MSc Data Science

Data-driven Graph Construction:
Unlocking Insights through Clustering and Partitioning

***Authors*:**
Soumyajoy Kundu
soumyajoy.mds2023@cmi.ac.in
Sreeja Choudhary
sreeja.mds2023@cmi.ac.in
Suneha Sen
suneha.mds2023@cmi.ac.in
Soumya Dasgupta
soumya@cmi.ac.in

***Supervisor*:**
Priyavrat Deshpande
Associate Professor
Chennai Mathematical Institute
pdeshpande@cmi.ac.in

May 2024

Work Contributions

Member	Roll Numbers	Contributions
Soumya Dasgupta	MDS202348	1. Understanding the content of the paper.
Soumyajoy Kundu	MDS202349	1. Understanding the content of the paper. 2. Explained the paper to others. 3. Structuring the flow and writing the report. 4. Exploring the topic, studying related works and references. 5. Proposed new ideas to the team. 6. Implementing ideas and experimenting with code.
Sreeja Choudhary	MDS202350	1. Understanding the content of the paper. 2. Experimenting with code. 3. Discussed relevant ideas.
Suneha Sen	MDS202351	1. Understanding the content of the paper. 2. Discussed relevant ideas. 3. Helped to write *Introduction*.

Abstract

Graph partitioning is a vital task in data analysis and computational sciences, offering broad applications. This study scrutinizes the efficiency of spectral partitioning techniques, specifically leveraging Fiedler's vector alongside machine learning algorithms like k-means, for partitioning complex graphs. By investigating this fusion, we aim to elucidate its suitability for addressing challenges inherent in intricate graph structures. Through examination of diverse datasets, including synthetic and real-world examples, we assess the efficacy of spectral partitioning compared to traditional machine learning methods. Our findings highlight the exceptional effectiveness of spectral partitioning, particularly in scenarios with intricate and non-convex structures. This integration of spectral partitioning and machine learning presents a promising avenue for overcoming challenges in data analysis and computational sciences. Nonetheless, the choice between these techniques depends on factors such as dataset characteristics and interpretability requirements. Further research and empirical validations across a broader spectrum of datasets are imperative for substantiating and generalizing these findings, thereby facilitating informed decision-making in graph partitioning tasks.

Contents

1 Introduction

In the domain of data analysis and computational sciences, effectively partitioning complex graphs is a fundamental endeavor with widespread applications. Graphs serve as powerful representations of interconnected systems, ranging from social networks to biological structures. Among the various techniques for graph partitioning, Spectral partitioning methods, including the utilization of Fiedler's vector, and machine learning algorithms like k-means, have garnered significant attention.

This paper embarks on an investigation into existing methods of graph partitioning, focusing particularly on Spectral partitioning techniques leveraging Fiedler's vector alongside machine learning algorithms such as k-means. Graph partitioning plays a pivotal role in numerous fields, facilitating tasks such as community detection and network optimization. By examining the amalgamation of Spectral partitioning and machine learning, this study aims to shed light on the effectiveness and applicability of these methodologies in addressing the challenges posed by complex graph structures. [1]

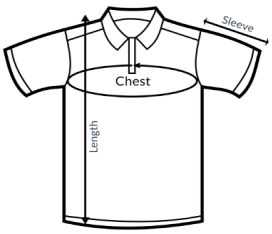
2 Unsupervised Learning

Unsupervised learning techniques have revolutionized the field of data analysis by enabling the discovery of hidden patterns and structures within complex datasets without the need for explicit labels or guidance. Within this realm, the application of unsupervised learning to graph clustering and bipartitioning has garnered significant attention due to its potential to unveil the intricate relationships and communities embedded within graph data.

Our investigation encompasses both theoretical underpinnings and practical considerations, aiming to provide a holistic understanding of the strengths, limitations, and applicability of some unsupervised learning techniques in the context of graph analysis. Furthermore, we conduct a comparative analysis of these algorithms on various benchmark datasets, shedding light on their performance across different graph types, sizes, and structural characteristics.

Clustering, often known as Unsupervised Learning is the process of organizing data instances into groups whose members are similar in some way. A cluster is therefore a collection of data instances which are "similar" to each other and are "dissimilar" to instances in other clusters. In the clustering literature, a data instance is also called an object as the instance may represent an object in the realworld. It is also called a data point as it can be seen as a point in an rdimension space, where r is the number of attributes in the data.[2]

Example : Tailoring T-shirt Sizes with Clustering



POLO T-SHIRTS
SIZE GUIDE

In Inches	S	M	L	XL	2XL
Chest	38	40	42	44	46
Length	26	27	28	29	30
Sleeve	7	7.5	8	8.5	9

Sizes may rarely vary in the range of +/- 0.5 inches

Figure 1

In the production and sale of T-shirts, finding the right balance between cost-effectiveness and customer satisfaction is crucial. Tailoring each T-shirt to individual customer sizes is expensive, while producing only one size may not meet the diverse needs of customers. Clustering offers a solution by grouping customers based on their sizes and creating generalized sizes of T-shirts for each group.This is why we see small, medium and large size T-shirts in shopping malls, and seldom see T-shirts with only a single size.

The process is usually as follows:

- The T-shirt manufacturer first samples a large number of people and measure their sizes to produce a measurement database.
- It then clusters the data, which partitions the data into some similarity subsets, i.e., clusters.
- For each cluster, it computes the average of the sizes and then uses the average to mass-produce T-shirts for all people of similar size.

Example : Social Network Analysis

Figure 2

Imagine a large social media platform where millions of users interact by following, friending, or messaging each other. The interactions between users can be represented as a graph, where each user is a node, and the connections between them (friendships, follows, messages) are edges.

- The platform aims to understand the community structure within its user base. Communities may represent groups of users with shared interests, geographic proximity, or similar behavior.
- Unsupervised learning techniques can be applied to partition the user graph into clusters. These algorithms group together users who are densely connected within their cluster while having sparse connections to users in other clusters.
- By identifying clusters in the social network graph, the platform can gain insights into user behavior, identify influential users, personalize content recommendations, and detect anomalous activities or communities.

2.1 K - Means Clustering

The k-means algorithm is the best known partitional clustering algorithm.[3] It is perhaps also the most widely used among all clustering algorithms due to its simplicity and efficiency. Given a set of data points and the required number of k clusters (k is specified by the user), this algorithm iteratively partitions the data into k clusters based on a distance function.

Let the set of data points (or instances) D be $x_1, x_2, \ldots, x_n$, where $x_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_r})$ is a vector in a real-valued space $X \subseteq \mathbb{R}^r$, and r is the number of attributes in the data (or the number of dimensions of the data space). The k-means algorithm partitions the given data into k clusters. Each cluster has a cluster center, which is also called the cluster centroid. The centroid, usually used to represent the cluster, is simply the mean of all the data points in the cluster, which gives the name to the algorithm, i.e., since there are k clusters, thus k means.

The k-means algorithm can be used for any application data set where the mean can be defined and computed. In Euclidean space, the mean of a cluster is computed with:

$$m_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

where $|C_j|$ is the number of data points in cluster C_j. The distance from a data point x_i to a cluster mean (centroid) m_j is computed with,

$$dist(x_i, m_j) = ||x_i - m_j|| = \sqrt{\sum_{k=1}^{r}(x_{ik} - m_{ik})^2} \text{ : Euclidean Distance}$$

Algorithm 1: K - Means Clustering

Input: Data items are points in n dimensions.
Fix the number of partitions K in advance

Output: Partition into K clusters.
Each cluster is represented by its geometric centre, i.e centroid or mean.

```
1 Choose k points initially at random
2 In each iteration,
3      Assign each point to nearest centroids
4      Recompute centroids
5 Termination Step,
6      Clusters stabilize
7      Sum of squares of distances of points from centroid in each cluster
   is below threshold
```

2.2 DBSCAN: Density Based Spatial Clustering of Applications with Noise

In this section, we present the algorithm DBSCAN (Density Based Spatial Clustering of Applications with Noise) which is designed to discover the clusters and the noise in a spatial database.[4]

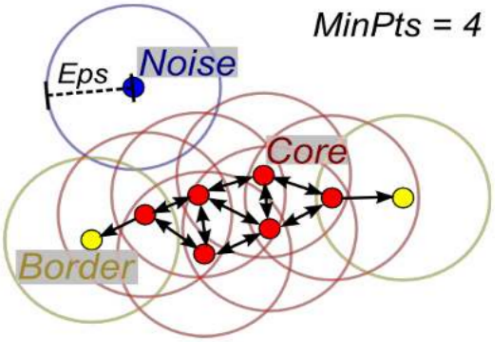


Figure 2.2.1: DBSCAN

Firstly, we construct a small ball around each point, radius `Eps` then identify a threshold for neighbours within ball, `MinPts`. The *Core point* has at least `MinPts` neighbours inside `Eps` ball. Now, connect each core point to all its neighbours such that, *Border points* that are attached to core points but not core themselves are part of this graph, but cannot add edges to extend the graph. The Noise are disconnected points. Formally, edges from core points to neighbours define a directed graph. Finally, discarding the edge directions, we are left with connected components which are clusters.

Algorithm 2: DBSCAN (SetOfPoints, Eps, MinPts)

Input: Data items are points in n dimensions.
Fix `Eps` and `MinPts` in advance

Output: Partition into clusters.

```
1  // SetOfPoints is UNCLASSIFIED
2  ClusterId := nextId(NOISE);
3  FOR i FROM 1 TO SetOfPoints.size DO
4       Point := SetOfPoints.get(i);
5       IF Point.CiId = UNCLASSIFIED THEN
6          IF ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts) THEN
7          ClusterId := nextId(ClusterId)
8          END IF
9       END IF
10 END FOR
11 END; // DBSCAN
```

3 Graph Partitioning

Graph partitioning is the reduction of a graph to a smaller graph by partitioning its set of nodes into mutually exclusive groups. Edges of the original graph that cross between the groups will produce edges in the partitioned graph. Graph Partitioning finds applications in various fields today. Data Clustering, Scientific computing, circuit partitioning for various stages of VLSI design and task scheduling in multiprocessor systems are some of the important areas for which the problem of graph partitioning is a very central one.

The problem of graph partitioning or graph clustering refers to a general class of problems that deals with the following task: given a graph $G = (V, E)$, group the vertices of a graph into groups or clusters or communities. (One might be interested in cases where this graph is weighted, directed, etc., but for now let's consider non - directed, possibly edges with equally weighted, graphs. Dealing with weighted graphs is straightforward, but extensions to directed graphs are more problematic.)

3.1 Notations

To begin, we define terms and notation that will be used in the rest of this document.[5]

A graph in this document will always refer to a simple, undirected graph $G = (V, E)$ which contains a set of vertices V and a set of edges E. Edges connect vertices and are labeled by their endpoints: for $u, v \in V$, if an edge connects vertices u and v, then $(u, v) \in E$. Vertices will each be assigned a unique positive integer n, where $1 \leq n \leq |V|$. That is, the vertices will be labeled with numbers $1, 2, \ldots, |V|$, with no two vertices sharing the same label.

Definition 2.1.1

A *graph* is a collection of vertices (nodes or points) connected by edges (line segments).

Definition 2.1.2

A graph is *simple* if has no multiple edges, (meaning two vertices can only be connected by one edge) and no loops (a vertex cannot have an edge connecting it to itself).

Definition 2.1.3

A graph is *connected* if it is in one single connected piece. All the graphs we will look at will be simple connected graphs.

Definition 2.1.4

The *degree* of a vertex v, written $d(v)$, is equal to the number of edges incident on v, i.e, the number of edges in E of the form (v, k), where $k \in V$.

Given a graph $G = (V, E)$, multiple matrices can be defined from it. Notice that the labels of vertices are being uses as matrix indices.

Definition 2.1.5

The *adjacency matrix* of G, $A(G) = (a_{ij})$ is defined such that,

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{if } (i,j) \notin E \end{cases}$$

For a simple graph G with n vertices the adjacency matrix is the $n \times n$ symmetric matrix A. In some context, if it is a weighted graph then for $(i, j) \in E$, $a_{ij} = w(i, j)$, where $w(i, j)$ is the weight assigned to the edge connecting vertices i and j. In this document, every edge weight is 1.

Definition 2.1.6

The *degree matrix* of G, $D(G) = (a_{ij})$ is defined such that,

$$d_{ij} = \begin{cases} d(i) & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

For a simple graph G with n vertices the degree matrix is the $n \times n$ diagonal matrix D.

Definition 2.1.6

The *Incidence matrix* of G, $H(G) = (H_{v,e})$ is defined such that,

$$H_{v,e} = \begin{cases} 1 & \text{if } v \text{ is adjacent to } e, \\ 0 & \text{otherwise} \end{cases}$$

For a simple graph $G(V, E)$ with $|V|(= n)$ vertices and $|E|$ edges, the *Incidence matrix, H* is of the order $|V| \times |E|$.

Definition 2.1.7

The *Laplacian matrix* of the graph G is the matrix $L(G) = (l_{ij})$, defined such that,

$$L(G) = D(G) - A(G)$$

The term *Laplacian matrix* for a graph is actually very general. There are lots of different Laplacian matrices, this one is by far the most common and is technically *the unnormalized graph Laplacian matrix* but since it's the only one we will look at we will simply called it the *Laplacian matrix*.

A Toy Example

One way we might immediately describe this graph is that it is a square connnected to a triangle. What we are doing when we see this is we are breaking the graph into those two subgraphs.

This process, of breaking a graph into two or more subgraphs, has generic uses when analyzing networks. Consequently what we'd like to know is if there is a way of doing this easily.

In order to investigate we first need some more definitions.

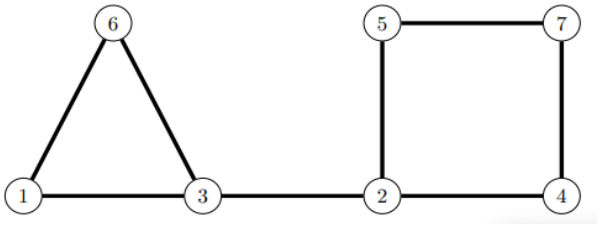


Figure 3.1.1: A simple connected graph with $|V| = 7$ and $|E| = 8$

Definition 2.1.8

Given a graph G with n vertices $V = 1, 2, ..., n$ For an integer $k \geq 2$ a *k-partition* of G is an partition of the vertices into into k subsets $V_1, \ldots, V_k$ such that the subsets do not overlap and their union is all of V. We will write $P = (V_1, V_2, ..., V_k)$. A 2-*partition* is often just called a *partition*.

Definition 2.1.9

For a partition $P = (V_1, V_2)$ of a graph G we define the *cut* of P, denoted $cut(P)$, as the number of edges joining a vertex in V_1 with a vertex in V_2.

Definition 2.1.10

A *minimum cut* is a partition P of a graph G in a manner that minimizes $cut(P)$. In other words, it's the minimum number of edges we need to remove to partition the graph.

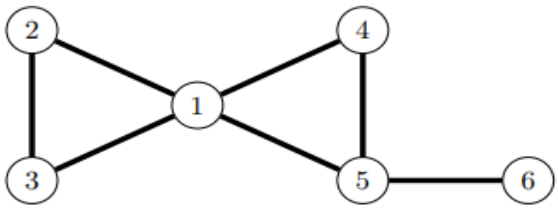


Figure 3.1.2: A simple connected graph with $|V| = 6$ and $|E| = 7$

One problem with a minimum cut is that if there is a stray vertex connected to the rest of the graph by one edge then this would be a minimum cut. This tends to leave the subgraphs unbalanced which is somewhat unsatifactory.

3.2 Partitioning Problem

Now the graph partitioning problem [6] in general can be stated as follows:

> Given a graph $G = (V, E)$, where V is the set of nodes $\{1, 2, \ldots, |V|\}$ and E the set of edges, construct a 2 - way partitioning P_2 which divides the nodes into a set of 2 disjoint subsets $P_2 = \{G_1, G_2\}$ so as to optimize some objective function $f(P_2)$.

To fulfill the bi-partitioning task, i.e. to divide vertices into two disjoint groups G_1 and G_2. Some basic questions comes up in this context.[7]

- How can we define a *good partition* of $G(V, E)$?
- How can we efficiently identify such a partition?
- What makes a *good partition*?

A *good partition* is to *maximize the number of within group connections* and to *minimize the number of between group connections*. A very common objective function is merely the *size of the cut*. Minimum such cut however, may divide the vertex set very unevenly and this is undesirable. There are, in literature, objective functions which measure the size balancing effect of a partition. They are described below.

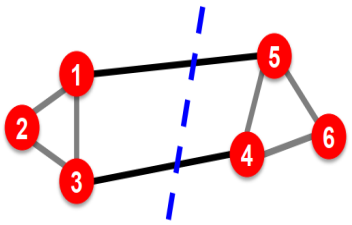


Figure 3.2.1: An example of optimal graph partition.

3.3 Min - Max Cut

Given an undirected, simple graph $G = G(E, V)$ with node set V, edge set E and weight matrix W (in our case, Weight matrix, W is equivalent to Adjacency matrix, A as all the edge weights are same i.e, $W_{uv} = 1$), we wish to partition it into two subgraphs G_1 and G_2 using the min-max clustering principle [8] - minimize similarity between clusters and maximize similarity within a cluster. The similarity or association between two nodes is their edge weight, $W_{uv} = 1$ (in our case). Thus the similarity between subgraphs G_1, G_2 is the cut size:

$$cut(G_1, G_2) = W(G_1, G_2)$$

where, $W(G_1, G_2) = \sum_{u \in G_1, v \in G_2} W_{uv}$; and $W(G_1) \equiv W(G_2)$ Note, $W(G_1)$ is the similarity or association within a cluster (subgraph G_1) is the sum of all edge weights within G_1 (i.e, number of edges in G_1). Thus, the *min - max clustering principle* requires that we minimize $cut(G_1, G_2)$ while maximizing $W(G_1)$ and $W(G_2)$ at the same time. All these requirements can be simultaneously satisfied by the following objective function,

$$Mcut = \frac{cut(G_1, G_2)}{W(G_1)} + \frac{cut(G_1, G_2)}{W(G_2)}$$

Let $\mathbf{x}$ and $\mathbf{y}$ be vectors conformally partitioned with A and B, i.e., $\mathbf{x} = (1, \ldots, 1, 0, \ldots, 0)^T$ and $\mathbf{y} = (0, \ldots, 0, 1, \ldots, 1)^T$. Then,

$$cut(G_1, G_2) = \mathbf{x}^T(D - A)\mathbf{x} = \mathbf{y}^T(D - A)\mathbf{y} = \mathbf{x}^T L\mathbf{x} = \mathbf{y}^T L\mathbf{y}$$

$$W(G_1) = \mathbf{x}^T A\mathbf{x} \text{ and } W(G_2) = \mathbf{y}^T A\mathbf{y}$$

Hence, the objective function can be rewritten as,

$$Mcut = \frac{\mathbf{x}^T L\mathbf{x}}{\mathbf{x}^T A\mathbf{x}} + \frac{\mathbf{y}^T L\mathbf{y}}{\mathbf{y}^T A\mathbf{y}}$$

$Mcut$ is invariant under changes of $||\mathbf{x}||_2$ and $||\mathbf{y}||_2$, and $\mathbf{x}^T D\mathbf{y} = 0$ and $\mathbf{x}^T L\mathbf{x} > 0$, $\mathbf{y}^T W\mathbf{y} > 0$. It is observed that the above objective function can be relaxed into the following optimization problem,

$$Mcut = \min_{\mathbf{x},\mathbf{y}} \frac{\mathbf{x}^T L\mathbf{x}}{\mathbf{x}^T A\mathbf{x}} + \frac{\mathbf{y}^T L\mathbf{y}}{\mathbf{y}^T A\mathbf{y}}$$

subject to, $||\mathbf{x}||_2 = 1$ and $||\mathbf{y}||_2 = 1$, & $\mathbf{x}^T\mathbf{y} = 0$ and $\mathbf{x}^T M\mathbf{x} > 0$, $\mathbf{y}^T M\mathbf{y} > 0$ where, $M = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

3.4 Normalised Cut

Given an undirected, simple graph $G = G(E, V)$ with node set V, edge set E, we wish to partition the graph into two sets G_1 and G_2, let x be an $N = |V|$ dimensional indicator vector,

$$x_i = \begin{cases} 1 & \text{if node } i \in G_1 \\ -1 & \text{if node } i \in G_2 \end{cases}$$

Let $d(i) = \sum_j W(i,j)$ be the total connection from node i to all other nodes. Then,

$$\begin{aligned} Ncut &= \frac{cut(G_1, G_2)}{vol(G_1)} + \frac{cut(G_2, G_1)}{vol(G_2)} \\ &= \frac{\sum_{x_i>0, x_i<0} -w_{ij}x_i x_j}{\sum_{x_i>0} d(i)} + \frac{\sum_{x_i<0, x_i>0} -w_{ij}x_i x_j}{\sum_{x_i<0} d(i)} \end{aligned}$$

Like *Mcut*, here we obtain,

$$Ncut(x) = \min_{\mathbf{y}} \frac{\mathbf{y}^T L \mathbf{y}}{\mathbf{y}^T D \mathbf{y}} \text{ subject to } y_i \in \{1, -1\} \text{ and } \mathbf{y}^T D \mathbf{1} = 0$$

The above expression is the Rayleigh quotient for *Ncut*. If y is relaxed to take on real values, we can minimize the above equation by solving the generalized eigenvalue system,

$$L\mathbf{y} = \lambda D \mathbf{y}$$

3.5 Ratio Cut

An alternative to Normalised cut is the Ratio cut [9], which can be trivially adapted to solve clustering problems. The idea of *Ratio cut* is to replace the volume $vol(G_j)$ of each subgraph G_j of the partition by its size $|G_j|$ (the number of nodes in G_j). It is defined as,

$$Rcut = \frac{cut(G_1, G_2)}{|G_1|} + \frac{cut(G_2, G_1)}{|G_2|}$$

Therefore, ratio cut is the special case of normalized cut where $D = I$. So, the optimization problem is expressed as,

$$Rcut(x) = \min_{\mathbf{y}} \frac{\mathbf{y}^T L \mathbf{y}}{\mathbf{y}^T \mathbf{y}} \text{ subject to } y_i \in \{1, -1\} \text{ and } \mathbf{y}^T \mathbf{1} = 0$$

Key Points to note...

The objective function of graph partitioning depends on two key points,

- Minimizing the number of edges between the two subgraphs.

$$f_{obj} \; \alpha \; cut(G_1, G_2)$$

- Partitions in such a way that the subgraphs have a nearly equal number of vertices (as close to equal as is possible)

$$f_{obj} \; \alpha \; \frac{1}{|G_i|} \text{ or } \frac{1}{vol(G_i)}$$

3.6 Comparison

1. Min - max Cut:

   Objective

   Minimize the maximum number of edges between any two partitions.

   Pros

   - Simple and intuitive objective.
   - Tends to produce partitions with fewer edges crossing between them.

   Cons

   - May lead to imbalanced partition sizes.
   - Does not consider the overall connectivity of the partitions.

2. Normalized Cut:

   Objective

   Minimize the normalized cut cost, which considers both intra-cluster similarity and inter-cluster dissimilarity.

   Pros

   - Considers both the balance of partitions and the overall connectivity.
   - Tends to produce more balanced partition sizes.

   Cons

   - Computationally more expensive compared to min-max cut.
   - Sensitive to noise and outliers in the data.

3. Ratio Cut:

   Objective

   Minimize the ratio of the number of edges between partitions to the total number of edges within partitions.

   Pros

   - Measures the ratio of inter-cluster communication to intra-cluster communication.
   - Tends to produce balanced partitions.

   Cons

   - Similar to normalized cut, it is computationally more expensive than min-max cut.
   - Sensitivity to noise and outliers in the data.

Min-max cut tends to produce partitions with fewer edges crossing between them but may result in imbalanced partition sizes. *Normalized cut* and *ratio cut* take into account both the balance of partitions and the overall connectivity, resulting in more balanced partition sizes. However, they are *computationally more expensive* compared to min-max cut.

3.7 Complexity

Note, that in the above mentioned methods, we consider two sets of vertices, which can be viewed as subgraphs and try to compute the optimized value of the objective function, and then the minimum of all is our required cut.

The exact computation of cut - based partitioning criteria is often *NP-hard*, approximation algorithms are commonly used to find approximate solutions in polynomial time. The time

complexity of these algorithms depends on the specific method used and the size of the graph. For very large graphs, *scalability* becomes a significant consideration.

Since graph partitioning is a hard problem, most solution approaches are based on heuristics. There is a wide variety of heuristics that have been suggested and tried. Broadly, partitioning methods fall into two categories. The first variety includes those that work *locally*. The other set of methods are those that look at the *global connectivity* information while partitioning.

Among global methods, the one that based on *spectrum* of associated matrices work quite well. Now, in the next section, we will describe the method of Spectral Graph Partitioning and see how it outperforms the previous algorithms.

4 Spectral Graph Partitioning

Spectral graph partitioning is a method of partitioning a graph into two subgraphs in such a way that the subgraphs have a nearly equal number of vertices (as close to equal as is possible) while also minimizing the number of edges between the two subgraphs. The *spectrum* of a matrix is the set of that matrix's eigenvalues; therefore, spectral partitioning of a graph uses the eigenvalues of a matrix associated with the graph to perform the optimal partition.

We present the theory and method used by Fiedler to perform a spectral graph partition and restate the proof of his theorem that the subgraphs generated by a spectral graph partition are, in fact, connected.[10]

4.1 Methodology

Fiedler's theory of spectral graph partitioning is based upon a very simple idea. For a graph $G = (V, E)$ which is to be partitioned, the Laplacian matrix $L(G)$ is formed, and the eigenvectoreigenvalue pairs of $L(G)$ are calculated. Label the eigenvalues so that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. The eigenvector x_2, which corresponds to λ_2 (the second-smallest eigenvalue), is known as the *Fiedler vector*, and is used to partition the vertices.[11]

Let $n = |V|$. Recall that each of the n vertices of G is assigned a label, which is a unique number in the set $\{1, 2, \ldots, n\}$. Also note that $L(G) \in \mathbb{R}^{n \times n}$ and that $x_2 \in \mathbb{R}^n$. Therefore each vertex corresponds to a single entry of x_2. More precisely, the vertex with label i corresponds to entry i of x_2, which we will call x_i. To form the partition, simply create two graphs G_1 and G_2, and for each vertex i of G, if $x_i < 0$, put vertex i in G_1; otherwise, put it in G_2. This defines a partition of G into two sets, but just by examining this method, it is not evident that this partition attempts to minimize the number of edges between G_1 and G_2, or even that the two subgraphs are connected.

Algorithm 3: Fiedler Partitioning

Input: Graph G

Output: Subgraph G_1 and G_2

```
Compute the Laplacian matrix L of the graph G.
Compute the eigenvectors and eigenvalues of L.
Sort the eigenvalues in ascending order, and let λ2 be the second smallest
 eigenvalue.
Let x2 be the corresponding eigenvector associated with λ2
Let n = |V|, the number of vertices in G
Initialize two empty subgraphs G1 and G2
For each vertex i in G:
     If x2[i] < 0, add vertex i to subgraph G1
     Otherwise, add vertex i to subgraph G2
Return the partitioned subgraphs G1 and G2
```

4.2 Reasoning for Correctness of Partition

Why using the second-smallest eigenvalue effectively divides the matrix?

To begin, it is not even clear at first why using the second-smallest eigenvalue would produce such a partition. In fact, the use of eigenvectors and eigenvalues to perform a graph partition is not at all an obvious thing to try.

To explain this, Demmel uses an example of a vibrating string. When a string is plucked, waves propogate along its length at a certain frequency. If the frequency causes the wave to appear as if it does not move, the wave is called a standing wave. If the length of the string is l, this will occur when the wavelength is $\frac{2l}{n}$ for $n \in \mathbb{N}$. It turns out that the frequencies that generate these standing waves are tied directly to the eigenvalues and eigenvectors of the matrix that describes the motion of individual points along the string. (The points are the vertices and the section of string between two points forms an edge, so the string is thought of as a chain graph; the matrix that is used is the Laplacian matrix of this chain graph multiplied by a spring constant k which does not affect the eigenvectors or which eigenvalue is the secondsmallest.) A more in-depth analysis of the physics involved in determining the eigenvectors is provided by Demmel at [12]

A string vibrating at a frequency related to the second-smallest eigenvalue and its corresponding eigenvector of a Laplacian graph will vibrate with a standing wave equal to this one: a sine wave whose wavelength is equal to the length of the string. Half of the string is above the equilibrium line, corresponding to points with positive entries in the eigenvector, and the other half is below it, corresponding to points with negative entries.

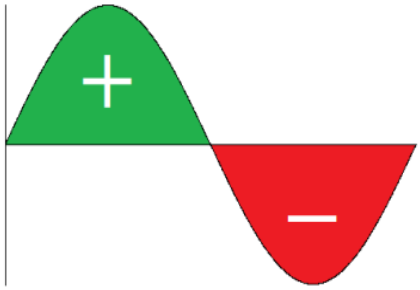


Figure 4.2.1: A vibrating string

The second-smallest eigenvalue corresponds to a standing wave that is equal to one whole wavelength. As shown in the above figure, this results in a wave with half of the points below the equilibrium point, and half above. The value of the entry for each vertex in the corresponding eigenvector determines the "height" of the vertex above or below the line. Using the sign of this "height" as the tool of determining which subgraph to place the vertices in, it is apparent that about half of the vertices would go into each subgraph. The actual number depends on their position on the string. For general graphs, the same analogy applies, but in a more complicated way. Demmel suggests visualizing a surface for the case of planar graphs; about one half of the surface would be below an equilibrium plane and the other half would be above it.

The connectedness of the subgraphs

The most important result proved by Fiedler is that the two subgraphs resulting from the spectral partition are always connected. Before getting into his proof, we will follow Demmel once again and introduce some preliminary definitions and lemmas.

4.3 Fiedler's Theorem of Connectivity of Spectral Graph Partitions

The Main Theorem

Let G be a connected graph and let $L(G)$ be its Laplacian matrix. Create the subgraphs G_1 and G_2 using the method described in Section 4.1. The two subgraphs G_1 and G_2 are both connected.

First we state a few definitions, lemmas and theorems that we will need to prove the above main theorem.

Definition 4.3.1

A matrix A is *non-negative*, denoted by $A \geq 0$, if, for every element a_{ij}, we have $a_{ij} \geq 0$. Similar notions exist for *positive*, *non-postive*, and *non-negative* matrices using the appropriate relational symbols. This definition can also be extended to vectors.

Definition 4.3.2

The *spectral radius* of an $n \times n$ matrix A, denoted by $\rho(A)$, is given by $\rho(A) = \max_{i=1}^{n} |\lambda_i|$, where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of A. That is, $\rho(A)$ is the eigenvalue of A with maximum absolute value.

Definition 4.3.3

An $n \times n$ matrix A is called *positive-definite* if, for all vectors $\mathbf{x} \in \mathbb{R}^n$ where $\mathbf{x} \neq 0$, $\mathbf{x}^T A\mathbf{x} > 0$. The matrix A is *positive-semidefinite* if $\mathbf{x}^T A\mathbf{x} \geq 0$.

Lemma 4.3.1

The matrix A is symmetric and positive-definite if and only if every eigenvalue of A is positive. Also, A is symmetric and positive-semidefinite if and only if every eigenvalue of A is nonnegative.

Proof:

This can be shown by performing the eigenvalue decomposition of A:

$$A = Q\Lambda Q^{-1}, \text{ where } Q^T = Q^{-1}$$

Letting $\mathbf{z} \in \mathbb{R}^n$ be any nonzero vector, $\mathbf{z}^T A\mathbf{z} > 0$ if and only if,

$$\mathbf{z}^T(Q\Lambda Q^{-1})\mathbf{z} = (Q\mathbf{z})^T\Lambda(Q\mathbf{z}) > 0$$

Since Λ is a diagonal matrix, this will always be true only if each of its diagonal elements is positive. Since Λ contains the eigenvalues of A, each eigenvalue must be positive. A similar argument works for showing positive semi-definiteness.

Lemma 4.3.2

If A is an $n \times n$ symmetric matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, then $\lambda_1 = \min_{v \neq 0} \frac{\mathbf{v}^T A\mathbf{v}}{\mathbf{v}^T\mathbf{v}}$ and $\lambda_1 = \max_{\mathbf{v} \neq 0} \frac{\mathbf{v}^T A\mathbf{v}}{\mathbf{v}^T\mathbf{v}}$. That is, λ_1 and λ_n are the minimum and maximum values of the expression $\frac{\mathbf{v}^T A\mathbf{v}}{\mathbf{v}^T\mathbf{v}}$ taken over all nonzero vectors $\mathbf{v} \in \mathbb{R}^{n \times n}$.

Proof:

Perform the eigenvalue decomposition of A: $A = Q\Lambda Q^{-1}$, where $Q^{-1} = Q^T$, meaning that this can also be written as $A = Q\Lambda Q^T$. Note that $QQ^T = I_n$. Now,

$$\frac{\mathbf{v}^T A\mathbf{v}}{\mathbf{v}^T\mathbf{v}} = \frac{\mathbf{v}^T Q\Lambda Q^T\mathbf{v}}{\mathbf{v}^T QQ^T\mathbf{v}}$$

Let $\mathbf{y} = Q^T\mathbf{v}$. Then, this expression is equal to $\mathbf{y}\Lambda\mathbf{y}^T = \mathbf{y}\mathbf{y}^T$, which is equal to $\lambda_1 y_1^2 + \lambda_2 y_2^2 + \ldots + \lambda_n y_n^2$. Let this expression be equal to α. Then, since $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, we have that

$$\alpha \geq \frac{\lambda_1 y_1^2 + \lambda_1 y_2^2 + \ldots + \lambda_1 y_n^2}{y_1^2 + y_2^2 + \ldots + y_n^2} \quad \text{(with all instances of } \lambda_i \text{ replaced by } \lambda_1\text{)}$$

and

$$\alpha \leq \frac{\lambda_n y_1^2 + \lambda_n y_2^2 + \ldots + \lambda_n y_n^2}{y_1^2 + y_2^2 + \ldots + y_n^2} \quad \text{(with all instances of } \lambda_i \text{ replaced by } \lambda_n\text{).}$$

Recall that $\mathbf{y} = Q^T\mathbf{v}$. If $\mathbf{v} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$, then $\alpha = \lambda_1$. Also, if $\mathbf{v} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$, then $\alpha = \lambda_n$.

Lemma 4.3.3

Let A be an $n \times n$ matrix and let X be any $n \times n$ non-singular matrix. Then, X^TAX is symmetric positive definite if and only if A is symmetric positive definite.

Proof:

From Lemma 4.3.2, the smallest eigenvalue of X^TAX is $\min_{\mathbf{v}\neq 0} \frac{\mathbf{v}^TX^TAX\mathbf{v}}{\mathbf{v}^T\mathbf{v}}$. From here,

$$\min_{\mathbf{v}\neq 0} \frac{\mathbf{v}^TX^TAX\mathbf{v}}{\mathbf{v}^T\mathbf{v}} = \min_{\mathbf{v}\neq 0} \frac{\mathbf{v}^TX^TAX\mathbf{v}}{\mathbf{v}^TX^TX\mathbf{v}} \cdot \frac{\mathbf{v}^TX^TAX\mathbf{v}}{\mathbf{v}^T\mathbf{v}},$$

Now, Introducing a factor of $\frac{\mathbf{v}^TX^TAX\mathbf{v}}{\mathbf{v}^TX^TX\mathbf{v}}$.

$$\geq \min_{\mathbf{v}\neq 0} \frac{\mathbf{v}^TX^TX\mathbf{v}}{\mathbf{v}^TX^TX\mathbf{v}} \cdot \min_{\mathbf{v}\neq 0} \frac{\mathbf{v}^TX^TX\mathbf{v}}{\mathbf{v}^T\mathbf{v}}$$

The two factors could be smaller using different vectors $\mathbf{v}$:

$$= \min_{\mathbf{y}\neq 0} \frac{\mathbf{y}^TA\mathbf{y}}{\mathbf{y}^T\mathbf{y}} \cdot \min_{\mathbf{v}\neq 0} \frac{\mathbf{v}^TX^TX\mathbf{v}}{\mathbf{v}^T\mathbf{v}},$$

where $\mathbf{y} = X\mathbf{v}$.

$$= \lambda_1(A) \cdot \lambda_1(X^TX) \quad \text{by Lemma 4.3.2}$$

The notation $\lambda_1(M)$ refers to the smallest eigenvalue of matrix M. The number $\mathbf{v}^TX^TX\mathbf{v}$ is an inner product of a vector with itself and is therefore nonnegative. The same is true of $\mathbf{v}^T\mathbf{v}$. Therefore, $\frac{\mathbf{v}^TX^TX\mathbf{v}}{\mathbf{v}^T\mathbf{v}}$ is nonnegative, and $\lambda_1(X^TX) \geq 0$. Also, since X is nonsingular, X^TX is also nonsingular, and therefore cannot have an eigenvalue of 0 (since the determinant of a matrix is the product The chain of equations and inequalities shows that the smallest eigenvalue of X^TAX is equal to $\lambda_1(A) \cdot \lambda_1(X^TX)$. The matrix A is positive-definite, so $\lambda_1(A) > 0$, and it was just stated that $\lambda_1(X^TX) > 0$. So the smallest eigenvalue of X^TAX is positive, and therefore so are all of its eigenvalues, making it positive-definite. To show the other direction, follow the equations and inequalities in the other direction to show that the smallest eigenvalue of X^TAX is positive.

Lemma 4.3.4

Let A be an $n \times n$ symmetric matrix. If $\rho(A) < 1$, meaning that the value of any eigenvalue of A is on the interval $(-1, 1)$, then $I_n - A$ is nonsingular and $(I_n - A)^{-1} = \sum_{i=0}^{\infty} A^i$.

Proof:

The eigenvalues of the matrix $I_n - A$ are on the interval $(0, 2)$ since, every eigenvalue of I_n is 1 and the eigenvalues of A are on the interval $(-1, 1)$. This is true only because I_n is diagonal and possibly because A is symmetric, and is not true in general cases. Since the eigenvalues of $I_n - A$ are all positive, $I_n - A$ is nonsingular. With the eigenvalue decomposition $A = Q\Lambda Q^T$, we see that

$$A^i = (Q\Lambda Q^T)(Q\Lambda Q^T)\cdots(Q\Lambda Q^T) \quad (i \text{ times}) = Q\Lambda^iQ^T$$

since, $QQ^T = I_n$ and the Q^T at the end of one term cancels with the Q at the beginning of the next term. The matrix Λ is diagonal, meaning that each diagonal element of Λ^i is

equal to the corresponding element of Λ multiplied by itself i times. Since, each element of Λ, $|\lambda_i| < 1 \forall i = 1, 2, \ldots, n$, Λ^i goes to a matrix where every element is 0 as i goes to ∞, and therefore A^i does as well. Therefore, $\sum_{i=0}^{\infty} A^i$ is convergent. Now notice that

$$(I_n - A) \cdot \sum_{i=0}^{m} A^i = (I_n - A) + (A - A^2) + (A^2 - A^3) + \cdots + (A^m - A^{m+1}) = I_n - A^{m+1},$$

since it is a telescoping series. As m goes to ∞, $I_n - A^{m+1} = (I_n - A) \cdot \sum_{i=0}^{m} A^i$ goes to I_n, and therefore $\sum_{i=0}^{\infty} A^i$ converges to $(I_n - A)^{-1}$.

Theorem 4.3.1: Cauchy Interlacing Theorem

Given an $n \times n$ matrix A, let $A(i : j; i : j)$ be the $(j - i + 1) \times (j - i + 1)$ submatrix of A that consists only of rows $i, i + 1, \ldots, j$ and columns $i, i + 1, \ldots, j$ of A. Let A be symmetric and let its eigenvalues be $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Further, let the eigenvalues of $A(i : j; i : j)$ be $\chi_1 \leq \chi_2 \leq \ldots \leq \chi_{j-i+1}$. Then, for some $k \leq j - i + 1$, the matrix A has at least k eigenvalues less than or equal to χ_k.

Proof:

The proof of this can be found in *Matrix Analysis* by R. Horn and C. Johnson (1988).

Corollary :

If A is positive-definite, then $A(i : j; i : j)$ is also positive-definite since all eigenvalues of A are positive and, by using the theorem with $k = 1$, $\lambda_1 \leq \chi_1$. Using the facts that $\lambda_1 > 0$ and $\chi_1 \leq \chi_2 \leq \ldots \leq \chi_{j-i+1}$, we have that $0 < \lambda_1 \leq \chi_1 \leq \chi_2 \leq \ldots \leq \chi_{j-i+1}$ and therefore all eigenvalues of $A(i : j; i : j)$ are positive, making $A(i : j; i : j)$ positive-definite from Remark 4.3.1.

With all of these facts now shown, we can prove the important theorem that the subgraphs created from a spectral graph partition are connected.

Fiedler's Theorem of Connectivity of Spectral Graph Partitions

Let G be a connected graph and let $L(G)$ be its Laplacian matrix. Create the subgraphs G_1 and G_2 using the method described in Section 4.1. The two subgraphs G_1 and G_2 are both connected.

Proof:

To prove this, we use a proof by contradiction. Assume, to the contrary, that G_1 is composed of 2 connected components. Let $\mathbf{x}$ be the eigenvector corresponding to λ_2, the second-smallest eigenvalue of $L(G)$. Both $L(G)$ and $\mathbf{x}$ can be written in a block form:

$$L(G) = \begin{bmatrix} \mathbf{L}_{1,1} & \mathbf{O} & \mathbf{L}_{1,3} \\ \mathbf{O} & \mathbf{L}_{2,2} & \mathbf{L}_{2,3} \\ \mathbf{L}_{1,3}^T & \mathbf{L}_{2,3}^T & \mathbf{L}_{3,3} \end{bmatrix} \text{ and } \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are positive and $\mathbf{x}_3$ is negative.

Here, we use simply $\mathbf{x}$ for the Fiedler vector, since the subscripts are reserved for the blocks of that vector. The symbol $\mathbf{O}$ represents a matrix of appropriate size which has 0 as every entry. Recall that $L(G)$ is *symmetric*, so the diagonal blocks are also *symmetric*, and the off-diagonal block in position (i, j) is the transpose of the block in position (j, i). Also, due to the nature of the Laplacian matrix, all off-diagonal blocks are nonpositive.

Due to the way the graph partition is constructed using the values of $\mathbf{x}$, this construction puts the vertices corresponding to the first 2 blocks of $\mathbf{x}$ into G_1, but the matrix $L(G)$ shows that there are no edges from any vertex of $\mathbf{x}_1$ to any vertex of $\mathbf{x}_2$, meaning that G_1 really does consist of 2 connected components.

Since $\mathbf{x}$ is an eigenvector of $L(G)$ and λ_2 is its eigenvalue, we have that

$$L(G)\ \mathbf{x} = \lambda_2\ \mathbf{x}$$

. Multiplying this out for only the first block x_1 of $\mathbf{x}$, we see that

$$\mathbf{L}_{1,1}\ \mathbf{x}_1 + \ \mathbf{L}_{1,3}\ \mathbf{x}_3 = \lambda_2\ \mathbf{x}_1$$

. Recall that every eigenvalue of $L(G)$ is non-negative. By the corollary to the *Cauchy Interlacing Theorem* (Theorem 4.3.1), every eigenvalue of $L_{1,1}$ is also non-negative.

Let $\epsilon > 0$. Add $\epsilon\ \mathbf{x}_1$ to both sides of the above equation, resulting in

$$\epsilon\ \mathbf{x}_1 + \ \mathbf{L}_{1,1}\ \mathbf{x}_1 + \ \mathbf{L}_{1,3}\ \mathbf{x}_3 = \epsilon\ \mathbf{x}_1 + \lambda_2\ \mathbf{x}_1$$
$$(\epsilon\ \mathbf{I} + \ \mathbf{L}_{1,1})\ \mathbf{x}_1 + \ \mathbf{L}_{1,3}\ \mathbf{x}_3 = (\epsilon + \lambda_2)\ \mathbf{x}_1,$$

where $\mathbf{I}$ is of appropriate size.

By the same properties of $\mathbf{I}$ and symmetric matrices as was mentioned at the beginning of the proof of Lemma 4.3.4,

$$\text{the eigenvalues of } \epsilon\ \mathbf{I} + \ \mathbf{L}_{1,1} \geq \epsilon > 0, \text{ so } \epsilon\ \mathbf{I} + \ \mathbf{L}_{1,1} \text{ is } \textit{positive - definite}.$$

Using *Demmel's method* of constructing the proof, let $\epsilon\ \mathbf{I} + \ \mathbf{L}_{1,1} = D - A$, where the matrix D is diagonal, and each diagonal element is equal to the corresponding element on the diagonal of $\epsilon\ \mathbf{I} + \ \mathbf{L}_{1,1}$, and where $-A$ contains the other (off-diagonal) elements of $\epsilon\ \mathbf{I} + \ \mathbf{L}_{1,1}$, and every diagonal element of $-A$ is 0.

This can be further decomposed into

$$D^{\frac{1}{2}}(I - (D^{\frac{1}{2}})^{-1}AD^{\frac{1}{2}})D^{\frac{1}{2}},$$

where $D^{\frac{1}{2}}$ is equal to the matrix D after every element of it has had the square root taken of it. It is true that $D^{\frac{1}{2}}D^{\frac{1}{2}} = D$ since D is diagonal and non-negative. Multiply this expression through to see that it really is equal to $D - A$.

Finally,

$$\text{Let } M = (D^{\frac{1}{2}})^{-1}A(D^{\frac{1}{2}})^{-1}, \text{ so that } \epsilon\ \mathbf{I} + \ \mathbf{L}_{1,1} = D - A = D^{\frac{1}{2}}(I - M)D^{\frac{1}{2}}$$

.

Recall that since $D^{\frac{1}{2}}$ is diagonal, $(D^{\frac{1}{2}})^T = D^{\frac{1}{2}}$. By Lemma 2.7, since

$$\mathbf{I} + \ \mathbf{L}_{1,1} = D^{\frac{1}{2}}(I - M)D^{\frac{1}{2}} = (D^{\frac{1}{2}})^T(I - M)D^{\frac{1}{2}} \text{ is positive-definite}$$

Then, $I - M$ is also *positive-definite*. By the same reasoning before involving eigenvalues of I and symmetric matrices, if λ is an eigenvalue of M, then $1 - \lambda$ is an eigenvalue of $I - M$. Each eigenvalue of $I - M$ is also greater than -1 by Lemma 4.3.2:

$$
\begin{aligned}
\lambda_1(M) &= \min_{v\neq 0} \frac{\mathbf{v}^T M \mathbf{v}}{\mathbf{v}^T\mathbf{v}} && \text{minimum eigen value of M}\\
&\geq \min_{\mathbf{v}\neq 0} \frac{-|\mathbf{v}|^T M |\mathbf{v}|}{|\mathbf{v}|^T|\mathbf{v}|} && \text{since } M \text{ the signs of the elements of } \mathbf{v}\\
&= -\max_{\mathbf{v}\neq 0} \frac{|\mathbf{v}|^T M |\mathbf{v}|}{\mathbf{v}^T\mathbf{v}} && \text{since } \min_{a\in S}(-a) = -\max_{a\in S} a\\
&\geq -\max_{\mathbf{v}\neq 0} \frac{|\mathbf{v}|^T M |\mathbf{v}|}{\mathbf{v}^T\mathbf{v}} && \text{since } M \text{ is positive and nothing is known about}\\
& && \text{the signs of the elements of } \mathbf{v}\\
&= -\lambda_{n_1}(M) && \text{by Lemma 4.3.2; here, } n1 \text{ is the number of elements in } x1\\
&> -1 && \text{all eigenvalues of } M \text{ are less than 1}
\end{aligned}
$$

So, the absolute value of every eigenvalue of M is less than 1.

Continuing to follow Demmel, let $Y = (\epsilon\ \mathbf{I} + \mathbf{L}_{1,1})^{-1}$. Recall that $\epsilon\ \mathbf{I} + \mathbf{L}_{1,1} = D^{\frac{1}{2}}(I-M)D^{\frac{1}{2}}$. Therefore,

$$
\begin{aligned}
Y = (D^{\frac{1}{2}}(I-M)D^{\frac{1}{2}})^{-1} &= (D^{\frac{1}{2}})^{-1}(I-M)^{-1}(D^{\frac{1}{2}})^{-1}\\
&= (D^{\frac{1}{2}})^{-1}\left(\sum_{i=0}^{\infty} M^i\right)(D^{\frac{1}{2}})^{-1} \text{ By Lemma 4.3.4}
\end{aligned}
$$

The matrix Y is nonnegative since M is nonnegative and $D^{\frac{1}{2}}$ is positive. In fact, since M is symmetric and describes a graph, M is positive and therefore so is Y.

Recall,

$$(\epsilon I + \mathbf{L}_{1,1})\mathbf{x}_1 + \mathbf{L}_{1,3}\mathbf{x}_3 = (\epsilon + \lambda_2)\mathbf{x}_1$$

Multiplying both sides of this by Y gives

$$\mathbf{x}_1 + Y\mathbf{L}_{1,3}\mathbf{x}_3 = (\epsilon + \lambda_2)Y\mathbf{x}_1$$

Since $Y(\epsilon I + \mathbf{L}_{1,1}) = I$. Further multiplying that equation by $\mathbf{x}_1^T$ gives.

$$\mathbf{x}_1^T\mathbf{x}_1 + \mathbf{x}_1^T Y\mathbf{L}_{1,3}\mathbf{x}_3 = (\epsilon + \lambda_2)\mathbf{x}_1^T Y\mathbf{x}_1$$

Once again, let n_1 be the number of elements in vector $\mathbf{x}_1$. This equation and Lemma 4.3.2 can be used to show that:

$$(\epsilon + \lambda_2)\lambda_{n_1}(Y) = \max_{\mathbf{v}\neq 0}\left\{(\epsilon + \lambda_2)\cdot\frac{\mathbf{v}^T Y\mathbf{v}}{\mathbf{v}^T\mathbf{v}}\right\} \quad \text{by Lemma 4.3.2}$$

Using $\mathbf{x}_1$ for the vector $\mathbf{v}$ could only result in a value that is no larger than the maximum obtained with any $\mathbf{v}$,

$$\geq (\epsilon + \lambda_2)\cdot\frac{\mathbf{x}_1^T Y^T Y\mathbf{x}_1}{\mathbf{x}_1^T\mathbf{x}_1}$$

Substituting $\mathbf{x}_1^T\mathbf{x}_1 + \mathbf{x}_1^T Y\mathbf{L}_{1,3}\mathbf{x}_3$ for $(\epsilon + \lambda_2)\mathbf{x}_1^T Y\mathbf{x}_1$ as in the above equation,

$$= \frac{\mathbf{x}_1^T\mathbf{x}_1 + \mathbf{x}_1^T Y\mathbf{L}_{1,3}\mathbf{x}_3}{\mathbf{x}_1^T\mathbf{x}_1}$$

$$= 1 + \frac{\mathbf{x}_1^T Y\mathbf{L}_{1,3}\mathbf{x}_3}{\mathbf{x}_1^T\mathbf{x}_1} \quad \text{since } \frac{\mathbf{x}_1^T\mathbf{x}_1}{\mathbf{x}_1^T\mathbf{x}_1} = 1$$

The vector $\mathbf{L}_{1,3}\mathbf{x}_3$ is positive since $\mathbf{L}_{1,3}$ is nonpositive and $\mathbf{x}_3$ is negative, and if $\mathbf{L}_{1,3}$ were zero, G would not be connected. Recall that the matrix Y and the vector $\mathbf{x}_1$ are positive as well. So, $\mathbf{x}_1^T Y \mathbf{L}_{1,3}\mathbf{x}_3$ is positive. Also, $\mathbf{x}_1^T\mathbf{x}_1$ is also positive. Therefore, by the equation and inequality chain above, $(\epsilon + \lambda_2)\lambda_{n1}(Y) > 1$. All eigenvalues of Y are positive and since $Y = (\epsilon\mathbf{I} + \mathbf{L}_{1,1})$, the eigenvalues of Y are the reciprocals of ϵ

All of these steps can be performed again on the block $\mathbf{L}_{2,2}$, using other corresponding vectors and matrices, to see that $\lambda_1(\mathbf{L}_{2,2}) < \lambda_2$. Therefore, the matrix

$$\begin{bmatrix} \mathbf{L}_{1,1} & \mathbf{O} \\ \mathbf{O} & \mathbf{L}_{2,2} \end{bmatrix}$$

which is a leading submatrix of $L(G)$ has two eigenvalues which are less than λ_2. By Theorem 2.6, the Cauchy Interlacing Theorem, $L(G)$ has 2 eigenvalues whose values are less than λ_2. However, this contradicts λ_2 being the second-smallest eigenvalue of $L(G)$. Due to this contradiction, the original assumption must be false, and the two subgraphs G_1 and G_2 must be connected. [13]

Figure 4.3.1: A graph with two communities that have strong interconnectedness but very few edges between them.

4.4 Algebraic Connectivity

The *second-smallest eigenvalue* of $L(G)$, $\lambda_2(L(G))$, is often called the algebraic connectivity of the graph G. A basic intuition behind the use of this term is that a graph with a higher algebraic connectivity typically has more edges, and can therefore be thought of as being *more connected*. As an example, consider the two graphs K_{100}, the complete graph on 100 vertices, and C_{100}, the cycle graph on 100 vertices. By generating the Laplacian matrix of both of these graphs, we can find that the second-smallest eigenvalue of K_{100} is 100 and the second-smallest eigenvalue of C_{100} is about 0.003947. The algebraic connectivity of K_{100} is many orders of magnitude greater than that of C_{100}, which makes sense, since K_{100} has 4950 edges and C_{100} has only 100 edges on the same number of vertices—clearly, K_{100} can be seen as being much more *connected* than C_{100}. In general, the smaller the algebraic connectivity of a graph, the more likely it is that the Fiedler method will find a *good* partition.[14]

4.5 Some natural questions

What effect does connectedness have on λ_2 (of L)?

The following lemma answers this question.[15] It also justifies why λ_2 is called the algebraic connectivity of the graph.

Lemma 4.5.1

$\lambda_2 > 0 \Leftrightarrow$ the graph is connected.

Proof:

($\Longrightarrow$) If G has two connected components, then $(1, \ldots, 1, 0, \ldots, 0)$ and $(0, \ldots, 0, 1, \ldots, 1)$ are linearly independent eigenvectors for 0. This means $(\lambda_1 =)\lambda_2 = 0$.

($\Longleftarrow$) Suppose G is connected. If $\mathbf{v}$ is an eigenvector corresponding to 0 then

$$0 = \mathbf{v}^T L\mathbf{v} = \sum_{\{i,j\}\in E} (v_i - v_j)^2 \Rightarrow v_i = v_j \text{ for every edge } \{i,j\}.$$

Due to connectedness, $\mathbf{v} \in \langle(1,\ldots,1)\rangle$. So, geometric (= algebraic) multiplicity of $(1,\ldots,1)$, which means that $\lambda_2 \neq 0$.

What if all entries in the Fiedler vector are positive?

Say $L(G)\mathbf{v} = \lambda\mathbf{v}$ with $\lambda > 0$, then

$$\begin{aligned} l_{11}v_1 + \cdots + l_{1n}v_n &= \lambda v_1 \\ &\vdots \\ l_{n1}v_1 + \cdots + l_{nn}v_n &= \lambda v_n \end{aligned}$$

Adding these gives

$$\sum_{i=1}^{n} l_{i1}v_1 + \cdots + \sum_{i=1}^{n} l_{in}v_n = \lambda \sum_{i=1}^{n} v_i.$$

All the sums, $\sum_{i=1}^{n} l_{ik} = 0$ for $k = 1, 2, \ldots, n$ because of the way $L(G)$ is defined. Indeed, the k^{th} sum has $+deg(k)$ contribution from k and -1 contribution from each neighbour of k, which makes the total 0. It follows that $\sum_{i=1}^{n} v_i = 0$ because $\lambda > 0$.

How to balance the 0's (if at all) in the Fiedler vector?

There can be cases where the presence of 0's causes problems in balancing the sizes of G_1, G_2. What do we do if there are only a few nonzero components of the Fiedler vector and a bunch of 0's?

We do not have a solution to this yet, and this is indeed a drawback of the Fiedler vector method because it gives only an approximate solution. By observation, we had conjectured that

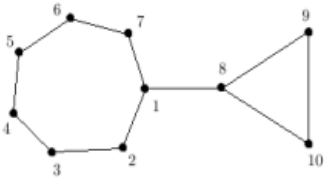
For a connected graph G with Laplacian $L(G)$, let $\mathbf{v}$ be a Fiedler vector.
Look at $S = \{i : v_i = 0\}$. Then for each $i \in S$, $\exists$ j and k such that
$v_j > 0$, $v_k < 0$ and both j and k are neighbours of i.

However, upon exploring further examples, it has been discovered that the conjecture is *false*. A counterexample is $K_{n,n}$, the complete bipartite graph on $2n$ vertices.

4.6 Examples

At this point, with the general theory fully discussed and proven, we will show some examples to illustrate how using the Fiedler vector (the eigenvector corresponding to the second-smallest 7eigenvalue) of the Laplacian matrix of various graphs effectively partitions them in an ideal way in order to minimize the number of edges between the generated subgraphs.

Example 1



(a) The graph A

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{bmatrix} 3 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix} \end{array}$$

(b) The Laplacian matrix $L(A)$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{bmatrix} 0.0521 \\ -0.1147 \\ -0.2543 \\ -0.3335 \\ -0.3335 \\ -0.2543 \\ -0.1147 \\ 0.3734 \\ 0.4897 \\ 0.4897 \end{bmatrix}$$

(c) The Fiedler vector $\mathbf{x}_A$

Figure 4.6.1 : The graph A and its associated Laplacian matrix and Fiedler vector are depicted here. The algebraic connectivity of A is about 0.2375. Note that the sum of all elements of the Fiedler vector is 0. This is true of the Fiedler vector of any graph.

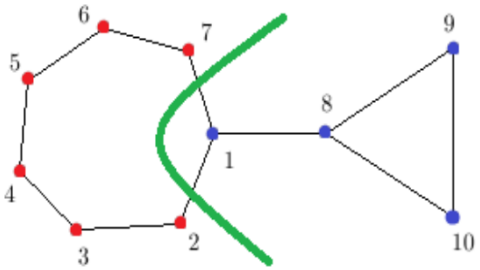
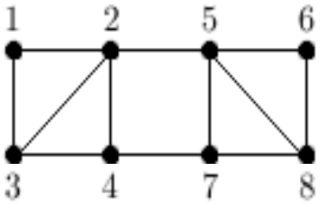


Figure 4.6.2 : This is the spectral partition of the graph A according to Fiedler's method, where each subgraph is shown consisting of vertices of all the same color, divided by a curved green line that forms the cut.

- First, no matter in what way we look at this partition, it is not perfect.
- If we were looking for a partition that simply minimized the number of edges that cross the partition, we could get one by having only edge $(1, 8)$ be in the cut, thereby changing which subgraph the vertex 1 is in.
- If we were looking to form a partition where the number of vertices in each subgraph is as near to equal as possible, then by removing edge $(7, 1)$ from the cut and adding edge $(6, 7)$ to the cut, both subgraphs would have 5 vertices. In fact, if this were done, the number of edges in the cut would not increase at all.

Example 2



(a) The graph B

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 3 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 & 0 & -1 \\ 0 & 0 & 0 & -1 & -1 & 0 & 3 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & 3 \end{bmatrix} \end{array}$$

(b) The Laplacian matrix $L(B)$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{bmatrix} 0.4886 \\ 0.2503 \\ 0.4006 \\ 0.1953 \\ -0.2503 \\ -0.4886 \\ -0.1953 \\ -0.4006 \end{bmatrix}$$

(c) The Fiedler vector $\mathbf{x}_B$

Figure 4.6.3: The graph B and its associated Laplacian matrix and Fiedler vector are depicted here. The algebraic connectivity of A is about 0.6678.

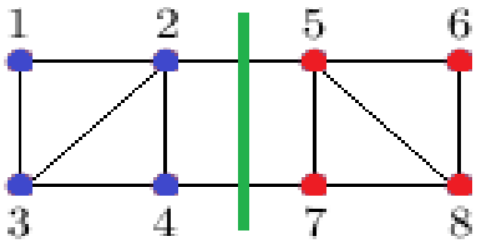


Figure 4.6.4: This is the spectral partition of the graph B according to Fiedler's method, where each subgraph is shown consisting of vertices of all the same color, divided by a green line that forms the cut.

- In this case, the graph B was formed more nicely, allowing Fiedler's spectral graph partitioning method to easily choose the best partition.
- We can observe that each subgraph has the same number of vertices, and the cut is of smallest possible size of any cut that could be made on B (disregarding the sizes of the subgraphs).

Example 3

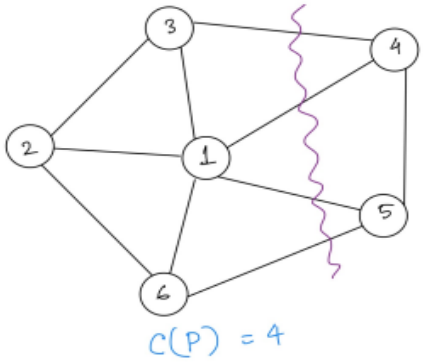


Figure 4.6.5: The graph C and its cut as processed by the algorithm.

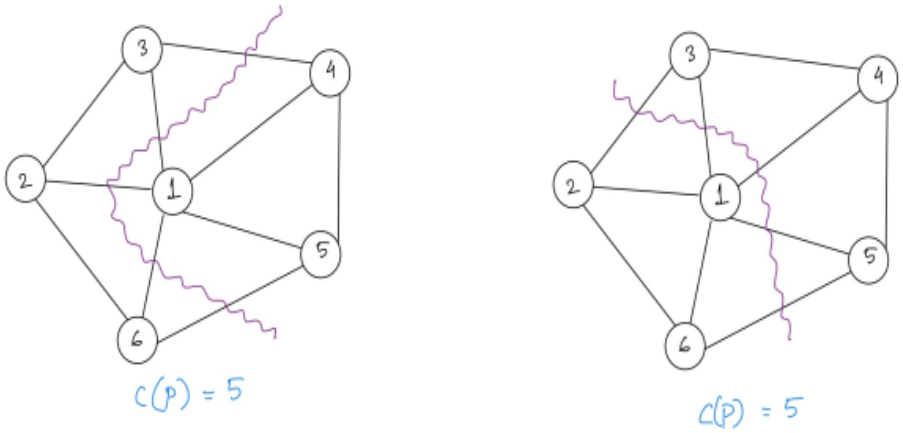


Figure 4.6.6: Ideal cuts

Fun Exercise...

(Try to apply what we have learned so far in Real life)

A class of ten students needs to be split up. The goal is to get two groups of size as close as possible while minimizing the number of friendships that must be broken up. If the friendships are given in the following table use the Fiedler method to split up the class. How many friendships must be broken up? Draw the two resulting friend networks and indicate with dotted lines where the broken friendships are.

	Austin	Beth	Charlie	Dana	Erik	Fiona	Greg	Helen	Ian	Justin
Austin		✓	✓							✓
Beth	✓				✓					✓
Charlie	✓			✓			✓		✓	
Dana			✓		✓				✓	
Erik		✓		✓		✓		✓		✓
Fiona					✓			✓		
Greg			✓					✓		
Helen					✓	✓	✓		✓	
Ian			✓	✓				✓		✓
Justin	✓	✓			✓				✓	

4.7 Limitations and Complexity

Spectral graph partitioning, particularly Fiedler's method, offers a simple and elegant approach to partitioning graphs based on their spectral properties. However, it is not without limitations and computational challenges. Below are the some limitations and complexities associated with Fiedler's spectral graph partitioning algorithm,

1. **Sub-Optimal Partitioning**:
    - Fiedler's method may lead to sub-optimal partitioning, as demonstrated by simple counterexamples with a small number of vertices.
    - The partitions may not always minimize the number of edges in the cut or achieve near-equal sizes for each subgraph.
2. **Optimization Challenge**:
    - Finding the eigenvalues and eigenvectors of the Laplacian matrix is computationally expensive, especially for large graphs.
    - The computation of eigenvalues involves solving an optimization problem, which can be time-consuming and resource-intensive.
3. **Complexity of Eigenvalue Computation**:
    - The complexity of computing eigenvalues and eigenvectors increases with the size of the graph.
    - For large graphs, the computational cost of eigenvalue computation may become prohibitive, making Fiedler's method impractical.
4. **Need for Refinement**:
    - Additional refinement techniques may be required to improve the quality of partitioning obtained from Fiedler's method.

- Post-processing steps, such as K-means clustering or iterative improvement algorithms, may be needed to refine the initial partition.

5. **Dependency on Graph Structure**:

   - The effectiveness of Fiedler's method depends on the structure of the input graph.
   - Graphs with irregular or complex structures may not yield satisfactory partitioning results using Fiedler's method alone.

6. **Scalability Issues**:

   - Fiedler's method may face scalability issues when applied to very large graphs with millions or billions of vertices and edges.
   - The computational resources required to partition such large graphs using Fiedler's method may be prohibitive.

7. **Requirement for Vertex Coordinates**:

   - Some graph partitioning methods, unlike Fiedler's method, require vertex coordinates to be assigned.
   - Fiedler's method is acclaimed for its simplicity and independence from vertex coordinates, but this may limit its applicability in certain contexts.

In summary, while Fiedler's spectral graph partitioning algorithm offers simplicity and elegance, it also has limitations and computational challenges, particularly when applied to large or irregular graphs. Future research may focus on addressing these challenges and developing more efficient and effective graph partitioning algorithms.

5 Experiments - Clustering and Partitioning

In the context of data analysis, there are two primary approaches to clustering or partitioning: taking raw data and transforming it into a graph representation, or starting with graph data and directly applying clustering or partitioning algorithms. Here we have shown two examples related to both clustering and partitioning seperately. Some of the algorithmic references are taken from a github repository. [16]

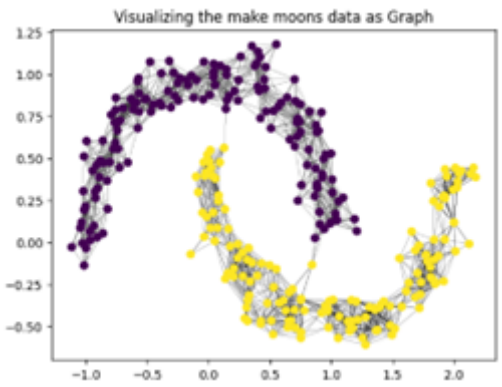
Then, we have explored how to transform a raw tabular data into graph representation where nodes represent data points and edges represent relationships or similarities between them. Once the graph is constructed, clustering or partitioning algorithms like K-means, DBSCAN, or Spectral partitioning has been applied to identify meaningful groupings or partitions within the data.

5.1 Make Moons

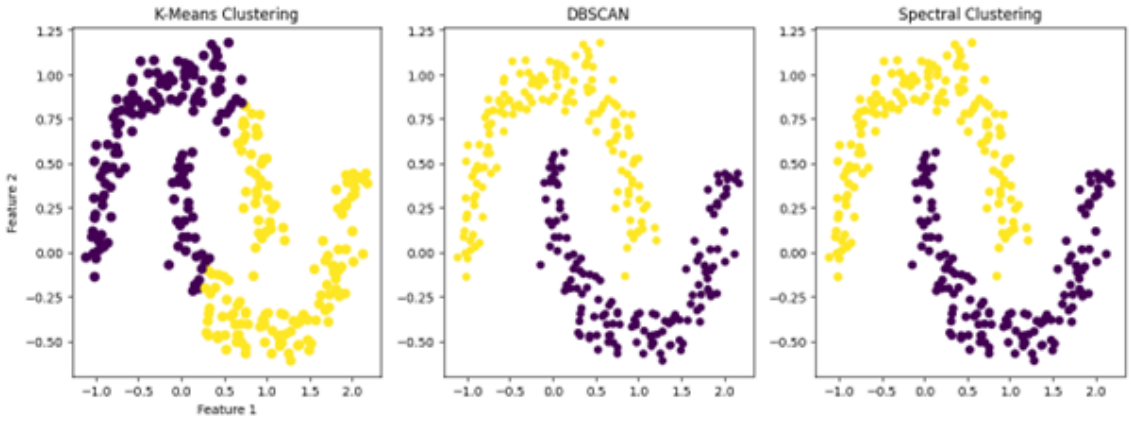
Description

- A simple toy dataset to visualize clustering and classification algorithms.
- Make two interleaving half circles.

Graph



Clustering



K-Means Clustering	DBSCAN	Spectral partitioning
74.67%	99.67%	99.67%

Table 1: Accuracy Summary

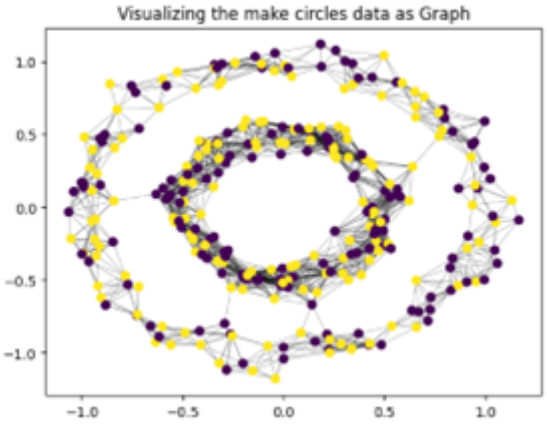
- K-Means clustering with 2 clusters achieved an accuracy of 74.67%, while both DBSCAN and Spectral partitioning achieved a significantly higher accuracy of 99.67%.
- ML techniques like KMeans may not perform as well as linear algebra techniques like Spectral partitioning for graph partitioning tasks.
- Spectral partitioning and DBSCAN tend to outperform KMeans, especially when dealing with complex or non-linear structures in the graph.
- The choice of partitioning algorithm depends on factors such as data characteristics, computational resources, and the desired quality of partitioning.

5.2 Make Circles

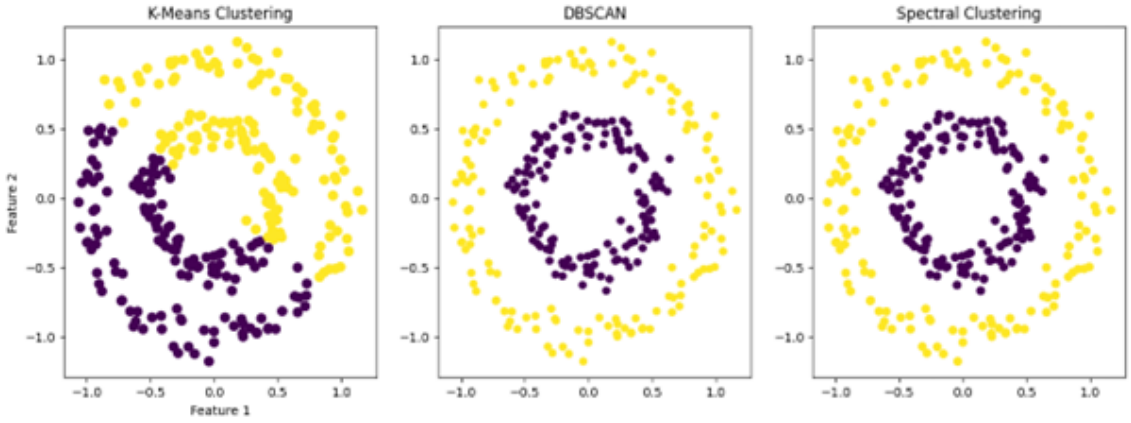
Description

- A simple toy dataset to visualize clustering and classification algorithms.
- Make a large circle containing a smaller circle in 2D.

Graph



Clustering



K-Means Clustering	DBSCAN	Spectral partitioning
51%	100%	100%

Table 2: Accuracy Summary

- K-Means clustering with 2 clusters achieved an accuracy of 51%, while both DBSCAN and Spectral partitioning achieved a significantly higher accuracy of 100%.
- ML techniques like KMeans may not perform as well as linear algebra techniques like Spectral partitioning for graph partitioning tasks.
- Spectral partitioning and DBSCAN tend to outperform KMeans, especially when dealing with complex or non-linear structures in the graph.
- The choice of partitioning algorithm depends on factors such as data characteristics, computational resources, and the desired quality of partitioning.

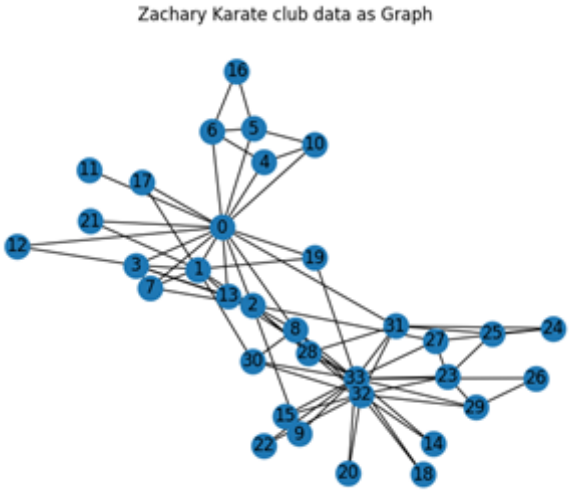
5.3 Zachary Karate Club

Description

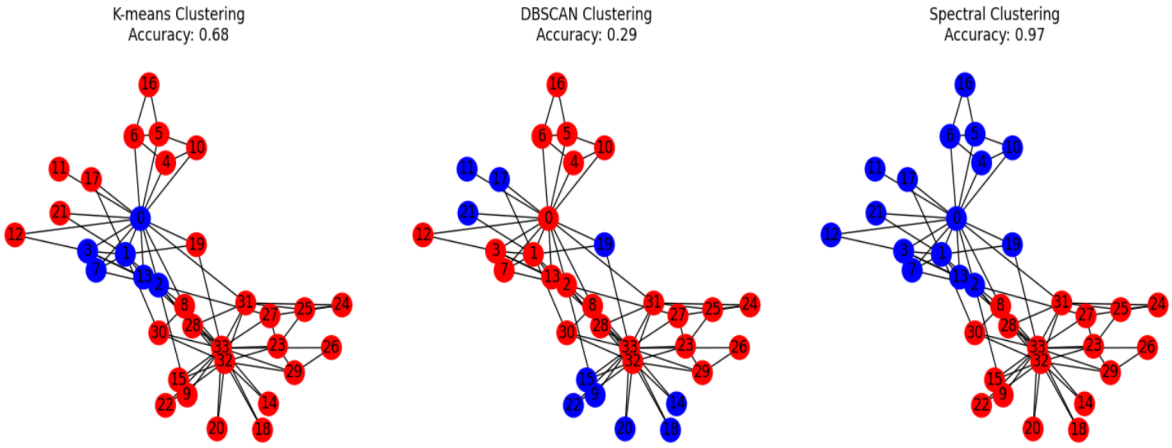
- This is the well-known and much-used Zachary karate club network.

- The data was collected from the members of a university karate club by Wayne Zachary in 1977.
- Each node represents a member of the club, and each edge represents a tie between two members of the club. The network is *undirected*.
- An often discussed problem using this dataset is to find the two groups of people into which the karate club split after an argument between two teachers.

Graph



Partitioning



K-Means Clustering	DBSCAN	Spectral partitioning
68%	29%	97%

Table 3: Accuracy Summary

- Linear algebra techniques like Spectral partitioning outperformed traditional machine learning techniques like KMeans and DBSCAN for graph partitioning in this case.

- Spectral partitioning demonstrated superior performance in accurately partitioning the graph into two clusters.
- The success of Spectral partitioning highlights the importance of leveraging graph-specific properties for effective partitioning, especially when dealing with complex graph structures.

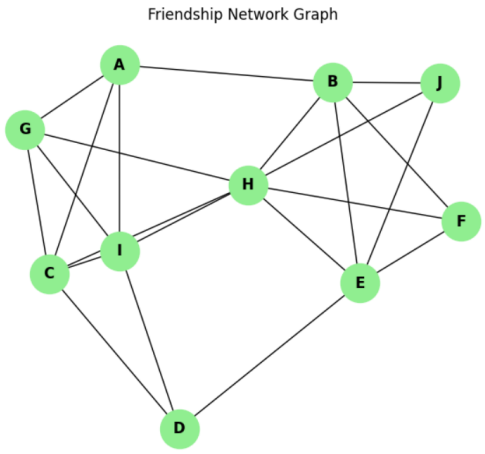
5.4 Friendship Network

Description

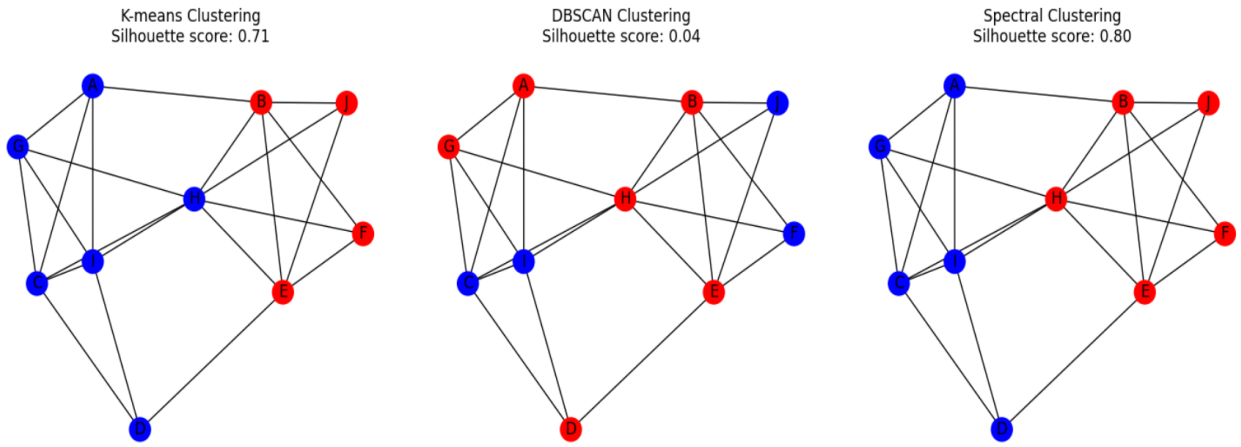
- A class of ten students needs to be split up.
- The goal is to get two groups of size as close as possible while minimizing the number of friendships that must be broken up.
- Each node represents the firendship between two people. The network is *undirected*.

	Austin	Beth	Charlie	Dana	Erik	Fiona	Greg	Helen	Ian	Justin
Austin		✓	✓							✓
Beth	✓				✓					✓
Charlie	✓			✓			✓		✓	
Dana			✓		✓				✓	
Erik		✓		✓		✓		✓		✓
Fiona					✓			✓		
Greg			✓					✓		
Helen					✓	✓	✓		✓	
Ian			✓	✓				✓		✓
Justin	✓	✓			✓				✓	

Graph



Partitioning



K-Means Clustering	DBSCAN	Spectral partitioning
0.71	0.04	0.80

Table 4: Silhoutte Score Summary

Note: The *Silhouette Score* is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). It quantifies the quality of grouping into clusters in a dataset.

- Spectral partitioning achieves the highest silhouette score of 0.80, indicating well-separated and balanced groups while minimizing friendship disruptions.
- Spectral partitioning effectively leverages the connectivity structure of the friendship network, ensuring accurate group partitioning.
- DBSCAN fails miserably
- Unlike K-means, Spectral partitioning's flexibility to handle non-convex shapes suits scenarios with irregular friendship distributions.

5.5 IPL Players Network

Description

The sample IPL data consists of 100 data points, each representing an IPL player with various attributes such as team affiliation, playing role, performance statistics (runs, wickets, matches played), and match outcomes. This is a generated data.

1. **Player**

   Represents the name of the IPL player.

2. **Team**

   Indicates the IPL team to which the player belongs.

3. **Role**

   Specifies the role of the player in cricket, such as '`Batsman`', '`Bowler`', or '`All-Rounder`'.

4. **Runs**

   Represents the total runs scored by the player in IPL matches.

5. **Wickets**

   Indicates the total number of wickets taken by the player in IPL matches.

6. **Matches Played**

   Specifies the total number of matches played by the player in the IPL.

7. **Match Result**

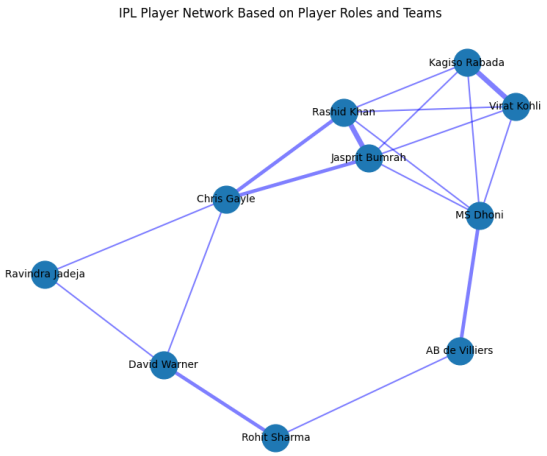
   Denotes the outcome of the matches played by the player, such as '`Win`', '`Loss`', or '`Tie`'.

Here is a preview of the data consisting of the first 10 rows.

Player	Team	Role	Runs	Wickets	Matches Played	Match Result
Rashid Khan	MI	All-Rounder	765	50	18	Win
MS Dhoni	DC	All-Rounder	672	36	8	Tie
Ravindra Jadeja	CSK	Batsman	352	26	9	Loss
Rashid Khan	RR	Bowler	1	26	16	Tie
Jasprit Bumrah	RCB	Batsman	399	36	17	Tie
AB de Villiers	RR	All-Rounder	684	37	11	Win
Jasprit Bumrah	MI	All-Rounder	855	3	5	Tie
David Warner	MI	Batsman	778	11	8	Loss
MS Dhoni	KKR	All-Rounder	274	14	5	Win
Jasprit Bumrah	RR	All-Rounder	349	34	5	Win

Table 5: First 10 rows of IPL Player Data

Graph



- **Node Representation**:

  Each node in the graph represents an IPL player. Nodes are labeled with player names and annotated with attributes such as team affiliation and playing role.

- **Edge Representation**:

Edges between nodes represent relationships between players based on shared attributes. The weight of each edge reflects the similarity between players, determined by factors such as being from the same team or having the same role. The weights is reflected by the thickness of respective edges.

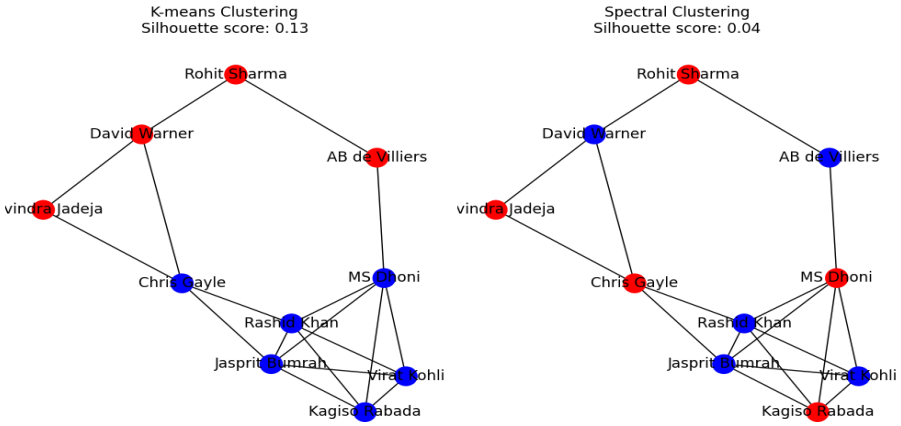
- **Edge Weight Calculation**:

  Edge weights are calculated based on predefined parameters such as the similarity of team affiliation and playing roles. Higher weights indicate stronger relationships between players, influenced by factors like playing together in the same team or sharing similar roles.

- **Interpretation**:

  The resulting graph provides insights into player relationships within the IPL based on team affiliations and playing roles. Clusters of nodes with dense connections represent groups of players who share common attributes, such as playing for the same team or having similar roles.

Partitioning or Clustering



- K-means clustering shows slightly better cluster separation with a silhouette score of 0.13, indicating some level of distinctiveness between clusters, as shown in the above picture.
- Spectral partitioning, on the other hand, exhibits lower cluster separation with a silhouette score of 0.04, suggesting clusters are less well-separated and may have more overlap.
- Overall, the silhouette scores suggest that neither K-means nor Spectral partitioning achieves optimal clustering solutions, highlighting the complexity of the data and the challenge of partitioning it effectively into cohesive clusters, while K-means outperforms Spectral partitioning in this case.

6 Conclusion

This study delved into the comparative effectiveness of Spectral partitioning techniques utilizing Fiedler's vector against conventional machine learning algorithms such as k-means for graph partitioning.

Based on the research investigation conducted on various datasets including synthetic datasets (such as circles and moons) as well as real-world datasets (such as the karate dataset), it has been observed that Spectral partitioning techniques leveraging Fiedler's vector demonstrate remarkable effectiveness in partitioning complex graphs. When compared with machine learning

algorithms like k-means, spectral partitioning consistently outperforms or performs equally well, particularly in scenarios where the graph structures are intricate and non-convex.

The amalgamation of Spectral partitioning and machine learning techniques presents a promising approach for addressing the challenges posed by complex graph structures in data analysis and computational sciences. Spectral partitioning methods, by leveraging the spectral properties of the graph Laplacian matrix and Fiedler's vector, exhibit robustness in capturing underlying structures and partitioning graphs with irregular shapes and varying densities.

Additionally, the introduction of new experiments has expanded the scope of this research. In particular, the analysis of a friendship network revealed that Spectral partitioning outperformed both K-means and DBSCAN algorithms, providing valuable insights into community detection within social networks. Furthermore, the exploration of tabular data transformed into graphs showcased the effectiveness of K-means over Spectral partitioning, highlighting the importance of considering different data characteristics when selecting partitioning algorithms.

In conclusion, this research not only reaffirms the efficacy of Spectral partitioning techniques but also underscores the importance of considering the specific characteristics of the dataset in selecting appropriate partitioning algorithms. The findings presented here contribute to the ongoing discourse in the field of graph partitioning and offer valuable insights for researchers and practitioners alike. Further exploration and empirical evaluations on a broader range of datasets will continue to enhance our understanding and inform decision-making in graph partitioning tasks.

7 Future Scope

Graph partitioning is a fundamental task in data analysis and computational sciences. Here are some future scope questions and aspects to explore further:

1. **Multi-way Partitioning**:

   Explore extending Spectral partitioning for multi-way partitioning to divide graphs into more than two clusters while maintaining balance.[17]

2. **Dynamic Partitioning**:

   Investigate dynamic partitioning techniques that adapt to graph structure changes over time, ensuring efficient updates.

3. **Scalability**:

   Address scalability challenges for Spectral partitioning in large-scale networks by developing distributed algorithms.

4. **Domain Integration**:

   Integrate domain knowledge into partitioning algorithms to guide the process and improve partition quality.

5. **Evaluation Metrics**:

   Develop comprehensive metrics for evaluating partitioning algorithms, considering quality, scalability, and interpretability.

6. **Real-world Applications**:

   Explore applications in diverse domains like social networks and biology to demonstrate practical utility.[18]

7. **Tabular to Graph Conversion**:

   Converting tabular data into graph structures presents a crucial yet challenging task, requiring methods to accurately represent relationships between data points and define node attributes. Efficient conversion methods are essential for leveraging graph-based algorithms and extracting meaningful insights from diverse tabular datasets across various domains.

8 Acknowledgement

We would like to express sincere gratitude to our professor, *Priyavrat Deshpande* for his invaluable guidance and support throughout this research work. His expertise and insights have been instrumental in shaping the direction of this study.

We are also grateful to Dr. James Demmel for his work in explaining Dr. Miroslav Fiedler's theory, which served as a cornerstone for this research. His explanations were invaluable and greatly contributed to the understanding and interpretation of spectral graph partitioning methods.

Furthermore, the contributions of all the group members, our collaboration and efforts were essential to the completion of this project.

Finally, we would like to express our appreciation to the broader machine learning community for their contributions to the development of techniques such as K-means and DBSCAN, which have been instrumental in the analysis and interpretation of the results presented in this work.

9 References

[1] Gilbert Strang. *Lecture 35: Finding Clusters in Graphs.* MIT OpenCourseWare, https://youtu.be/cxTmmasBiC8?si=5LxrfFRPml9aEmyb.

[2] Shaina Race Bennett. *Linear Algebra for Data Science with examples in R.* 2021. North Carolina State University, https://shainarace.github.io/LinearAlgebra/clusintro.html.

[3] Bing Liu. *Web Data Mining, 2nd Edition.* University of Berkeley, https://sirius.cs.put.poznan.pl/~inf89721/Seminarium/Web_Data_Mining__2nd_Edition__Exploring_Hyperlinks__Contents__and_Usage_Data.pdf.

[4] Jiirg Sander Xiaowei Xu Martin Ester, Hans-Peter Kriegel. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* 1996. Institute for Computer Science, University of Munich, https://cmi.ac.in/~madhavan/courses/dmml2024/literature/DBScan-paper.pdf.

[5] Justin Wyss-Gallifent. *Graph Theory.* 2021. Department of Mathematics, University of Maryland, https://www.math.umd.edu/~immortal/MATH401/book/ch_graph_theory.pdf.

[6] Sachin B. Patkar, Viraj Kumar, Bijit Hore, and Hardeep Kaur. *A Graph Partitioning System For Natural Unbalanced Partitions.* Department of Mathematics, Indian Institute of Technology, Bombay, https://ics.uci.edu/~bhore/papers/graph-partitioning.pdf.

[7] Stanford Jure Leskovec and Panayiotis Tsaparas. *Community Detection: Graph Cuts Spectral Clustering.* 2024. University of Ioannina ,https://www.eecs.yorku.ca/~papaggel/courses/eecs4414/docs/lectures/07-spectral.pdf.

[8] Hongyuan Zha Ming Gu; H.D. Simon C.H.Q. Ding, Xiaofeng He. *Spectral Min-Max Cut for Graph Partitioning and Data Clustering*. 2001.

[9] Jean Gallier. *Graph Clustering Using Ratio Cuts*. 2017. Department of Computer and Information Science, University of Pennsylvania, https://www.cis.upenn.edu/~cis5150/cis515-15-spectral-clust-chap5.pdf.

[10] Brian Slininger. *Fiedler's Theory of Spectral Graph Partitioning*. Department of Computer Science, University of California, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=beeee28a9c52287ac9a20868d067d1ed20788036.

[11] Bor-Yiing Su Katya Gonina, Sayak Ray. *Graph Partitioning*. 2024. University of Berkeley, https://patterns.eecs.berkeley.edu/?page_id=571.

[12] James Demmel. Graph partitioning, part 2. 1999. CS267: Notes for Lecture 23, https://people.eecs.berkeley.edu/~demmel/cs267/lecture20/lecture20.

[13] James Demmel. Spectral partitioning. 1996. CS267: Notes for Lecture 23(b), https://people.eecs.berkeley.edu/~demmel/cs267/lecture20/lecture20b.html.

[14] Stefan Steinerberger Adela Depavia. *Spectral clustering revisited: Information hidden in the Fiedler vector*. 2020. https://arxiv.org/pdf/2003.09969.

[15] Sagnik Mukherjee Sagnik Dutta, Nilava Metya. *Fiedler Vector Method*. 2022. Chennai Mathematical Institute, https://nilavam.github.io/FV/report.pdf.

[16] Sacheet Ambedkar. *Community Detection using Spectral Decomposition algorithm*. 2022. Indian Institute of Science, Bangalore, https://github.com/SacheetA/Community-Detection-using-Spectral-Decomposition-algorithm.

[17] Vipin Kumar George Karypis. *Multilevel Graph Partitioning*. 2020. Massachusetts Institute of Technology, https://people.csail.mit.edu/jshun/6886-s18/lectures/lecture13-1.pdf.

[18] Fan Chung. *Four graph partitioning algorithms*. University of California, San Diego, https://mathweb.ucsd.edu/~fan/talks/mlg.pdf.

Appendix

We have uploaded all the experiments conducted in this research to this GitHub repository or you can access it via the following Google Collab notebook.