Analysis of Adjacency Matrices by Spectral Graph Theory

Venkata Renu Jeevesh Madala
2022102009

Saumya Balina
2022102069

Moida Praneeth Jain
2022101093

Shreyas Badami
2021113003

MONSOON SEMESTER 2023

1 Objective

The adjacency matrix provides a concise representation of the connections between nodes in a graph, while eigenvalues reveal important properties about the graph's structure. Through analysis and computation of eigenvalues, this project aims to gain insights into the characteristics and behavior of graph networks, such as centrality measures, connectivity, and clustering.
In a recent paper by Miller J. [9], methods to calculate the number of paths of length k between any two vertices of a graph have been explored in great depth. These methods include calculating the eigenvalues of the graph's adjacency matrix and using them to formulate the number of paths. In this project, we aim to implement these methods and analyze them by comparing them against the usual method of matrix multiplication.

2 Introduction

2.1 Eigenvalue and Eigenvector

For a matrix A, a vector which when multiplied by A gives a scalar multiple of itself is called an eigenvector of A. This scalar multiple is called the corresponding eigenvalue.

$$Ax = \lambda x$$

x is an eigenvector and λ is the corresponding eigenvalue

2.2 Graph

A graph is a set of vertices V connected by a set of edges E.

$$G = (V, E)$$

2.2.1 Directed Graph

A directed graph is a graph where each edge represents an ordered pair of vertices. In-degree of a vertex is the number of edges directed towards the vertex. Out-degree of a vertex is the number of edges directed from the vertex.

2.2.2 Undirected Graph

An undirected graph is a graph where each edge represents an unordered pair of vertices. Degree of a vertex is the number of edges the vertex is connected to.

2.2.3 Path

A sequence of vertices

$$V_1V_2V_3...V_n$$

is a path if

$$\forall i \in \{1, 2, 3, .., n-1\} \qquad (V_i, V_{i+1}) \in E$$

The length of such a path is $n - 1$

2.3 Adjacency Matrix of a graph

For a graph G with n vertices, its adjacency matrix $A(G)$ is an n by n square matrix with $a_{i,j}$ equal to the number of edges connecting V_i to V_j

2.3.1 Properties

For an undirected graph G

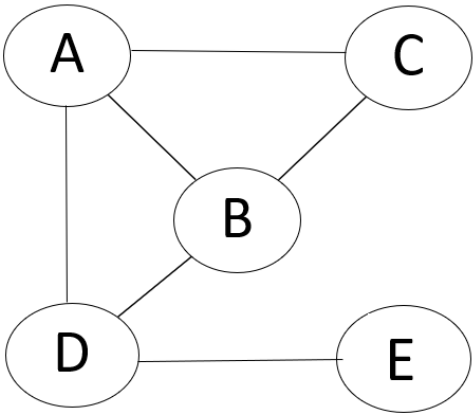
- $A(G)$ is symmetric
- $A(G)$ has real eigenvalues
- (i, j) entry of $A(G)^k$ represents the number of paths of length k from V_i to V_j $\quad \forall k \geq 1$

3 Centrality Measures

Centrality measures use graph theory to calculate the importance of any given node in a network. Centrality has different types and each type or metric defines the "importance" of a node from different perspectives and further provides relevant analytical information about the graph and its nodes.[6]

3.1 Eigen Centrality

Eigen Centrality measures the importance of a node in a graph as a function of the importance of its neighbours. If a node is connected to highly important nodes, it will have a higher Eigen Vector Centrality score as compared to a node which is connected to lesser important nodes.
Eigen Centrality measures a node's influence by also taking into account how well connected a node is, how many links their connections have, and so on through the network.
By calculating the extended connections of a node, Eigen Centrality can identify nodes with influence over the whole network, not just those directly connected to it.



The adjacency matrix A of the above graph will be as shown below:

$$\begin{bmatrix} - & 1 & 1 & 1 & 0 \\ 1 & - & 1 & 1 & 0 \\ 1 & 1 & - & 0 & 0 \\ 1 & 1 & 0 & - & 1 \\ 0 & 0 & 0 & 1 & - \end{bmatrix}$$

Let's assume that in the above graph, the importance of each node is measured by its degree, such that the higher the degree of a node, the more important it is in the graph. Degrees of various nodes are shown below:

Node	Degree
A	3
B	3
C	2
D	3
E	1

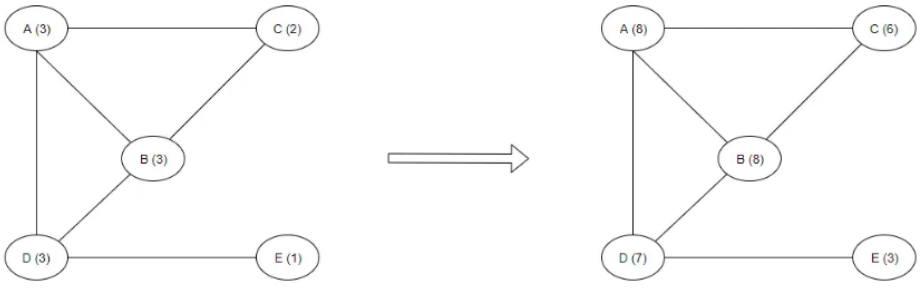
The above can also be represented as a vector V as shown below:

$$\begin{bmatrix} 3 \\ 3 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

Mathematically the Eigen Vector Centrality is calculated as below:

$$\begin{aligned} A \times V &= \begin{bmatrix} - & 1 & 1 & 1 & 0 \\ 1 & - & 1 & 1 & 0 \\ 1 & 1 & - & 0 & 0 \\ 1 & 1 & 0 & - & 1 \\ 0 & 0 & 0 & 1 & - \end{bmatrix} \begin{bmatrix} 3 \\ 3 \\ 2 \\ 3 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 \times 3 + 1 \times 3 + 1 \times 2 + 1 \times 3 + 0 \times 1 \\ 1 \times 3 + 0 \times 3 + 1 \times 2 + 1 \times 3 + 0 \times 1 \\ 1 \times 3 + 1 \times 3 + 0 \times 2 + 0 \times 3 + 0 \times 1 \\ 1 \times 3 + 1 \times 3 + 0 \times 2 + 0 \times 3 + 1 \times 1 \\ 0 \times 3 + 0 \times 3 + 0 \times 2 + 1 \times 3 + 0 \times 1 \end{bmatrix} \\ \Rightarrow A \times V &= \begin{bmatrix} 8 \\ 8 \\ 6 \\ 7 \\ 3 \end{bmatrix} \end{aligned}$$

The resultant 1-D vector in the above equation gives the Eigen Vector Centrality (EVC) score for each of the nodes in the graph.
The effect of the first iteration of multiplication can be visualized as shown below:



As you can see above, nodes A and B both have a high score of 8 since both of them are connected to multiple nodes with high degrees (importance)

while node E has a score of 3 since it is only connected to a single node of degree 3.
Observe that the EVC score value for each node in the resultant vector is nothing but the sum of degrees of its neighbouring nodes.
Multiplying the resultant vector again with the adjacency matrix of the graph helps the EVC score spread out in the graph to get a more globally prominent EVC score vs a localized EVC score for each node in the graph. After the first iteration of multiplication, each node's EVC score is a function of only its direct (first degree) neighbours, thus is a localized score which might not be accurate at a global level in the graph.
Thus, the following observations can be made:

1. After the first iteration of multiplication, each node gets its EVC score from its direct(first degree) neighbours.

2. In the second iteration, when we multiply the resultant vector again with the adjacency matrix, each node again gets its EVC score from its direct neighbours but the difference in the second iteration is that this time, the scores of the direct neighbours have already been impacted by their own direct(1st degree) neighbours previously(from the first iteration of multiplication) which eventually helps the EVC score of any node to be a function of its 2nd degree neighbouring nodes as well.

3. In subsequent iterations of multiplication, the EVC score of graph nodes keeps getting updated by getting impacted by EVC scores from neighbouring nodes of farther degrees (3rd, 4th and so on).

Repeated multiplication makes the EVC score of every node eventually be a function of or dependent on several degrees of its neighbouring nodes, thereby providing a globally accurate EVC score for each node. Usually, the process of multiplying the EVC vector with the adjacency matrix is repeated until the EVC values for nodes in the graph reach an equilibrium or stop showing the appreciable change.
One sample application of EVC is the calculation of Page Rank or Page Rank algorithm used by Google and many other companies to rank web pages on the internet by relevance. Page Rank is a direct variant of EVC. Web pages on the World Wide Web have links that point to/from other web pages. You can think of each web page as a node in the graph and each outgoing/incoming link as a directed edge leading to/from another web page on the web, thereby making up the whole World Wide Web graph. The graph of web pages in the world wide web undergoes several iterations

of EVS calculation so as to calculate globally accurate relevance rankings of each web page. The web pages with high EVC scores can then be targeted for marketing and other commercial purposes. [5]

4 Clustering

Graph clustering, also known as community detection, is the process of partitioning a graph into groups or clusters, where nodes within the same cluster are more closely connected to each other than to nodes in other clusters. The goal of graph clustering is to uncover meaningful structures or communities within a graph, allowing us to better understand the relationships and organization of the nodes. Clustering algorithms aim to maximize intra-cluster connectivity while minimizing inter-cluster connectivity, resulting in groups of nodes that are densely connected internally and sparsely connected externally.

4.1 Some popular methods of graph clustering

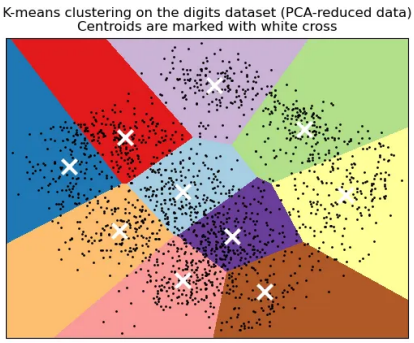
Several algorithms exist for graph clustering, each with its own approach and assumptions.

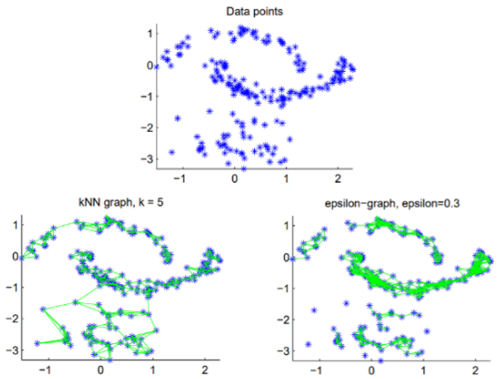
4.1.1 K-Means Graph Clustering Method

This method partitions the graph into clusters that appear with a k-shape. It uses k-means to compute the centroids and their vector quantities. The steps involved in constructing the K-means clustering method include: selecting K to represent the original centroids, assigning points to the nearest centroids from the K-cluster, and conducting a re-evaluation of the centroids to ensure they do not change. One advantage of this method is that it can handle large data sets.

4.1.2 Spectral clustering

This approach leverages the spectral properties of the graph to perform clustering. It involves transforming the graph into a matrix representation and using eigenvalue decomposition or other spectral techniques to extract clusters. Spectral clustering can handle graphs with irregular structures and is particularly effective for identifying clusters with non-convex shapes.





The choice of clustering algorithm depends on various factors, such as the nature of the graph, the size of the graph, the desired level of granularity, and the specific goals of the analysis. Additionally, it is important to consider the scalability and computational complexity of the algorithm, as some methods may be more suitable for large-scale graphs. Once the graph clustering algorithm has been applied, the resulting clusters can be analyzed to gain insights into the structure and organization of the graph. Cluster evaluation measures, such as modularity, silhouette coefficient, or normalized mutual information, can be used to assess the quality of the clustering results.

4.2 Connection between Clustering and Eigenvalues

4.2.1 Spectral Radius

One important eigenvalue associated with the adjacency matrix is the spectral radius, which is the largest eigenvalue of the matrix.

The spectral radius provides information about the expansion properties of the graph. If the spectral radius is small, it suggests that the graph has a well-connected structure with tight clusters. On the other hand, a large spectral radius indicates a more dispersed and less clustered graph.

4.2.2 Algebraic Connectivity

Another relevant eigenvalue is the second smallest eigenvalue, also known as the algebraic connectivity.
The algebraic connectivity measures the connectivity and robustness of the graph. Larger algebraic connectivity implies that the graph has more connections between its nodes, indicating a higher likelihood of clustering.

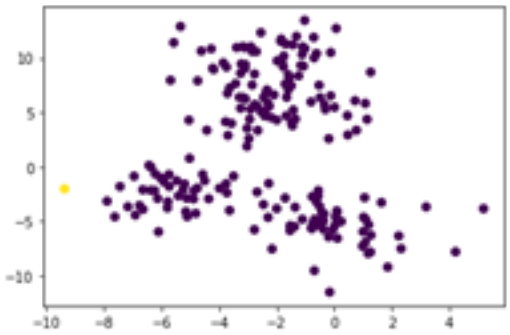
Additionally, the eigenvalues of the adjacency matrix can be used to identify the number of clusters in a graph. This is achieved by examining the gaps between consecutive eigenvalues. Large gaps suggest the presence of distinct clusters, while small gaps indicate a more homogeneous graph without well-defined clusters.
Furthermore, eigenvectors associated with the eigenvalues can provide information about the clustering structure within the graph. The eigenvectors are vectors that correspond to the eigenvalues and represent the distribution of nodes within the graph. By examining the components of the eigenvectors, it is possible to identify nodes that belong to the same cluster or community.

4.3 Clustering with Linear Algebra

4.3.1 Creating a Dataset

Our data consists of 200 2-vectors simulated by the `make_blobs` function in scikit-learn.
Plotting them, we see we see our data spanning following 2D-space.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

dataset = make_blobs(n_samples = 200,
                     centers=4,
                     n_features=2,
                     cluster_std=1.6,
                     random_state=50)
...
```



4.3.2 Clustering Analysis

Let's analyze the data using a clustering method. We start by specifying a cluster assignment vector c, which informs us about which vector belongs to which cluster.
For this example, we arbitrarily initiate the cluster assignments as $c = [1, 2, 3, 1, 2, 3, 1, 2, 3]$, indicating that cluster 1 contains vectors v_1, v_4, v_7, cluster 2 contains vectors v_2, v_5, v_8, and cluster 3 contains vectors v_3, v_6, v_9.

1. **Centroid Calculation**

   To calculate the similarity of the data points to each other, we use a group representative called the centroid. The centroid for each cluster is calculated as follows:

$$\text{centroid}_j = \frac{1}{|G_j|} \sum_{i \in G_j} x_i$$

   where G_j represents the elements in group j, and x_i denotes the vectors in the dataset.

2. **Clustering Quality**

We can assess the quality of the current clustering assignment by calculating the mean square distance from the vectors to their associated centroids. This is done using the following equation:

$$\mathbf{z}_1 = \frac{1}{|G_1|} \sum_{i \in G_1} \mathbf{x}_i$$

Now we have every ingredient needed to calculate the quality of the current clustering assignment. For this, we calculate the mean square distance from the vectors to their associated representatives with the following equation:

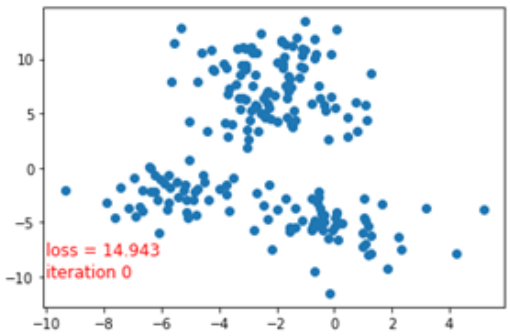
$$\text{Quality} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{z}_{c_i}\|^2$$

where N is the total number of vectors, x_i represents the vectors, and z_{c_i} denotes the centroid corresponding to the cluster assignment c_i.

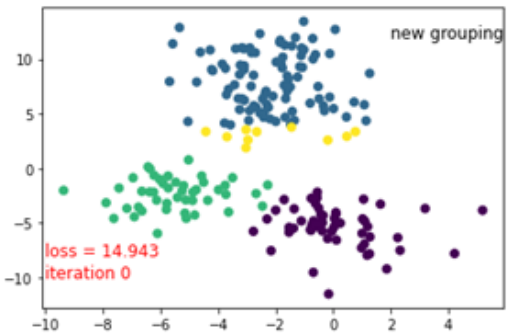
```
def clustering_objective(data, grouping, centroids):
    J_obj = 0
    for i in range(len(data)):
        for j in range(len(centroids)):
            if grouping[i] == (j+1):
                J_obj += np.linalg.norm(data[i] -
                         centroids[j])**2
    J_obj = J_obj/len(data)
    return J_obj

clustering_objective(data, init_group, init_cent)

'''
outputs:

15.680185185185186
'''
```

4.4 Visual Application

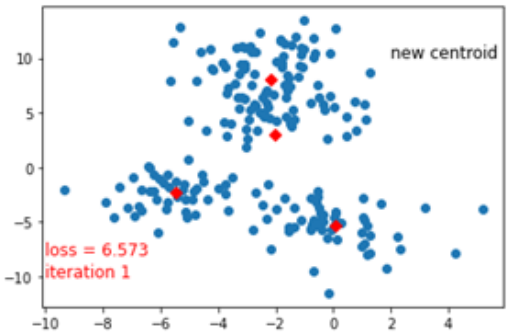
With the initial dataset we have a high loss (mean-squared error):



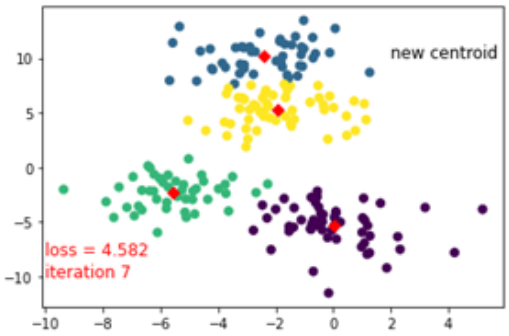
1. In Step 1 we assign the data points to the cluster with the nearest centroid:



2. In Step 2 we calculate the centroids again with the assigned data points:



3. Our final clustering solution then looks like this:



In this example, linear algebra techniques were used to perform clustering analysis. The centroids obtained from the clustering process can be used as representatives of the clusters, providing valuable insights into the dataset.

5 Connectivity

We will analyse the connectivity of a graph using its adjacency matrix and its eigenvalues.

5.1 Connected Components [1]

Consider a regular undirected graph G (each vertex having degree d), and some eigenvector x of $A(G)$ with its corresponding eigenvalue λ

$$A(G) * x = \lambda * x$$

$$\lambda x_i = \sum_j A_{ij} x_j$$

$$|\lambda||x_i| \leq \sum_j |A_{ij}||x_i|$$

$$|\lambda||x_i| \leq d|x_i| \qquad (Since \quad |x_j| \leq |x_i| \forall j)$$

$$|\lambda| \leq d$$

All the eigenvalues of this graph are smaller than or equal to d. [2]
Upon setting all x_i to 1, we get $Ax = dx$, which gives

$$\lambda_{max} = d$$

The multiplicity of the maximum eigenvalue (equal to d) gives the number of connected components of G. [3] The number of connected components of a graph have various applications in different fields

- Social Network Analysis: Connected components are used to identify a set of individuals as a community. By finding these components, insights on information flow and social interactions are studied by researchers.
- Image Processing: Segmentation of objects and finding regions of interest is done by finding the connected components of an image. The image is converted into a graph, and then pixels sharing similar characteristics are grouped together. This is useful for object recognition and feature extraction.
- Network Routing: In large-scale computer networks, connected components are used by routing algorithms to determine the best path that the data packets should take in order to reach their destination.

5.2 Number of paths in a graph [4]

Consider an undirected graph G with adjacency matrix $A(G)$. Let $\lambda_1, \lambda_2, \lambda_3, ..., \lambda_n$ be the eigenvalues of $A(G)$

$$\forall i, j \quad \exists c_1, c_2, c_3, ..., c_n \quad (A(G)^k)_{ij} = c_1\lambda_1^k + c_2\lambda_2^k + c_3\lambda_3^k + ... + c_n\lambda_n^k \quad \forall k \geq 1$$

For each index (i, j), upon identifying its corresponding c_i values, the number of paths from V_i to V_j of any length can be calculated in constant time using this equation.
Similarly, the total number of self loops of length k is given by

$$\lambda_1^k + \lambda_2^k + \lambda_3^k + ... + \lambda_n^k$$

Since the number of paths of length k between two vertices V_i and V_j is given by $(A(G)^k)_{i,j}$, for a given (i, j)
we precalculate $(A(G)^k)_{i,j} \quad \forall k \in \{1, 2, ..., n\}$ and store them in the matrix

$$Y = \begin{bmatrix} (A(G))_{i,j} \\ (A(G)^2)_{i,j} \\ (A(G)^2)_{i,j} \\ (A(G)^3)_{i,j} \\ . \\ . \\ . \\ (A(G)^n)_{i,j} \end{bmatrix}$$

Now, after calculating the eigenvalues of the adjacency matrix $A(G)$, we construct the followng matrix

$$A = \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 & . & . & . & \lambda_n \\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2 & . & . & . & \lambda_n^2 \\ \lambda_1^3 & \lambda_2^3 & \lambda_3^3 & . & . & . & \lambda_n^3 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ \lambda_1^n & \lambda_2^n & \lambda_3^n & . & . & . & \lambda_n^n \end{bmatrix}$$

To solve for the values of c_i, we solve the system of linear equations

$$A * C = Y$$

After we get the resultant C vector, we can compute the number of paths of length $k \quad \forall k \geq 1$ simply be performing the dot product

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ . \\ . \\ . \\ c_n \end{bmatrix} \cdot \begin{bmatrix} \lambda_1^k \\ \lambda_2^k \\ \lambda_3^k \\ . \\ . \\ . \\ \lambda_n^k \end{bmatrix}$$

Since we only have to precompute the c_i values by solving linear equation in n variables, we can find number of paths of length k between two vertices for any arbitrarily large k in a fraction of the time required by regular matrix multiplication.
The number of paths of given length between any two vertices is a very useful metric to have

- Network Analysis: The number of paths between any two access points in a network informs us of its reliability. We can figure out the number of alternate paths the network can take, and how to distribute the packets between all the paths in order to optimally transmit data.
- Reachability Analysis: The presence of a direct or indirect path between any two vertices gives insights on information flow and communication possibilities. The number of paths tells us how easily accessible any vertex is from any other vertex.

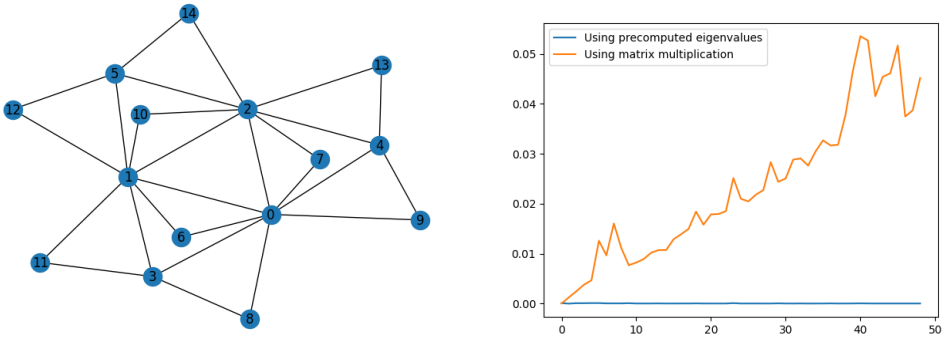


Figure 1: Comparison in speed for a graph with 15 nodes

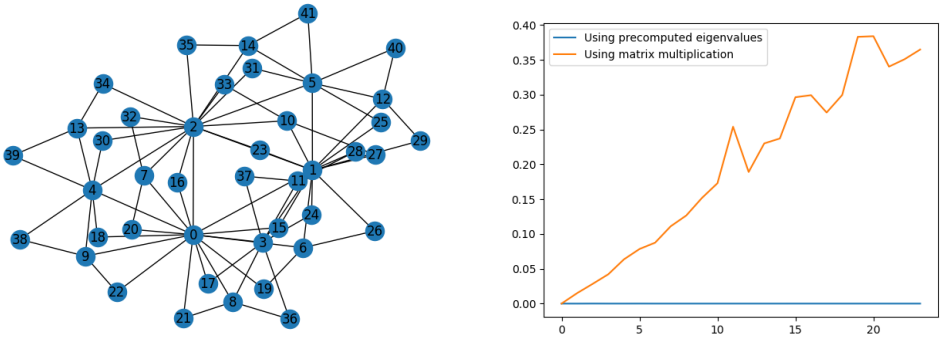


Figure 2: Comparison in speed for a graph with 42 nodes

- Disease Spread: While modelling the spread of a disease, individuals and locations act as vertices in the graph, and the number of paths quantify how likely it is that the disease spread to the other end of that path. This informs us regarding control strategies and which locations require quarantine measures the most.

6 Conculsion

We have successfully implemented the methods to calculate number of paths and shown that they are significantly faster than conventional methods, given a certain amount of precomputation.

References

[1] Science, B. O. C., Science, B. O. C. (2022). Connected Components in a Graph | Baeldung on Computer Science. Baeldung on Computer Science. https://www.baeldung.com/cs/graph-connected-components

[2] Aditya Bhaskara, CS Utah (2016). Graph Partitioning, basic linear algebra https://users.cs.utah.edu/ bhaskara/courses/x968/notes/lec4.pdf

[3] Aditya Bhaskara, CS Utah (2016). Eigenvalues of A_G and the Laplacion https://users.cs.utah.edu/ bhaskara/courses/x968/notes/lec5.pdf

[4] Derek Chen, Advay Goel (2021). Spectral Graph Theory https://math.mit.edu/research/highschool/primes/materials/2021/December/Chen-Goel.pdf

[5] Bhasin, J. (2022, March 30). Graph Analytics — Introduction and Concepts of Centrality. Medium. https://towardsdatascience.com/graph-analytics-introduction-and-concepts-of-centrality-8f5543b55de3

[6] Disney, A. (2022, October 27). Social network analysis: Understanding centrality measures. Cambridge Intelligence. https://cambridge-intelligence.com/keylines-faqs-social-network-analysis/

[7] Tuzsus, D. (2022, May 14). Introduction to Applied Linear Algebra: K-Means Clustering. Medium. https://medium.com/mlearning-ai/introduction-to-applied-linear-algebra-k-means-clustering-c6885cad0f7f

[8] Wong, K. J. (2023, May 27). 6 Types of Clustering Methods — An Overview - Towards Data Science. Medium. https://towardsdatascience.com/6-types-of-clustering-methods-an-overview-7522dba026ca

[9] Miller, J. (2020, June 8). An Introduction to Spectral Graph Theory https://sites.math.washington.edu/mõrrow/336_20/papers20/jason.pdf