Clustering Quality and Effectiveness of the Fiedler Vector

1. ABSTRACT

PURPOSE: Transform an edge list to an adjacency matrix and a Laplacian matrix, and to use the Fiedler vector to assign vertices in a graph to one of two clusters.

METHODS: The adjacency matrix was created by initializing a square matrix with zeroes, and then extracting the vertices for each edge from the given edge list that was provided by the instructor. The Fiedler Vector is created by finding the Laplacian matrix, which then allows for an eigenvector decomposition, leading to the eigenvector corresponding to the second smallest eigenvalue being found. The Fiedler Vector is then used to create two clustered sets with the values in the sets being either -1 or 1.

RESULTS: The Laplacian matrix is created along with the adjacency matrix, once the degree matrix has been found. None of the matrixes have been printed out. The twenty values are clustered into two sets of ten, with the values being either 1 or -1.

CONCLUSIONS: An adjacency matrix can only be formed by extracting the vertices from each edge, and the Laplacian matrix can only be formed once the adjacency and degree matrix are known. Furthermore, the set that contains ones are based on the corresponding entries in the Fiedler Vector that are greater than 0, and vice versa. Lastly, the values within the sets have been given a value of either 1 or -1, which shows a binary partitioning of the entries in the Fiedler Vector.

2. INTRODUCTION

The objective or scientific question that was posed here was to perform graph clustering by using MATLAB so as to apply concepts from graph theory, specifically with respect to spectral graph theory.

It’s important to understand some background concepts that relate to this report. The first of these would be understanding what an edge list is, and the topic of graph theory. A graph consists of vertices and these vertices are connected by edges. To represent an edge between two vertices, an edge list can be used, wherein each row of the edge list corresponds to an edge between two vertices. For the purpose to be successfully completed, the manipulation and interpretation of an edge list was quite important here. Other concepts that need to be looked at before previewing this report include a basic understanding of a Laplacian matrix, a Fiedler vector, and graph clustering. A Laplacian matrix is based on graph connectivity, and is derived from an adjacency matrix. A Fiedler vector is the second smallest eigenvalue of a Laplacian matrix, and is extremely useful for graph partitioning and clustering. Graph clustering is

specifically based upon grouping vertices of a graph based on certain specifications. Spectral clustering can be done through the Fiedler vector.

In order to test the scientific question, multiple steps needed to take place. The graph was clustered into two sets, one with values that contained ones and the other that contained values with negative ones. However, prior to this step, other concepts of graph theory and linear algebra were needed. A provided edge list had to be converted into an adjacency matrix. This was then converted into a Laplacian matrix, from which the Fiedler vector was extracted. Once the Fiedler vector had been solved for, the partitioning of the entries in the Fiedler vector allowed for graph clustering, leading to the creation of two sets.

3. METHODS

After having found the adjacency matrix from the edge list and the degree matrix from the diagonal entries of the degree values, the Laplacian matrix was calculated by subtracting the degree matrix from the adjacency matrix. From the Laplacian matrix, the eigenvalues and eigenvectors were computed. The second column of eigenvectors was extracted as the Fiedler vector, which then corresponded to the second smallest Eigenvalue, or the Fiedler eigenvalue. Then, the Fiedler vector's thresholding took place; any negative Fiedler vector values were assigned to one set and any positive or zero Fiedler vector values were assigned to the other set. Both sets were created from binary clustering, wherein the first set contained the values of -1, while the other set contained the values of 1.

For testing purposes, such that the reproduction of results takes place, an input edge list is required. This has been defined by the variable of 'elist'. This edge list must be a two-dimensional array, where there can be any number of rows but there can only be two columns. Without this format of the edge list, the reproduction of results from the code will not take place. Each row within the edge list is representative of an edge in the graph, and the two columns showcase the pair of vertices that from an edge. The first column shows the source vertex of an edge, while the second column shows the source destination of an edge. If these requirements are satisfied thoroughly, then, any type of graph, regardless of structure and size, can reproduce the results that have taken place. Additionally, besides this requirement, the reproduction of results will require the edge list to be loaded into the MATLAB code. If the name of the file that contains the edge list starts with a digit and is being used as a parameter, then, any letter must be put prior to this digit so as to ensure that the code runs successfully. The rigorous usage of this process will allow for the console to display the vertices in both sets, and visual verification can be done using the Cartesian and clustered views for the graph. While this process would also work for larger graphs, the optimization aspect for larger graphs will be missing. One way by which this could be improved is to use an interactive testing approach, such that users can input their own edge list during runtime for real-time verification. Although, by using this process, a correct and reliable solution can be provided for graph clustering.

The results were evaluated in two separate ways: visually, and quantitatively. A visual assessment of the results was confirmed by comparing the quality of clustering with the

Cartesian and clustered views that were produced. Along with this, the two sets were assigned vertices, and it was a requirement to check that the vertices in each set did not repeat in both sets. A quantitative aspect was also needed to evaluate the results, and this was done by assessing the number of vertices in each set based on the clustering vector. In the edge list that the instructor had provided, each set contained ten vertices, i.e., ten values within the set that contained the values of -1, and ten values within the set that contained the values of 1. When the personalized edge file was inputted into the code; the results were the same; ten vertices were part of each set. This allowed for a check of the effectiveness and balance of the clustering. Another quantitative aspect that took place, was displaying and comparing the vertices within both sets by using the MATLAB console output. Through the output that MATLAB provided, a comparison was made between the output provided and the expected outcomes based on the specific characteristics of the tested graph, as well as the theoretical expectations of the behavior of the Fiedler's vector in general. To produce the exact same results as are found within this report, the source of the dataset is one that is external and has been loaded into the MATLAB environment. This is by no means necessary; so long as the format is similar to that which has been described in the previous paragraph, any edge list can create a similar graph clustering, regardless of the structure and the size of the edge list.

4. RESULTS

Some of the most important statistics, for the purposes of this report, are in relation to the clustering that takes place by the Fiedler vector and the original properties of the edge list. Specifically, the graphs and tables are built around the values of the vertices after clustering, the number of vertices in each set after clustering, the degree of each vertex, and the connectedness or a formation of an edge between vertices, depending on the set that they are placed within. The second table and figure are crucial and will be spoken about further in the discussion section. The degree of each matrix can point towards the effectiveness of the Fiedler vector for the given dataset, and the connectedness or formation of edges between vertices, depending on whichever set they are located in, provides better emphasis on the quality of clustering.

Table 1: Value of each vertex after being clustered by the Fiedler vector

Vertex Number	Value of the Vertice
1	-1
2	1
3	1
4	-1
5	-1
6	1
7	-1
8	1

9	1
10	-1
11	1
12	-1
13	-1
14	1
15	-1
16	-1
17	1
18	1
19	1
20	-1

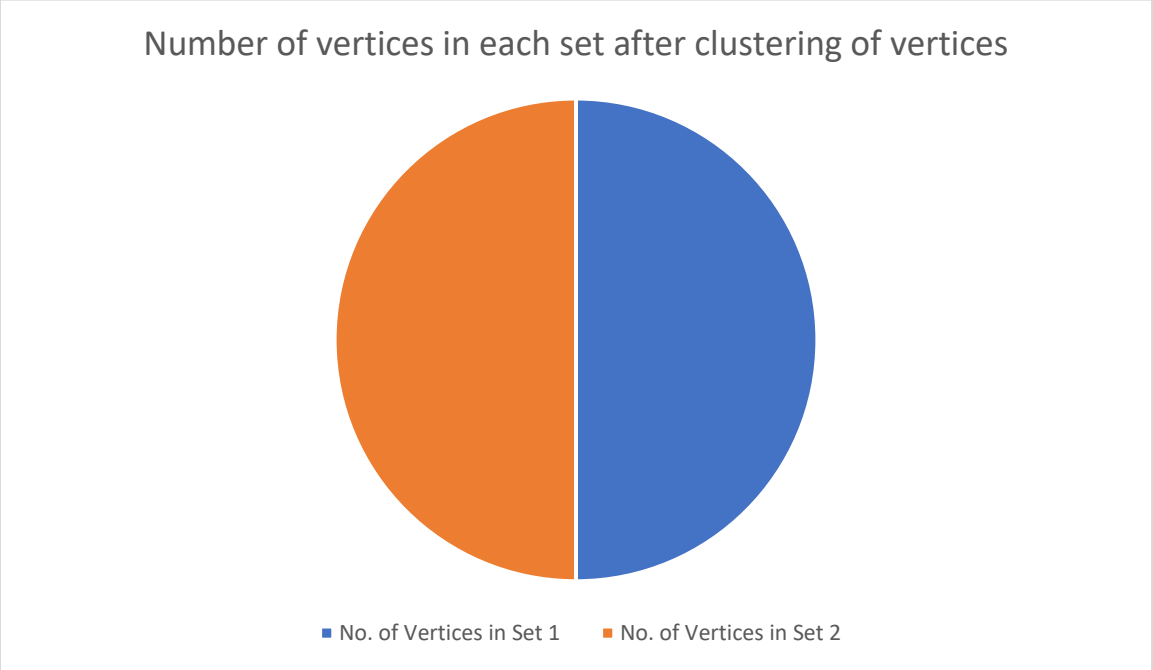

Figure 1: Number of vertices in each set after clustering of vertices

Table 2: Degree of each vertex based on the edge list

Vertex Number	Degree of the vertex
1	8
2	9
3	11
4	8
5	9
6	10

7	9
8	9
9	9
10	6
11	10
12	7
13	10
14	9
15	11
16	7
17	8
18	9
19	10
20	7

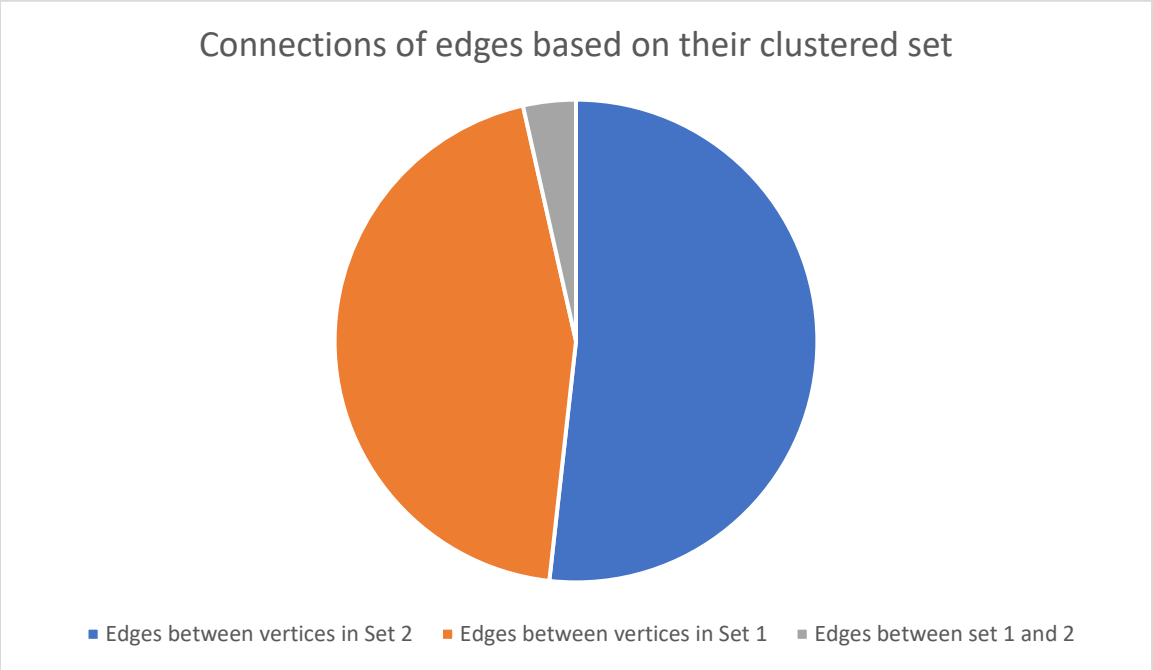

Figure 2: Connections of edges based on their clustered set

5. DISCUSSION

On the completion of the graph clustering and the plotting of the Cardinal and Cartesian graphs, distinctive trends could be noticed through the graph clustering that took place due to the application of the Fiedler vector. Two distinct sets, namely ‘set1’ and ‘set2’, were classified binarily based on the Fiedler vector. From these clusters, it was noticed that each set contained ten, or half, of the twenty vertices that were originally part of the edge list given. None of the vertices repeated in either set, and each set was clustered based on the Fiedler vector’s value,

specifically in regard to whether the vector's value was positive or not. The separation of the vertices from both sets was then visualized through a graphical representation of the clustered graphs. Regardless of the edge list that is provided, the code used should successfully be able to split the vertices in the edge list into two separate clusters, or sets. Overall, the analysis of the graph clustering results shows an effective process to cluster vertices into two different sets based on the Fiedler vector. The identified trends from the clustering further allow for broader implications and meaningful conclusions to be drawn, which will be spoken about in the forthcoming paragraphs.

One of the most important aspects of the results that needs to be looked at quite closely is the quality of the clustering. There are only six edges that contain one vertex from the first set and the other vertex from the second set. In comparison, there are eighty-eight edges that have vertices only from the second set and seventy-six edges that have vertices only from the first set. Out of the one hundred and seventy total edges that have been formed between the twenty vertices in the edge list, only 4% of them are edges that have one vertex in one of the sets and a second vertex in the other set. Therefore, it is safe to say that the quality of clustering is extremely efficient here. The role of the Fiedler vector in the partitioning of this graph is crucial, and understanding the importance of it is what led to the excellent quality of the clustering of the graph. Since distinct clusters have been formed, in this specific scenario, it can be said that the Fiedler vector was extremely effective here. None of the vertices lie in both sets, and the sets have been distributed evenly such that half of the vertices are in one set and half of the vertices are in the other set. Furthermore, due to these characteristics, the underlying structure of the graph is captured quite well by the Fiedler vector. This success can be attributed to the Laplacian matrix, which possess eigenvectors that reflect the connectivity patterns within the graph. By using the Fiedler vector, which is the second smallest eigenvector of the Laplacian matrix, the finer details of graph connectivity are exposed. The Fiedler vector's main role here was to maximize separation between clusters, and to highlight the connections between the different parts of the graph, which thus allows for effective partitioning. Its formula of finding subtle relationships between vertices also relates to the eigenvalues of the Laplacian matrix. The second eigenvalue of the Laplacian matrix gives a better idea of the graph's bipartite nature. This is another reason why vertices can be clearly split or separated into distinct sets through the Laplacian matrix, and more specifically, through the Fiedler vector. Having said this, while the clustering process used here would work on any edge set provided, the characteristics of the clustering of sets provided here would not necessarily be the same. The sets might have more or less vertices in each set, depending on the connectivity of the edges within the edge list. This will also then mean that the vertices in both sets might not be split equally, which is another potential limitation of the clustering process. The process however, of finding an adjacency, degree and Laplacian matrix, and then finding the Fiedler vector to cluster the graph, remains the exact same. In terms of real-life applications, it is possible for the Fiedler vector to be of merit, but other factors, including a dataset's characteristics, scale, and underlying structural complexities must also be considered. The Fiedler vector is extremely useful when looking at network

analysis, spectral graph drawing, image segmentation, and optimization problems. The reason why it does so well in these areas is due to its reflection on the algebraic connectivity of the graph, which showcases how well-connected the graph is and how it can be split into disconnected components. Through this process, the Fiedler vector is also able to apply important nodes or communities within a network. Along with this, the implementation of the Fiedler vector can segment images into regions with similar properties. From here, a pixel connection can be constructed, and the graph can be partitioned into meaningful regions. Many real-life applications are based on such ideas, and so, it can be said, provided other properties, characteristics, and trends in favor, that the Fiedler vector could come handy for similar real-life applications as those shown in the code here. There is no real chance of human error taking place within this code. The only way this may happen is if there exist typos somewhere in the code, or within the edge list. For example, typos within the edge list would lead to potential errors in the clustering, and even in the matrixes that allowed for the clustering. Similarly, an error in the understanding of important concepts (such as, possibly assigning an adjacency matrix to a degree matrix) could lead to an incorrect clustering of vertices. Besides this, it is virtually impossible for human error to take place, as the process to receive the clustering is quite smooth and is the same, regardless of the connections within the edge list. None of the errors mentioned here are part of the code, but if they were, they would give an incorrect clustering. This would happen because of one of the previous steps prior to the clustering being done incorrectly, which then impacted the clustering, either directly or indirectly.

In conclusion, the Fiedler vector clustered successfully, with high quality, so as to identify distinct sets within the graph. The sets, labeled as ‘set1’ and ‘set2’, each have half, or ten of the twenty vertices in the edge list and none of these vertices are part of both of these sets. Through this clustering approach, one receives valuable insights into an underlying structure of a graph.

6. REFERENCES

1. Ellis RE. Introduction, with a Review of Eigenvalues and Eigenvectors [Internet]. Queen’s University; 2021 [cited 2024 Jan 16]. Available from: https://research.cs.queensu.ca/home/cisc271/pdf/Class01.pdf
2. Ellis RE. Graphs: The Adjacency Matrix [Internet]. Queen’s University; 2021 [cited 2024 Jan 16]. Available from: https://research.cs.queensu.ca/home/cisc271/pdf/Class02.pdf
3. Ellis RE. Graphs: The Laplacian Matrix [Internet]. Queen’s University; 2021 [cited 2024 Jan 16]. Available from: https://research.cs.queensu.ca/home/cisc271/pdf/Class03.pdf
4. de Abreu NMM. Old and new results on algebraic connectivity of graphs. Linear Algebra and its Applications. 2007 May;423(1):53–73.
5. 2.L11: Spectral Clustering [Internet]. [cited 2024 Jan 17]. Available from: https://users.cs.utah.edu/~jeffp/teaching/cs5955/L11-spectral.pdf