Clustering of Graph Vertices

Abstract

PURPOSE: The purpose of this assignment is to become familiar with MATLAB operations and to explore vertex clustering.

METHODS: The text files were used as data sets and the starter code given by the professor were used to implement the necessary operations to classify the vertex clusters, and to graph the data as a grid and as clustered graph.

RESULTS: The results obtained in this assignment include two graphs describing the vertices and a table showing the vertex clustering.

CONCLUSIONS: The upper and lower graphs represent the edge connections between vertices and the clustering of vertices, respectively. The table shows the sets of clustered vertices.

Introduction

In this assignment we are looking to familiarize ourselves with MATLAB as well as analyze graphs in linear algebra.

To accurately understand the process used to implement this idea, several topics in linear algebra must be considered. A graph in linear algebra is often used to describe relationships and can be represented as a matrix. A graph also has two sets, the set of edges(e) and the set of vertices(v). A vertex can be any point of the graph, and an edge connects two vertices. Given a graph, which has a set of edges labelled $(v_i v_j)$and vertices labelled $(v_1, v_2, \dots v_n)$, we can then construct an adjacency matrix A, in which each entry a_{ij} of the matrix is either 1 if $(v_i v_j)$ is an element in the set of edges or 0 otherwise. From the adjacency matrix, we can then compute the Laplacian matrix, which can be found by taking the degree matrix of a graph and subtracting the adjacency matrix. The Laplacian matrix is unique in that it can be partitioned as diagonal blocks that are non-zero matrices and off diagonal blocks that are zero matrices. It is useful to note that the number of components of a graph matches the dimension of the null space. The first eigenvalue of a Laplacian matrix is always zero, since it is associated with the ones vector. As such, we know what the eigenvector will look like. The second eigenvector is more useful and is named the Fiedler vector. The Fiedler vector is the eigenvector of the second smallest eigenvalue. The signs of the vector entries of the Fiedler vector correspond to the visual clustering of the graph vertices, with the negative entries belonging to one set of vertices and the positive entries belonging to another set of vertices.

A testable hypothesis for this experiment is to show that the binary clustering of vertices from an edge list can be accurately displayed by a graph showing the connections of the edges between them.

Methods

To implement this problem, we can first load the text file containing the set of edges and save it to a variable, using that variable name, we ca then call the function a1_20144315 with the parameter being the file variable. The file in this parameter is represented as a matrix of edge pairs. Within this function, the number of vertices on the graph is calculated to determine the appropriate problem size. The adjacency matrix is found using a for loop that changes an initial zeros matrix and changes the values where there is a vertex from 0 to 1. The next step is to cluster the data, and to do this the degree matrix needs to be calculated from the adjacency matrix using equation 3.1 from the class notes $D(G) = diag([A]\underset{1}{\rightarrow})$. The Laplacian matrix is then defined from earlier as $L = D - A$, which corresponds to equation 3.2 in the class notes. From there we then compute the Fiedler vector by finding the eigenvalues and eigenmatrix of the Laplacian matrix and then setting the second eigenvector as the Fiedler vector. After finding the Fiedler vector a for loop is used to determine the sign of each entry. When the sign of the entry is discovered it is then placed in a clustering vector, with positive entries denoted as +1 and negative entries denoted as -1. Another for loop is then used to separate the positive values into set 1 and the negative values into set 2.

From the a1_20144315 function, the plot271a1 function is called, in which it produces two graphs, one being a Cartesian grid, and the other being a graph using the calculated clustering of vertices of a graph.

RESULTS

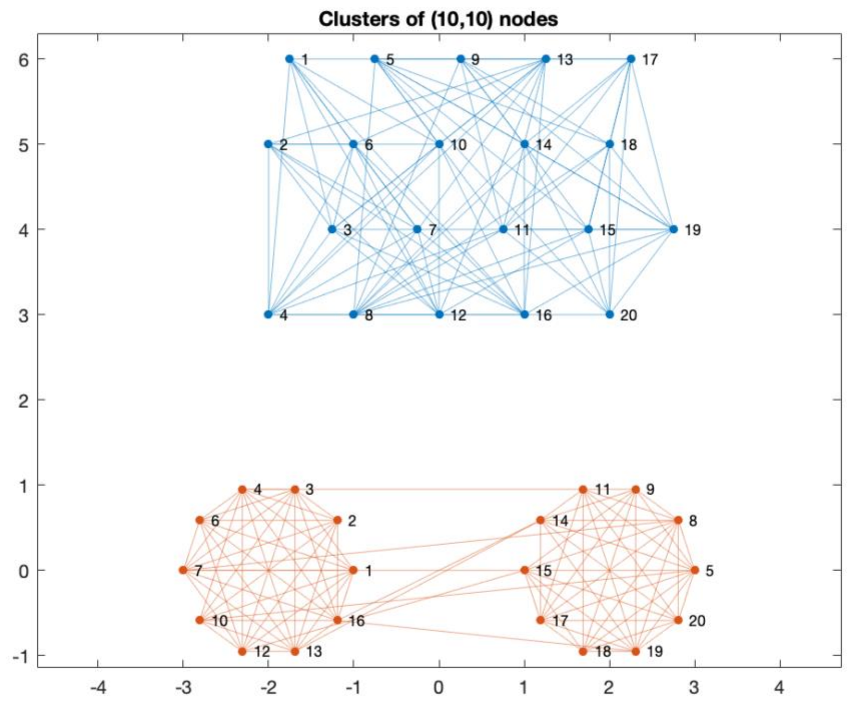

Figure 1. The output graphs of the 18ldjm.txt file. The upper graph shows the cartesian grid and the lower graph shows the vertex clustering.

Table 1. Vertices contained in each cluster. Each row corresponds to a set of clustered vertices.

Set	Vertices									
1	1	2	3	4	6	7	10	12	13	16
2	5	8	9	11	14	15	17	18	19	20

DISCUSSION

From the results, we can see that from the implemented algorithms there are two graphs produced as well as two separate sets.

In Figure 1, The upper graph shows all of the vertices that were discovered in the edge list. The number of vertices was found in the beginning of the algorithm where the problem size is equal to the max value found in the list. We can also see that edge connections are shown in the cartesian grid. The edge connections represent the original input edge list text file that was interpreted as an $M \times 2$ array of edges. These edges were used in a for loop to construct the adjacency matrix, in which we can see which vertices are connected to each other by an edge. This adjacency matrix is passed to the plot271a1 function. In the lower graph, we can more clearly see the clustering of the vertices. Each cluster is plotted as a circle of vectors, while still showing the edge connections. There are significantly more connections between the vertices within the circle, than in between the two clusters. This demonstrates that the calculation of the clustering vector was done accurately.

In Table 1, we can see that this table shows each vertex and which cluster it belongs to. The algorithm that determined this partitioning is in the a1_20144315 function when the Fiedler vector was calculated and a for loop was used to determine the sign of each entry to make the clustering vector. From the clustering vector another for loop is used to determine which vertex each entry belongs to, which was represented by the index of the loop iteration. The vertex number was then placed in the appropriate set to be displayed in the output.

The adjacency matrix, along with the binary clustering of vertices as a vector with entries +1 belonging to one set and entries -1 belonging to another, are then passed as parameters to the plot271a1 function, where the two graphs are produced by the code written by the professor. The adjacency matrix is used to create the first plot and the clustering vector is used to create the second.

Graph clustering can be useful for many real-world applications. For example, say we have an abstract representation of a map of cities, where there are edges representing the roads connecting vertices as street corners. We could then implement this algorithm do display the vertices on a graph and see the visual clustering of the streets. From this, we could see the clustering of the cities and see more highly urbanized areas on a map.

From these results, we can see that the mathematical concepts discussed, and the algorithms implemented were able to produce a visualization of the edge connection between vertices as a graph and was able to show specifically which vertices belong to which cluster. From this information, we can conclude that our testable hypothesis is valid.