicating that it has more stations, connections, and is overall a more complex network. However, the Tokyo network has a larger average degree, which suggests that it is more connected than the Paris network and that it may be easier for passengers to navigate the Tokyo subway system. Additionally, the Tokyo network has a smaller average path length than the Paris network, indicating that it takes fewer stops to travel from one station to another on average. Interestingly, while the Paris network is generally larger, its diameter differs from the Tokyo network by only 2, showing that the two networks are relatively similar in terms of maximum distance.

Figures 4.14 and 4.18 show that Tokyo has slightly stronger scores for both closeness centrality and eigenvector centrality, respectively. Additionally, even though the average clustering coefficient of both networks is very low, Tokyo has a higher score, indicating that it is slightly more connected than Paris. Taken together, this analysis suggests that despite its smaller size, Tokyo's subway system efficiently connects a large number of stations.

To calculate the average distance between stations, the total length of each subway system needs to be determined. It is cited that Paris has a length of 219 kilometers and Tokyo has a total length of 302.4 kilometers. [18][22] By dividing those lengths by the order of the respective networks, the results in Table 4.3 were obtained. It is noteworthy that the average distance between stations in the Paris network is significantly smaller than that of the Tokyo network, indicating that the Paris network may be more convenient for commuters traveling shorter distances, but the Tokyo network may serve a larger geographic area.

Although the Paris and Tokyo subway systems differ in their historical development and urban context, the network analysis conducted in this paper reveals intriguing similarities and differences between them. Despite having a larger order and size, the Paris subway system exhibits a comparable diameter and average path length to the Tokyo subway system. However, the Tokyo subway system has a higher average degree and a significantly larger average distance between stations, indicating a more decentralized and spacious layout compared to the Paris subway system. These insights contribute to a deeper understanding of the unique features and challenges of these two important subway systems, and may inform future efforts to improve their efficiency and sustainability.

Bibliography

[1] *A brief station guide of Paris Gare de Lyon*. URL: https://www.seat61.com/stations/paris-gare-de-lyon.htm.

[2] *An Overview of Japanese Subway System*. 2014. URL: http://www.jametro.or.jp/en/japan/.

[3] Dr. Alexis Byers. *Spectral Graph Theory*. class lecture, MATH 6995T ST Adv. Topics Graph Theory 2, Youngstown State University, Mar. 2022.

[4] Gary Chartrand and Ping Zhang. *A First Course In Graph Theory*. Dover, 2012. ISBN: 9780486483689.

[5] Sybil Derrible. "Network Centrality of Metro Systems". In: (). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3391279/.

[6] Sybil Derrible and Christopher Kennedy. "Network Analysis of World Subway Systems Using Updated Graph Theory". In: *Transportation Research Record* 2112.1 (2009), pp. 17–25. DOI: 10.3141/2112-03. eprint: https://doi.org/10.3141/2112-03. URL: https://doi.org/10.3141/2112-03.

[7] Leonhard Euler. "Solutio problematis ad geometriam situs pertinentis". In: (1741). URL: https://scholarlycommons.pacific.edu/euler-works/53/.

[8] *Greedy Color*. URL: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.coloring.greedy_color.html#networkx.algorithms.coloring.greedy_color.

[9] *Hibiya Station*. 2021. URL: https://www.tokyometro.jp/lang_en/station/hibiya/index.html.

[10] Xingtang Wu - Chi K. Tse - Hairong Dong - Ivan W. H. Ho and Francis C. M. Lau. "A Network Analysis of World's Metro Systems". In: (2016). URL: https://www.ieice.org/nolta/symposium/archive/2016/articles/1065.pdf.

[11] Mohieddin Jafari and Sayed Mohammad Fakhar. "Network Centrality Analysis of Tehran Urban and Suburban Railway System". In: (). URL: https://arxiv.org/ftp/arxiv/papers/1802/1802.06219.pdf.

[12] Oliver Knill. *Lecture 32: Perron Forbenius theorem*. Mar. 2011. URL: https://people.math.harvard.edu/~knill/teaching/math19b_2011/handouts/lecture34.pdf.

[13] Gita Krishna Kuhan Jeyapragasan and Yash Maniyar. "An Analysis of Subway Networks using Graph Theory and Graph Generation with GraphRNN". In: (2019). URL: http://snap.stanford.edu/class/cs224w-2019/project/26421498.pdf.

[14] *Metro in Paris*. 2022. URL: https://citytransit.uitp.org/paris/average-daily-ridership-during-a-work-day/metro.

[15] Mark Newman. *Networks*. Oxford University Press, 2018. ISBN: 0198805098.

[16] *Otemachi*. 2006. URL: https://tokyo-tokyo.com/Otemachi.htm.

[17] Teo Paoletti. *Leonard Euler's Solution to the Konigsberg Bridge Problem*. May 2011. URL: https://www.maa.org/press/periodicals/convergence/leonard-eulers-solution-to-the-konigsberg-bridge-problem.

[18] *Paris Metro*. URL: https://www.introducingparis.com/metro.

[19] *Paris Metro, Île-de-France*. Feb. 2020. URL: https://www.railway-technology.com/projects/paris-metro-france/.

[20] *The world's longest metro and subway systems*. Dec. 2013. URL: https://www.railway-technology.com/analysis/featurethe-worlds-longest-metro-and-subway-systems-4144725/.

[21] *The world's top 10 busiest metros*. Nov. 2014. URL: https://www.railway-technology.com/analysis/featurethe-worlds-top-10-busiest-metros-4433827/.

[22] *Tokyo*. URL: https://www.urbanrail.net/as/jp/tokyo/tokyo.htm.

[23] *Tokyo Metro Characteristics & Data*. 2016. URL: https://www.metro-ad.co.jp/en/characteristic/.

Appendix A

Data & Code

The datasets used in this paper can be found on Github at the following address:
`https://github.com/travisschauer/MastersThesis/tree/main/Data`

The Gephi files can be found in the same repository at the following address:
`https://github.com/travisschauer/MastersThesis`

The code used for both the analysis on the Paris network and the Tokyo network are appended starting on the next page. The files are also available on Github at the following address:
`https://github.com/travisschauer/MastersThesis/tree/main/PythonCode`

Documentation for NetworkX can be found here:
`https://networkx.org/documentation/stable/reference/index.html`

ParisAnalysis

February 27, 2023

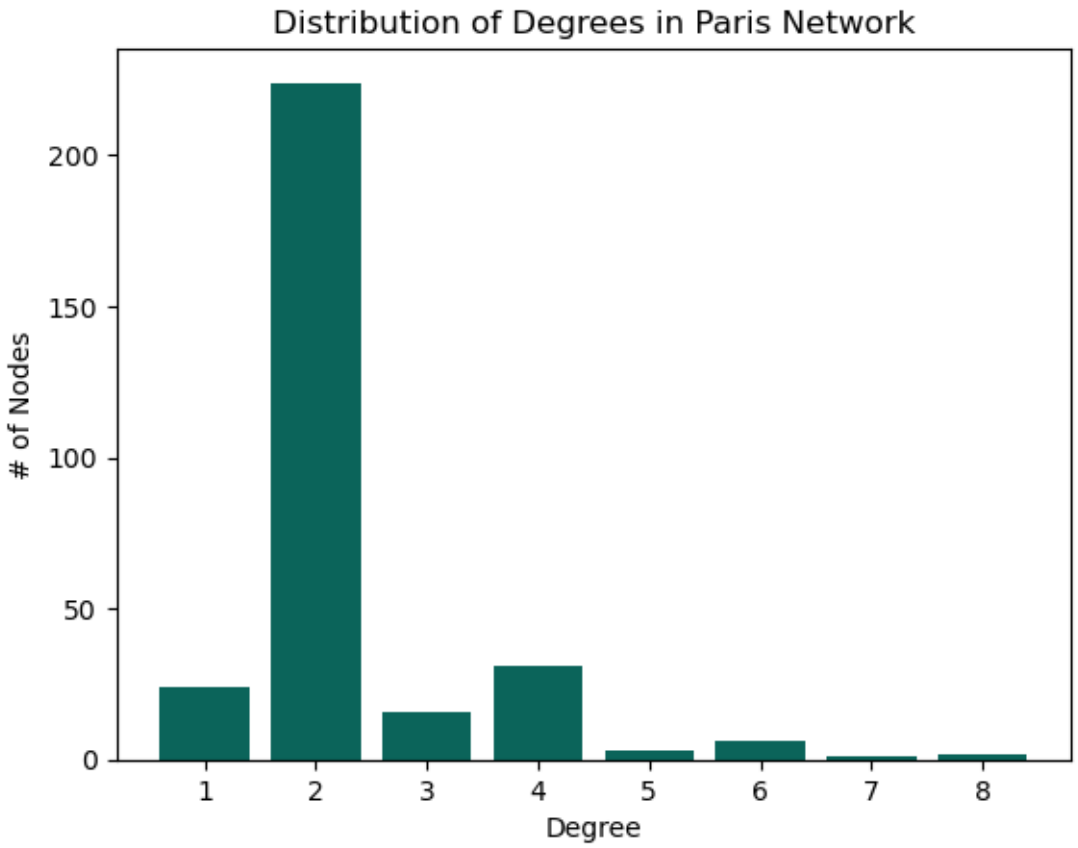
[94]:
```
import networkx as nx
import numpy.linalg
import matplotlib.pyplot as plt
import numpy as np
import csv

# create network object from CSV
ParisNetwork = nx.empty_graph()
with open("Paris.csv", "r") as csvFile:
    csvReader = csv.reader(csvFile)
    next(csvReader, None)
    for row in csvReader:
        ParisNetwork.add_edge(row[0],row[1])

#nx.draw(Graph)
```

1 Centralities

1.1 Degree Centrality

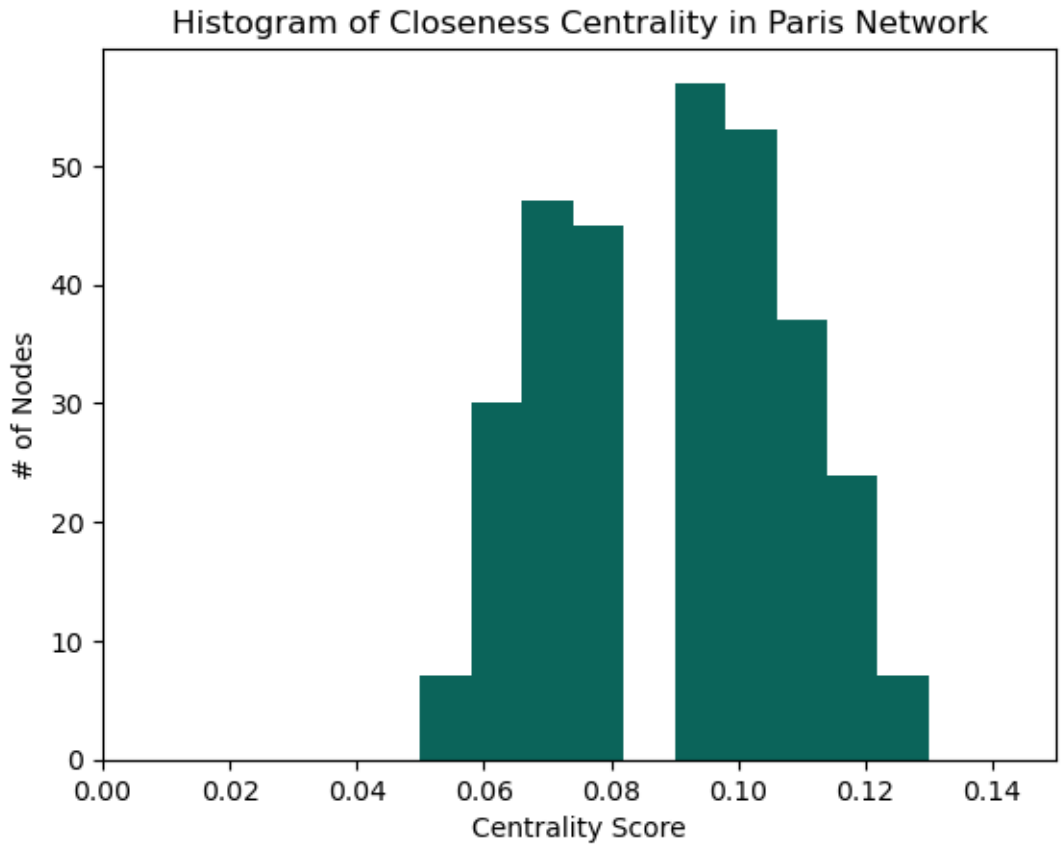
[85]:
```
# gather a list of all the degrees
degreeSequence = sorted((d for n, d in ParisNetwork.degree()), reverse=True)

# show as a bar plot
plt.bar(*np.unique(degreeSequence, return_counts=True), color='#0B645A')
plt.title("Distribution of Degrees in Paris Network")
plt.xlabel("Degree")
plt.ylabel("# of Nodes")
```

[85]:
```
Text(0, 0.5, '# of Nodes')
```



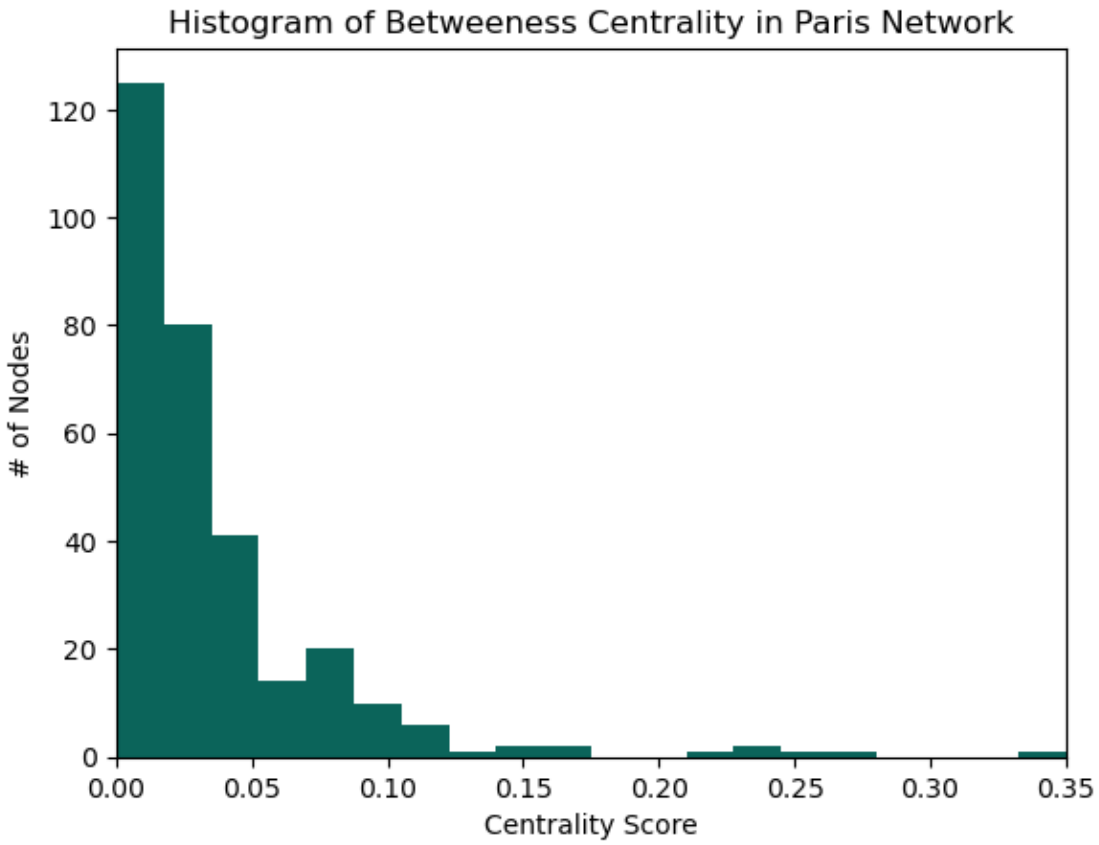
1.2 Closeness Centrality

[86]:
```
closenessCentrality = nx.closeness_centrality(ParisNetwork)
inside = [(f"{c:0.2f}",v) for v,c in closenessCentrality.items()]

#plot centrality
xs = [float(x) for x,y in inside]
plt.hist(xs, bins=10, color='#0B645A')
plt.xlim(0,0.15)
plt.title("Histogram of Closeness Centrality in Paris Network")
plt.xlabel("Centrality Score")
plt.ylabel("# of Nodes")
plt.show()
```



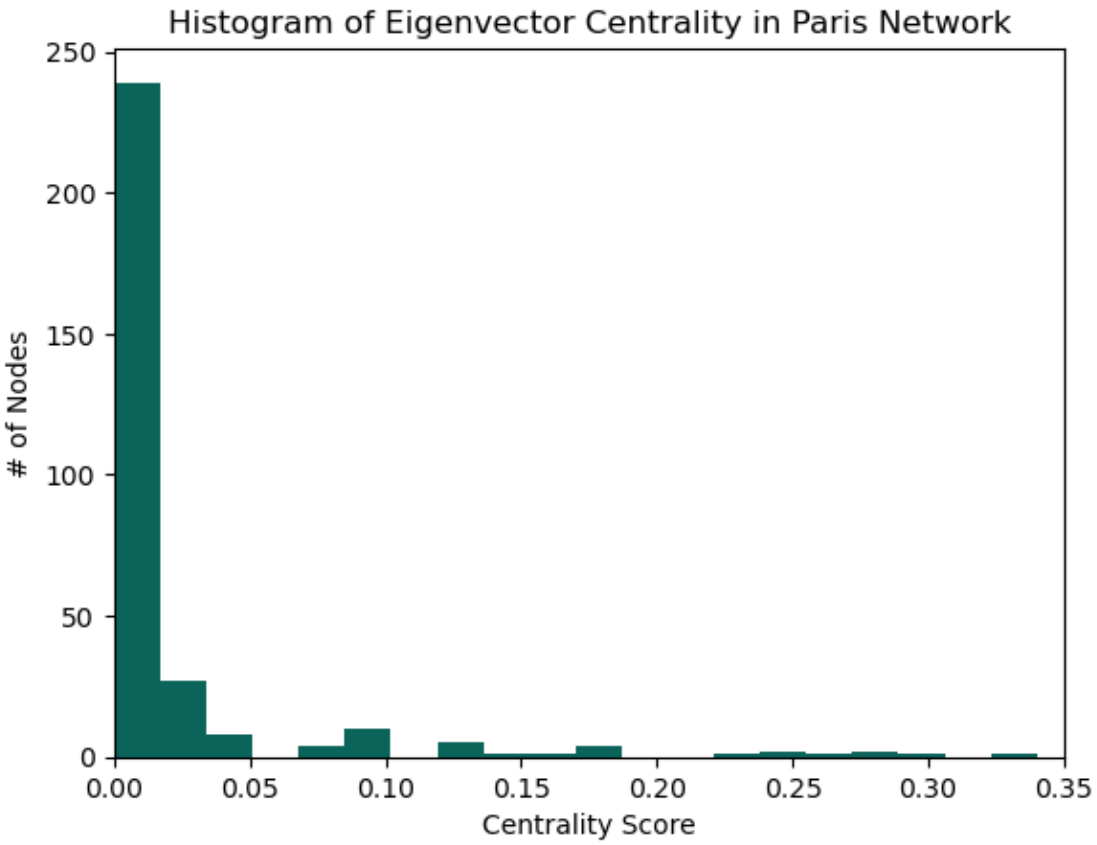
1.3 Betweeness Centrality

[87]:
```
betweenessCentrality = nx.betweenness_centrality(ParisNetwork)
inside = [(f"{c:0.2f}",v) for v,c in betweenessCentrality.items()]

#plot centrality
xs = [float(x) for x,y in inside]
plt.hist(xs, bins=20, color='#0B645A')
plt.xlim(0,.35)
plt.title("Histogram of Betweeness Centrality in Paris Network")
plt.xlabel("Centrality Score")
plt.ylabel("# of Nodes")
plt.show()
```



1.4 Eigenvector Centrality

[88]:
```
eigenvectorCentrality = nx.eigenvector_centrality(ParisNetwork)
inside = [(f"{c:0.2f}",v) for v,c in eigenvectorCentrality.items()]

#graph centrality
xs = [float(x) for x,y in inside]
plt.hist(xs, bins=20, color='#0B645A')
plt.xlim(0,.35)
plt.title("Histogram of Eigenvector Centrality in Paris Network")
plt.xlabel("Centrality Score")
plt.ylabel("# of Nodes")
plt.show()

#display centrality scores
#sorted(inside, reverse = True)
```



1.5 Calculate Eigenvalues

[98]:
```
L = nx.normalized_laplacian_matrix(ParisNetwork)
e = numpy.linalg.eigvals(L.A)
print("Largest Eigenvalue:",max(e))
```

```
Largest Eigenvalue: 1.9933970426027967
```

1.6 Chromatic Number

[152]:
```
chromaticNumber = nx.coloring.greedy_color(ParisNetwork,␣
 ↪strategy="largest_first")
#print(chromaticNumber)
```

[148]:
```
# create new csv with just node id and color
with open('parisColor.csv', 'w') as csvFile:
    csvWriter = csv.writer(csvFile)
    for key in chromaticNumber.keys():
        csvWriter.writerow([key, chromaticNumber[key]])
```

TokyoAnalysis

February 27, 2023

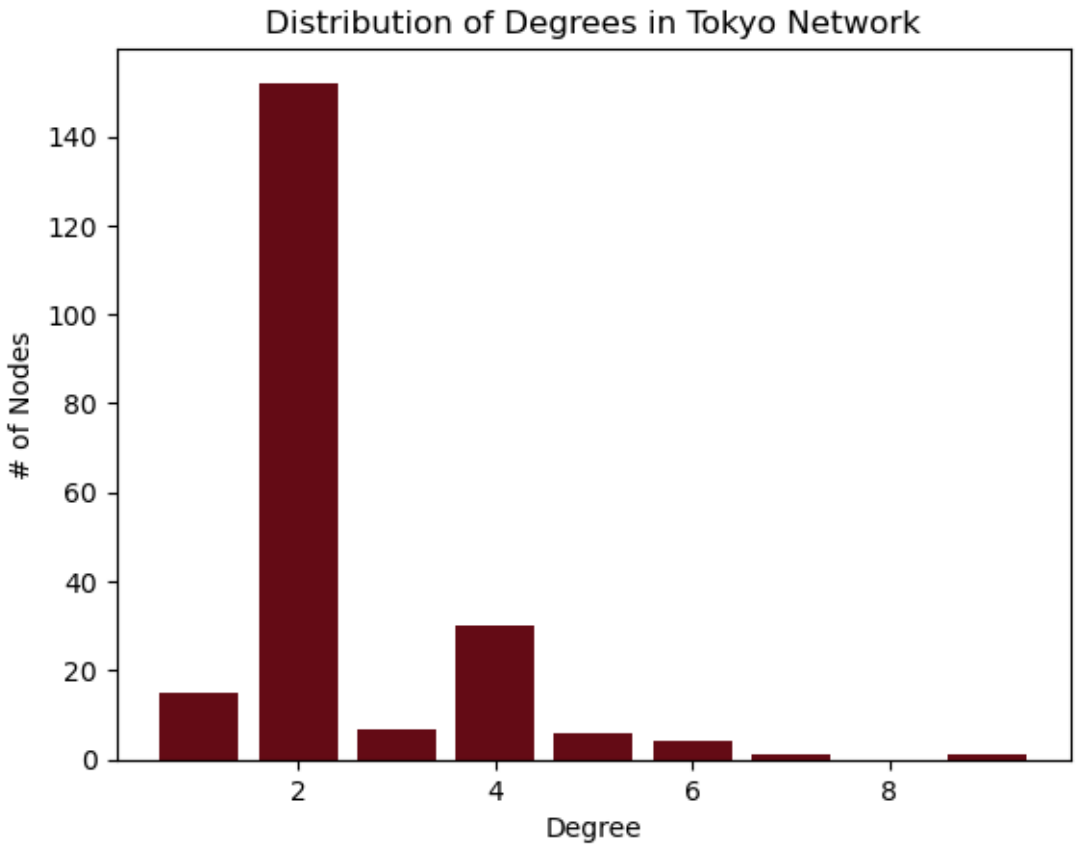
[2]:
```
import networkx as nx
import numpy.linalg
import matplotlib.pyplot as plt
import numpy as np
import csv

# create network object from CSV
ToykoNetwork = nx.empty_graph()
with open("Tokyo.csv", "r") as csvFile:
    csvReader = csv.reader(csvFile)
    next(csvReader, None)
    for row in csvReader:
        ToykoNetwork.add_edge(row[0],row[1])

#nx.draw(Graph)
```

1 Centralities

1.1 Degree Centrality

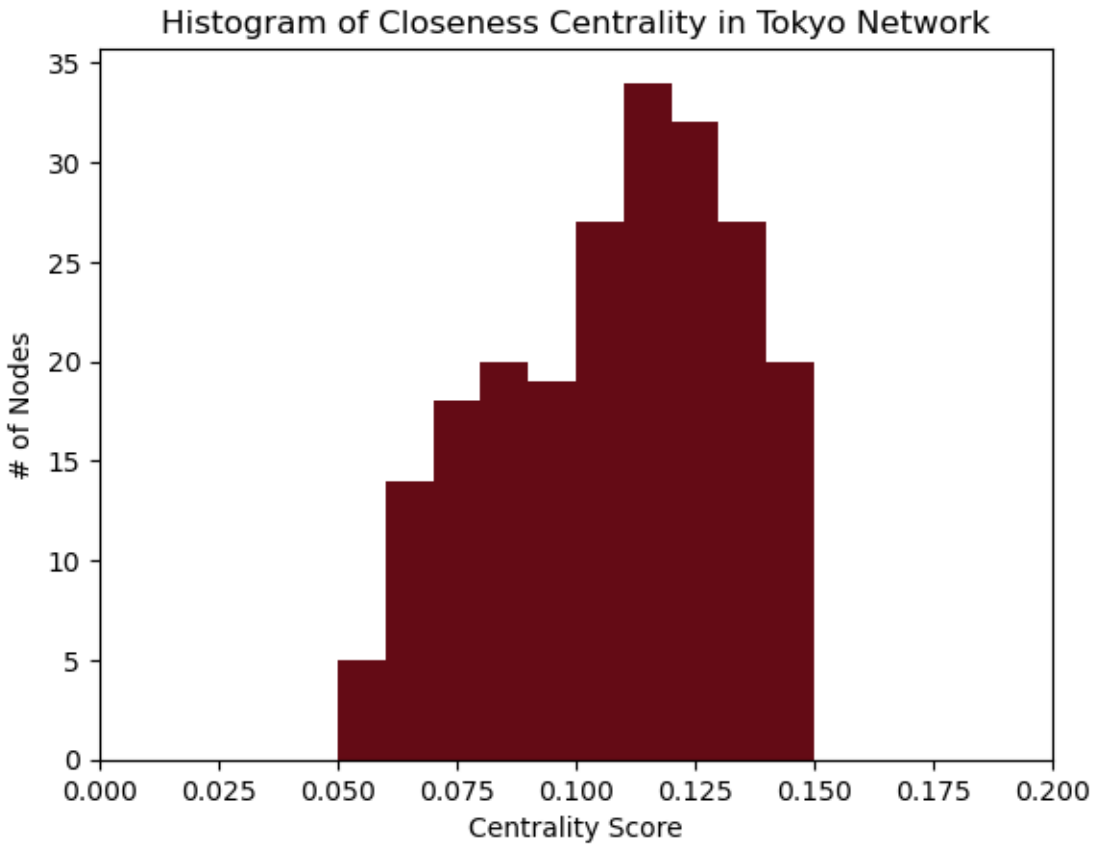
[3]:
```
# gather a list of all the degrees
degreeSequence = sorted((d for n, d in ToykoNetwork.degree()), reverse=True)

# show as a bar plot
plt.bar(*np.unique(degreeSequence, return_counts=True), color='#640B15')
plt.title("Distribution of Degrees in Tokyo Network")
plt.xlabel("Degree")
plt.ylabel("# of Nodes")
```

[3]:
```
Text(0, 0.5, '# of Nodes')
```



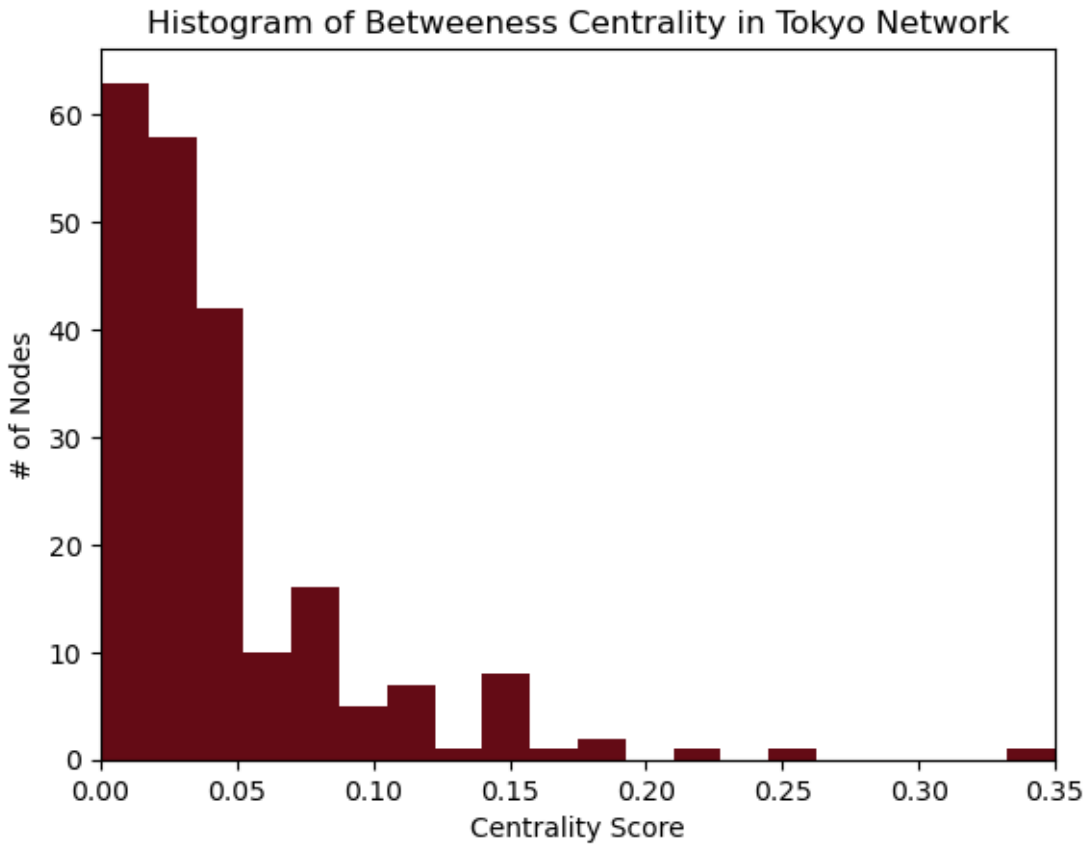
1.2 Closeness Centrality

[4]:
```
closenessCentrality = nx.closeness_centrality(ToykoNetwork)
inside = [(f"{c:0.2f}",v) for v,c in closenessCentrality.items()]

#plot centrality
xs = [float(x) for x,y in inside]
plt.hist(xs, bins=10, color='#640B15')
plt.xlim(0,0.2)
plt.title("Histogram of Closeness Centrality in Tokyo Network")
plt.xlabel("Centrality Score")
plt.ylabel("# of Nodes")
plt.show()
```



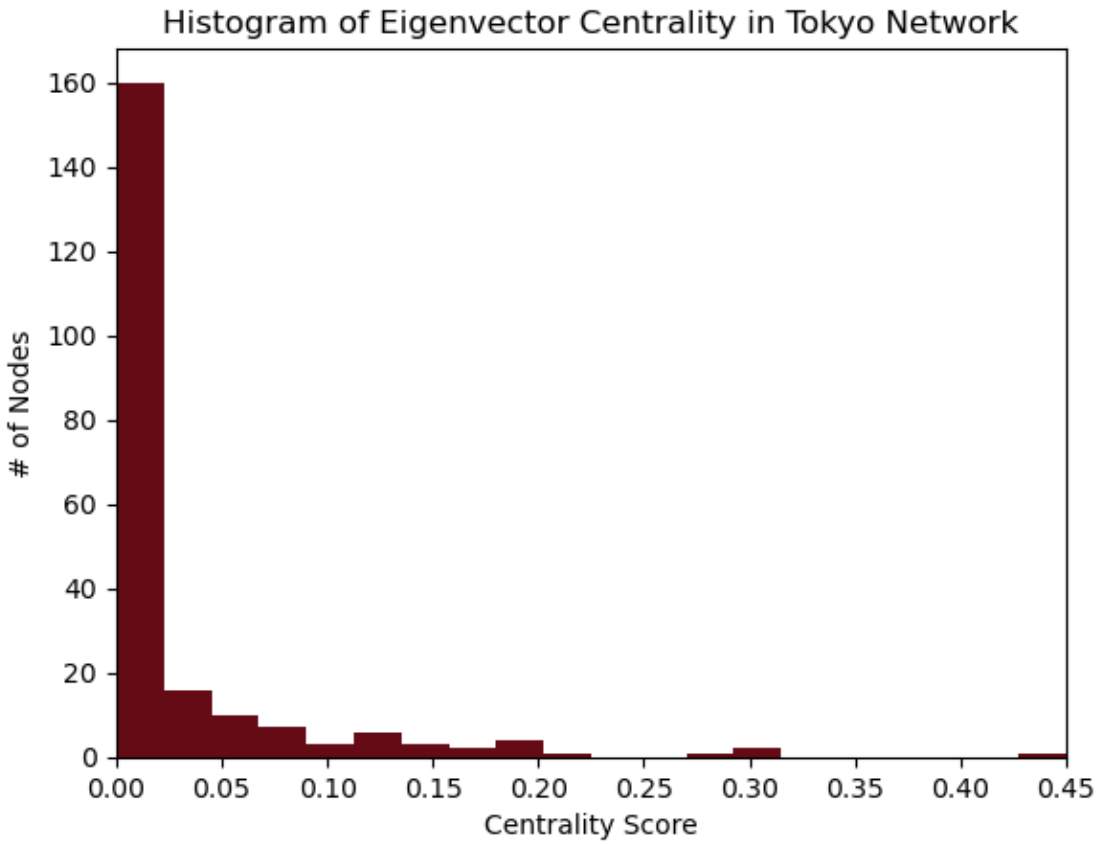
1.3 Betweeness Centrality

[4]:
```
betweenessCentrality = nx.betweenness_centrality(ToykoNetwork)
inside = [(f"{c:0.2f}",v) for v,c in betweenessCentrality.items()]

#plot centrality
xs = [float(x) for x,y in inside]
plt.hist(xs, bins=20, color='#640B15')
plt.xlim(0,.35)
plt.title("Histogram of Betweeness Centrality in Tokyo Network")
plt.xlabel("Centrality Score")
plt.ylabel("# of Nodes")
plt.show()
```



1.4 Eigenvector Centrality

[6]:
```
eigenvectorCentrality = nx.eigenvector_centrality(ToykoNetwork)
inside = [(f"{c:0.2f}",v) for v,c in eigenvectorCentrality.items()]

#graph centrality
xs = [float(x) for x,y in inside]
plt.hist(xs, bins=20, color='#640B15')
plt.xlim(0,.45)
plt.title("Histogram of Eigenvector Centrality in Tokyo Network")
plt.xlabel("Centrality Score")
plt.ylabel("# of Nodes")
plt.show()

#display centrality scores
#sorted(inside, reverse = True)
```



1.5 Calculate Eigenvalues

[15]:
```
L = nx.normalized_laplacian_matrix(ToykoNetwork)
e = numpy.linalg.eigvals(L.A)
print("Largest Eigenvalue:",max(e))
```

```
Largest Eigenvalue: (1.9949031806190038+0j)
```

```
<class 'networkx.utils.decorators.argmap'> compilation 16:4: FutureWarning:
normalized_laplacian_matrix will return a scipy.sparse array instead of a matrix
in Networkx 3.0.
```

1.6 Chromatic Number

[3]:
```
chromaticNumber = nx.coloring.greedy_color(ToykoNetwork,␣
↪strategy="largest_first")
#print(chromaticNumber)
```

[8]:
```
# create new csv with just node id and color
with open('tokyoColor.csv', 'w') as csvFile:
    csvWriter = csv.writer(csvFile)
```

```
    for key in chromaticNumber.keys():
        csvWriter.writerow([key, chromaticNumber[key]])
```