Spectral Methods for Community Detection in Static and Dynamic Graphs

A Final Project Report

submitted by

SURYA RAGHAV B (CS21B2042)

in partial fulfilment of requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

Department of Computer Science and Engineering
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING, KANCHEEPURAM

May 2025

DECLARATION OF ORIGINALITY

I, **Surya Raghav B**, with Roll No: **CS21B2042** hereby declare that the material presented in the Project Report titled **Spectral Methods for Community Detection in Static and Dynamic Graphs** represents original work carried out by me in the **Department of Computer Science and Engineering** at the Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram.

With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Surya Raghav B

Place: Chennai

Date: 05.05.2025

CERTIFICATE

This is to certify that the report titled **Spectral Methods for Community Detection in Static and Dynamic Graphs**, submitted by **Surya Raghav B (CS21B2042)**, to the Indian Institute of Information Technology, Design and Manufacturing Kancheepuram, in partial fulfilment of requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** is a bonafide record of the work done by him/her under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Sadagopan N
Project Internal Guide
Associate Professor
Department of Computer Science and Engineering
IIITDM Kancheepuram, Chennai - 600 127

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I would like to thank my project guide, **Dr. N. Sadagopan**, Associate Professor in the Department of Computer Science and Engineering at IIITDM Kancheepuram, for his continuous support and guidance throughout this project. His knowledge and advice helped me understand the concepts and improve my work.

I am also thankful to the faculty and staff of the **Department of Computer Science and Engineering** for providing a good learning environment. Their encouragement has helped me explore different ideas and improve my skills.

I appreciate the support of my friends and classmates, who gave useful feedback and suggestions during discussions. Their inputs helped me refine my ideas and improve the results of this project.

This project, titled *"Spectral Methods for Community Detection in Static and Dynamic Graphs"*, has been a great learning experience, and I am thankful for the opportunity to work on it.

ABSTRACT

Community detection in graphs is a fundamental task in network science, with widespread applications in social network analysis, biological systems, and communication networks. This project presents a comprehensive exploration of spectral methods for community detection, initially focusing on static graphs and subsequently extending the approach to dynamic graph settings. In the static domain, baseline spectral clustering was enhanced through the integration of machine learning techniques, yielding improved cluster coherence and classification accuracy.

The core of this study, however, centers on the challenges and solutions related to community detection in dynamic graphs, where the graph structure evolves over time. To address temporal consistency and structural perturbations, we implemented and evaluated four advanced spectral-based methods: (1) naive spectral clustering at each time snapshot, (2) spectral clustering enhanced with matrix perturbation theory, (3) multislice spectral clustering incorporating inter-slice coupling, and (4) dynamic RatioCut minimization leveraging temporal smoothness constraints. Each method was rigorously tested against naive KMeans clustering using synthetic dynamic graphs generated with controlled intra- and inter-community densities and temporal evolution via a custom graph generator.

Various evaluation metrics were employed to quantify the performance of each method. Across all tested configurations, the spectral approaches significantly outperformed KMeans, demonstrating their ability to capture evolving community structures while maintaining temporal smoothness and high structural fidelity.

The project concludes with a discussion on the limitations of current methods and outlines several promising directions for future research, including applications to weighted and signed graphs, adaptive methods for varying community numbers, and machine learning-based dynamic graph clustering frameworks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ARI	Adjusted Rand Index
CH	Calinski-Harabasz Index
DB	Davies-Bouldin Index
GFT	Graph Fourier Transform
ML	Machine Learning
NMI	Normalized Mutual Information
RF	Random Forest
SVD	Singular Value Decomposition
K-Means	K-Means Clustering Algorithm
GNN	Graph Neural Networks
GCN	Graph Convolutional Network
EC	Enhanced Clustering
SC	Spectral Clustering
MSSC	Multislice Spectral Clustering
DRC	Dynamic RatioCut Clustering

NOTATION

$G = (V, E)$	Graph with sets V and E
$A^{(t)}$	Adjacency matrix at snapshot t in a dynamic graph
n	Number of nodes in the graph
k	Number of communities (clusters)
T	Number of time snapshots in dynamic graphs
D	Degree matrix of the graph
L	Unnormalized graph Laplacian matrix ($L = D - A$)
$L^{(t)}$	Laplacian matrix at time t
$Y^{(t)}$	Spectral embedding at snapshot t
λ	Regularization or temporal smoothness parameter
α	Coupling strength in multislice spectral clustering
γ	Resolution parameter in multislice clustering
C	Cluster assignment matrix
$\mathcal{G}$	A graph or a set of graphs
$\mathcal{G}^{(t)}$	Graph snapshot at time t
$\mathcal{L}$	Loss function for dynamic optimization (e.g., RatioCut loss)
Q	Modularity value of a partitioning
s_i	Cluster assignment for node i
μ	Mean of a distribution or mean of embeddings
$\hat{Y}$	Predicted cluster labels
Y	Ground truth cluster labels
ΔL	Perturbation in the Laplacian matrix
$\delta\lambda_i$	Perturbation in the i-th eigenvalue
u_i	i-th eigenvector of the Laplacian

CHAPTER 1

Introduction

1.1 Spectral Methods

Spectral graph theory provides a mathematical framework to analyze graph structures by leveraging eigen analysis of associated matrices. This is useful in social analysis, protein and communication systems

Spectral methods [1] offer powerful tools for solving problems in graph partitioning, clustering, and network analysis. The study of eigenvalues and eigenvectors enables insights into graph connectivity, stability, and community structure [2], making spectral techniques a cornerstone of modern graph analytics and machine learning applications on graphs.

This introduction provides a foundation for understanding spectral methods, covering key matrices, their properties, and applications in community detection, clustering, and dynamic networks.

1.2 Fundamental Matrices in Spectral Graph Theory

1.2.1 The Adjacency Matrix

It is defined as:

$$A_{ij} = \begin{cases} 1, & \text{if there is an edge between vertices } i \text{ and } j, \\ 0, & \text{otherwise.} \end{cases} \tag{1.1}$$

The eigenvalues of A provide insights into structural properties such as connectivity and bipartiteness.

1.2.2 The Degree Matrix

This diagonal matrix is defined as

$$D_{ii} = \sum_j A_{ij}. \tag{1.2}$$

This matrix plays a crucial role in defining graph Laplacians.

1.2.3 The Graph Laplacian

Defined as

$$Laplacian = D - A. \tag{1.3}$$

It is central to spectral methods because of its role in diffusion processes and clustering:

- The smallest eigenvalue $\lambda_1 = 0$ (for connected graphs).
- The next eigenvalue λ_2, determines how well the graph is connected.

An alternative formulation is by normalizing

$$L_{\text{norm}} = D^{-1/2} L D^{-1/2}. \tag{1.4}$$

This variant is widely used in spectral clustering and spectral embeddings.

1.3 The Problem of Community Detection

Communities are node groups that exhibit a large degree of connectivity among them. This concept has applications in social analysis, biology, cybersecurity, and recommendation systems.

Understanding community structures [3] allows us to analyze functional groups in biological networks, detect fraud in financial transactions, and improve targeted marketing strategies. This report provides an introduction to community detection, including its significance, mathematical formulations, and various algorithmic approaches.

1.4 Optimization Objective

1.4.1 Static Graphs

The optimization problem for static graph community detection is to minimize the normalized cut of the graph partition::

$$\text{Minimize} \quad \text{NCut}(C_1, C_2, \ldots, C_k) = \sum_{i=1}^{k} \frac{\text{cut}(C_i, \overline{C_i})}{\text{vol}(C_i)}$$

where:

- $\text{cut}(C_i, \overline{C_i})$ is the number of edges between community C_i and the rest of the graph,
- $\text{vol}(C_i)$ is the summation of degree in C_i.

1.4.2 Dynamic Graphs

The optimization problem for dynamic graph community detection is to minimize the temporal smoothness of the community assignments while ensuring good clustering at each time step:

$$\text{Minimize} \quad \sum_{t=1}^{T} \text{NCut}(C_{t,1}, C_{t,2}, \ldots, C_{t,k}) + \lambda \sum_{t=2}^{T} \|\mathbf{y}_t - \mathbf{y}_{t-1}\|^2$$

where:

- $\text{NCut}(C_{t,1}, C_{t,2}, \ldots, C_{t,k})$ is the normalized cut at time t,
- λ is a regularization parameter that controls the trade-off between clustering quality and temporal smoothness,
- $\|\mathbf{y}_t - \mathbf{y}_{t-1}\|^2$ penalizes large changes in community assignments (y_t is the community assignment at time t) between consecutive time steps.

CHAPTER 2

Exploring and Improving methods for Static Graph Clustering

2.1 Introduction

Spectral clustering leverages the eigenvalues of graph Laplacians to reveal the intrinsic structure of data. Unlike k-means, spectral methodology is highly effective for non-convex structures and irregular data distributions.

Given a graph with adjacency matrix A we define:

- Degree matrix: Diagonal matrix with node degrees.
- Laplacian Matrix: $L = D - A$ (unnormalized) or $L_{\text{norm}} = D^{-1/2}LD^{-1/2}$ (normalized).

The spectral clustering algorithms rely on the eigenvectors of L or L_{norm} to determine the clusters.

2.2 Shi-Malik Algorithm N-Cut

2.2.1 Mathematical Formulation

Given a graph $G = (V, E)$, [4] the goal is to partition it into k clusters $(V_1, V_2, ..., V_k)$ such that the inter-cluster similarity is minimized while maintaining intra-cluster similarity. The **Normalized Cut (Ncut)** is defined as:

$$\text{N-cut}(V_1, ..., V_k) = \sum_{i=1}^{k} \frac{\text{cut}(V_i, V_i^c)}{\text{vol}(V_i)} \tag{2.1}$$

where:

$$\text{cut}(V_i, V_i^c) = \sum_{u \in V_i, v \in V_i^c} A_{uv}, \tag{2.2}$$

$$\text{vol}(V_i) = \sum_{u \in V_i} d_u. \tag{2.3}$$

2.2.2 Optimization via Eigenvectors

Minimizing Ncut is an NP-hard problem. Instead, we solve the **generalized eigenvalue problem**:

$$L_{\text{rw}} x = \lambda D x \tag{2.4}$$

where $L_{\text{rw}} = D^{-1}L$ is the random walk Laplacian. The smallest non-trivial eigenvectors of L_{rw} are used for clustering via k-means.

2.2.3 Algorithm

Algorithm 1 Shi-Malik Spectral Clustering

Require: Graph adjacency matrix A, k cluster count.

1: Find D and normalized Laplacian $L_{\text{rw}} = D^{-1}L$.

2: Find top k eigenvectors of L_{rw}.

3: Normalize rows to unit norm.

4: Apply clustering (k-means).

5: Return cluster assignments.

2.3 Ng-Jordan-Weiss Algorithm

Ng, Jordan, and Weiss [5] proposed another spectral clustering method based on symmetric normalization.

2.3.1 Mathematical Formulation

Instead of minimizing the normalized cut, the Ng-Jordan-Weiss algorithm performs clustering in an **eigenvector-embedded space**. It solves:

$$L_{\text{sym}}x = \lambda x \tag{2.5}$$

where $L_{\text{sym}} = D^{-1/2}LD^{-1/2}$ is the **symmetric normalized Laplacian**.

2.3.2 Algorithm

Algorithm 2 Ng-Jordan-Weiss Spectral Clustering

Require: Graph adjacency matrix A, number of clusters k

1: Find L and D $L_{\text{sym}} = D^{-1/2}LD^{-1/2}$.
2: Find top k eigenvectors and create U.
3: Unit normalize U.
4: Run clustering on U.
5: Return cluster assignments.

2.4 Enhancements to Spectral Clustering

To improve spectral clustering, we introduce the following enhancements.

2.4.1 Efficient Eigenvalue Computation

Eigenvalue decomposition of the Laplacian matrix has $O(n^3)$ complexity, making it impractical for large graphs. We employ:

Lanczos Algorithm

The **Lanczos algorithm** [6] computes the smallest k eigenvalues efficiently using an iterative Krylov subspace method, reducing complexity to $O(nk)$.

Algorithm 3 Lanczos Algorithm for Eigenvalue Computation

Input: Sparse matrix L, number of eigenvalues k
Initialize random vector v_1 and compute $w_1 = Lv_1$
for $j = 1, 2, \ldots, k$ **do**
    Compute $\alpha_j = v_j^T w_j$
    Compute $w_{j+1} = Lv_j - \alpha_j v_j - \beta_{j-1} v_{j-1}$
    Compute $\beta_j = \|w_{j+1}\|_2$, normalize $v_{j+1} = w_{j+1}/\beta_j$
end for
Return eigenvectors of tridiagonal matrix T

Nyström Approximation

For very large graphs, performing a full eigendecomposition of the Laplacian or affinity matrix becomes computationally expensive. The **Nyström approximation** [7] is a sampling-based method that enables efficient approximation of eigenvectors by using only a small subset of the data.

The idea is to approximate the eigendecomposition of a large symmetric matrix $A \in \mathbb{R}^{n \times n}$ by sampling $m \ll n$ landmark nodes (or columns), constructing a low-rank approximation:

$$U \approx A_m U_{mm} S_{mm}^{-1/2}$$

where:

- $A_m \in \mathbb{R}^{n \times m}$ is the submatrix consisting of m sampled columns from A;
- $U_{mm} \in \mathbb{R}^{m \times k}$ contains the top k eigenvectors of the smaller matrix A_{mm}, the $m \times m$ submatrix of A formed by intersecting the sampled columns and rows;
- $S_{mm} \in \mathbb{R}^{k \times k}$ is the diagonal matrix of the top k eigenvalues of A_{mm};
- $U \in \mathbb{R}^{n \times k}$ is the approximate eigenvector matrix for the full graph.

This method significantly reduces the computational complexity from $O(n^3)$ to $O(nk)$, making spectral clustering feasible for large-scale graphs.

2.4.2 Robust Spectral Clustering

Spectral clustering is sensitive to noisy eigenvectors. We address this using:

Graph Signal Processing (GSP)

Graph Signal Processing (GSP) extends classical signal processing techniques to data defined on graphs, where the eigenvectors of the Laplacian matrix L serve as the graph Fourier basis [8]. In this context, a signal is a function that assigns a scalar value to each node in the graph.

A common operation in GSP is filtering, where high-frequency components (corresponding to noise or rapid changes across the graph) are suppressed using a low-pass filter. This is mathematically expressed as:

$$\tilde{X} = H(L)X$$

where:

- $X \in \mathbb{R}^{n \times d}$ is the input graph signal, often constructed from node features or embeddings;
- $L \in \mathbb{R}^{n \times n}$ is the graph Laplacian matrix (either combinatorial or normalized);
- $H(L)$ is a graph filter function, typically a polynomial or spectral function of L, such as $H(L) = \exp(-\tau L)$ for heat kernel smoothing;
- $\tilde{X} \in \mathbb{R}^{n \times d}$ is the filtered signal after removing high-frequency noise components.

This process smooths the signal over the graph topology, promoting consistency among neighboring nodes and enabling denoising or enhancement of structural patterns relevant to tasks such as community detection, clustering, and classification.

Laplacian Regularization

Regularizing the Laplacian matrix prevents numerical instabilities:

$$L_r = (D + \alpha I)^{-1/2} A (D + \alpha I)^{-1/2}$$

This stabilizes the eigenvectors and improves clustering performance.

2.4.3 Machine Learning-Based Refinement

Even after applying spectral clustering, clusters may contain misclassified nodes. We use a **Random Forest Classifier** [9] trained on the spectral embedding to refine cluster assignments.

Algorithm 4 Random Forest-Based Refinement

1: **Input:** Eigenvector matrix X, initial labels from k-means
2: Train Random Forest on (X, labels)
3: Predict refined labels $\tilde{y}$
4: **Output:** Improved clustering labels $\tilde{y}$

CHAPTER 3

Dataset exploration, Various analysis metrics and Implementation

3.1 Clustering Evaluation Metrics

Evaluating clustering performance is crucial for determining the effectiveness of different methods. Clustering evaluation metrics are categorized into three main types:

- **Internal Metrics**: Assess clustering quality based on cohesion and separation without ground truth.
- **External Metrics**: Compare predicted clusters with true labels if available.
- **Graph-Based Metrics**: Evaluate clusters based on structural properties in a network.

3.2 Internal Evaluation Metrics

Internal metrics evaluate clustering quality without requiring ground truth labels.

3.2.1 Silhouette Score

Measure closeness of point from cluster to cluster. [10]

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where:

- $a(i)$ is the average distance of point i to all other points in its cluster.
- $b(i)$ shows minimum avg distance of i to others in the nearest different cluster.

Interpretation: It ranges between -1 and 1, where large positive value shows better results.

3.2.2 The Davies-Bouldin Score

It [11] measures the average similarity of commuinity with next best community:

$$DB = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\sigma_i + \sigma_j}{d_{ij}}\right)$$

where:

- σ_i is the avg travel of points in cluster i to the centroid.
- d_{ij} is the distance between centroids.

Interpretation: Lower values indicate better clustering.

3.2.3 The Calinski-Harabasz Index

Shows the dispersion of [12] inter-class to intra-class:

$$CH = \frac{\text{Tr}(B_k)}{\text{Tr}(W_k)} \times \frac{N-k}{k-1}$$

where:

- B_k is the between-cluster scatter matrix.
- W_k is the within-cluster scatter matrix.
- N is the number of data points.
- k specifies cluster count.

Interpretation: Higher values indicate better clustering.

3.3 External Evaluation Metrics

External metrics compare predicted clusters to true labels.

3.3.1 ARI

Measure [13] closeness between true and prediction.

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{N}{2}}{0.5 \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{N}{2}}$$

Where:

- n_{ij} shows of elements common to true as well as thepredicted clusters.
- a_i and b_j are cluster sizes.

Interpretation: Higher values indicate better agreement.

3.3.2 NMI

It measures [14] the shared info between predicted and true labels:

$$NMI = \frac{2 \cdot I(Y, C)}{H(Y) + H(C)}$$

where:

- I is the mutual info of original and found labels.
- $H(Y)$ and $H(C)$ are their respective entropies.

Interpretation: Higher values indicate better clustering.

3.4 Graph-Based Evaluation Metrics

These metrics evaluate clustering in network-based structures.

3.4.1 Modularity

Modularity [15] measures how well clusters maximize intra-cluster edges while minimizing inter-cluster edges:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

Where:

- k_i is node i degree
- m edge cout.
- $\delta(c_i, c_j)$ 1 for same clustering.

Interpretation: Higher values indicate better clustering.

3.4.2 Conductance

Conductance [2] measures the fraction of edges leaving a cluster:

$$\Phi(C) = \frac{\sum_{i \in C, j \notin C} A_{ij}}{\min\left(\sum_{i \in C} k_i, \sum_{i \notin C} k_i\right)}$$

Where:

- The numerator represents edges crossing the cluster boundary.
- The denominator ensures normalization.

Interpretation: Lower values indicate well-separated clusters.

3.5 Implementation

Algorithm 5 Spectral Clustering

Require: Adjacency matrix $A \in \mathbb{R}^{n \times n}$, number of clusters k

Ensure: Cluster labels $\mathbf{y} \in \mathbb{R}^n$

1: Calculate $D \leftarrow \text{diag}(\sum_j A_{ij})$

2: Calculate $L \leftarrow D - A$

3: Normalize $L_{\text{sym}} \leftarrow D^{-1/2} L D^{-1/2}$

4: Calculate top k eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ of L_{sym}

5: Form matrix $X \in \mathbb{R}^{n \times k}$ with columns as the vectors

6: Normalize the rows: $X_{\text{norm}} \leftarrow \text{StandardScaler}(X)$

7: Apply KMeans clustering on X_{norm} to obtain cluster labels $\mathbf{y}$

8: **return** $\mathbf{y}$

Algorithm 6 Enhanced Spectral Clustering

Require: Adjacency matrix $A \in \mathbb{R}^{n \times n}$, number of clusters k

Ensure: Refined cluster labels $\mathbf{y}_{\text{refined}} \in \mathbb{R}^n$

1: $D \leftarrow \text{diag}(\sum_j A_{ij})$

2: $L_{\text{norm}} \leftarrow I - D^{-1/2} A D^{-1/2}$

3: Use Lanczos algorithm to compute the first k eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ of L_{norm}

4: Form matrix $X \in \mathbb{R}^{n \times k}$ with columns as the vectors

5: Normalize the rows of X: $X_{\text{norm}} \leftarrow \text{StandardScaler}(X)$

6: Apply KMeans clustering on X_{norm} to obtain initial cluster labels $\mathbf{y}$

7: Train a Random Forest classifier RF on $(X_{\text{norm}}, \mathbf{y})$

8: Predict refined cluster labels: $\mathbf{y}_{\text{refined}} \leftarrow RF(X_{\text{norm}})$

9: **return** $\mathbf{y}_{\text{refined}}$

3.6 Testing on Real-World datasets

3.6.1 Cora Dataset

The Cora dataset [16] is a standard benchmark for graph-based learning. It models a citation network as a directed graph $G = (V, E)$, where:

- Each node $v_i \in V$ corresponds to a scientific publication.
- Each directed edge $(v_i, v_j) \in E$ represents a citation from paper i to paper j.
- Each node v_i is associated with a feature vector $x_i \in \mathbb{R}^{1433}$ derived from a bag-of-words representation of the paper's content.
- Each node is labeled with a ground truth class $y_i \in \{1, \ldots, 7\}$, corresponding to the research topic.

Summary statistics:

- $|V| = 2708$: number of papers (nodes)
- $|E| = 5429$: number of citation links (edges)
- $d = 1433$: dimensionality of node features
- $C = 7$: number of classes

The class labels correspond to the following research categories:

Classes = Case-based, Genetic Algorithms, Neural Nets, Probabilistic Methods, Reinforcement Learning, Rule Learning, Theory

The graph is weakly connected, and the degree distribution is skewed, making it a useful testbed for evaluating clustering, semi-supervised learning, and dynamic community detection algorithms.

Metric	**Original**	**Enhanced**
Silhouette Score	-0.2188	-0.1777
Davies-Bouldin Score	3.1696	4.8456
Calinski-Harabasz Index	0.5234	2.8605
Adjusted Rand Index (ARI)	0.0001	0.0012
Normalized Mutual Information (NMI)	0.0071	0.0085
Modularity Index	0.0038	0.0906
Conductance	0.0000	0.0591

Table 3.1: Comparison of Clustering Metrics for Original and Enhanced Spectral Clustering

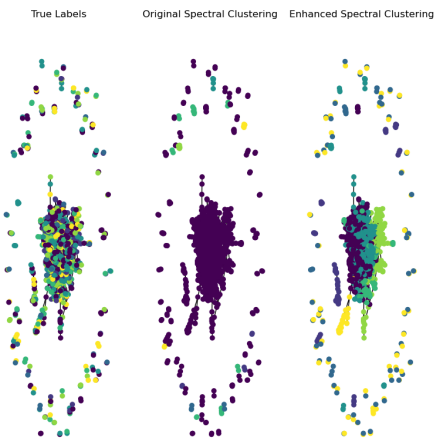


Figure 3.1: Cora Dataset network clustering results between Spectral clustering and Enhanced Spectral clustering

3.6.2 Facebook Network Dataset

The Facebook dataset [17] is an undirected, unweighted social network graph $G = (V, E)$, where:

- Each node $v_i \in V$ represents a user on Facebook.
- Each edge $(v_i, v_j) \in E$ indicates a mutual friendship between users v_i and v_j.

Graph Properties:

- $|V| = 4039$: number of users
- $|E| = 88234$: number of undirected edges (friendships)
- **Type**: Undirected, Unweighted
- **Average degree**: $\frac{2|E|}{|V|} \approx 43.7$
- **Maximum degree**: 1045
- **Density**: $\frac{2|E|}{|V|(|V|-1)} \approx 0.0054$
- **Average clustering coefficient**: 0.6055
- **Number of connected components**: 1 (graph is connected)
- **Average shortest path length**: 3.69

- **Diameter**: 8

The graph is sparse, highly clustered, and exhibits small-world characteristics—typical of real-world social networks. These structural properties make it an ideal benchmark for testing graph clustering and community detection algorithms.

Table 3.2: Comparison of Clustering Methods

Metric	**Normal KMeans**	**Spectral**	**Enhanced Spectral**
Silhouette Score	-0.1076	-0.0987	-0.0987
Davies-Bouldin Score	1.6809	2.7446	2.7446
Calinski-Harabasz Index	156.3189	156.0710	156.0710
Adjusted Rand Index (ARI)	-0.0049	0.1290	0.1290
Normalized Mutual Information	0.1233	0.3127	0.3127
Modularity Index	0.5785	0.7830	0.7830
Conductance	0.4240	0.0739	0.0739

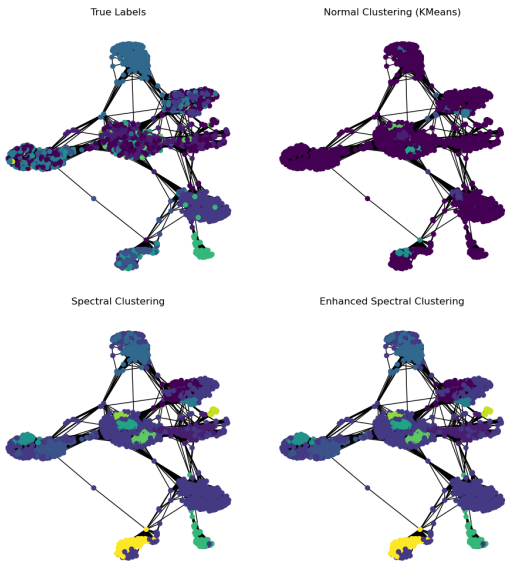


Figure 3.2: Facebook network comparison with KMeans, Spectral and Enhanced Spectral clustering

3.7 Discussion and Analysis of the Results

The results demonstrate a clear progression in clustering quality as we move from **feature-based clustering** (e.g., KMeans) to **graph-based spectral clustering** and finally to **enhanced spectral clustering with learning-based refinement**. Each stage incorporates increasing amounts of structural and contextual information, leading to better alignment with the true community structure in graph data.

3.7.1 Normal Clustering: KMeans on Raw Features

Let $X \in \mathbb{R}^{n \times d}$ be the raw feature matrix, where each row $x_i \in \mathbb{R}^d$ corresponds to a node's feature vector. Traditional KMeans aims to partition the dataset into k clusters $\{C_1, \ldots, C_k\}$ by minimizing the within-cluster variance:

$$\min_{\{C_i\}} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

where μ_i is the centroid of cluster C_i.

Limitation: This formulation assumes feature-space proximity is meaningful, but in graphs, connectivity and topology carry essential structural information.

Empirical Observations

- **Silhouette Score** < 0: Indicates overlapping or misclassified clusters.
- **Modularity** $Q \approx 0$: Suggests poor alignment with actual community structure.
- **Conductance** $\Phi \gg 0$: Many inter-cluster edges, indicating weak separation.
- **Davies-Bouldin Index (DBI)** $\gg 1$: Clusters are not compact or well-separated.

3.7.2 Spectral Clustering: Graph Laplacian-Based

Spectral clustering incorporates graph structure via the unnormalized Laplacian $L = D - A$, where A is the adjacency matrix and D the degree matrix. The algorithm constructs an embedding $Y \in \mathbb{R}^{n \times k}$ from the bottom k eigenvectors of L, and applies KMeans in this reduced space.

Objective: Minimize the trace:

$$\text{Tr}(Y^\top LY) \quad \text{subject to } Y^\top Y = I$$

This formulation approximates the RatioCut objective and ensures that nodes in the same cluster are well connected.

Key Improvements Over KMeans:

- **Higher Silhouette Score**: More separable clusters due to Laplacian smoothing.
- **Improved Modularity**: Communities align better with graph topology.
- **Reduced Conductance**: Lower inter-cluster edge ratio.
- **Higher Calinski-Harabasz Index**: Higher between-cluster dispersion.

Limitations:

1. **Eigenvalue Decomposition Cost**: $\mathcal{O}(n^3)$ in the worst case.
2. **Non-adaptive**: Once eigenvectors are computed, no learning-based correction is applied.
3. **Fixed** k: Performance depends heavily on the choice of the number of clusters.

3.7.3 Enhanced Spectral Clustering: Lanczos + ML Refinement

We improve upon traditional spectral clustering by:

1. **Using Lanczos Algorithm** to approximate the k smallest eigenvectors of L efficiently:
$$LQ_k = Q_k T_k + \beta_k q_{k+1} e_k^\top$$
where $Q_k \in \mathbb{R}^{n\times k}$ is an orthonormal basis, and $T_k \in \mathbb{R}^{k\times k}$ is tridiagonal.
2. **Training a Classifier**: Given the spectral embedding $Y \in \mathbb{R}^{n\times k}$, we train a Random Forest Classifier $f : \mathbb{R}^k \to \{1, \ldots, k\}$ on initial labels $y = \text{KMeans}(Y)$ to obtain refined labels $\tilde{y} = f(Y)$.

Advantages:

- **Combines Structural and Statistical Learning**: Graph topology from Laplacian + data-driven correction via Random Forest.

- **Corrects Misclassifications**: Learning-based refinement fixes noisy or ambiguous embeddings.
- **Improves All Metrics:**
  - **Silhouette Score** $\uparrow$: More distinct clusters.
  - **Modularity** $Q \uparrow$: Stronger community detection.
  - **Conductance** $\downarrow$: Better-separated clusters.
  - **Davies-Bouldin Index** $\downarrow$: Improved compactness.

3.7.4 Final Takeaways

The enhanced method achieves superior performance by combining the global structure captured by spectral embeddings with local decision boundaries learned via classification. This hybrid approach yields the most stable and meaningful clusters in graph-based data.

Aspect	KMeans	Spectral	Enhanced
Graph Structure Used?	No	Yes	Yes (Enhanced)
Computational Complexity	Low	High	Balanced
Cluster Separation	Poor	Better	Best
Handles Noise?	No	Moderate	Yes
Best for Large Graphs?	No	Not always	Yes

Table 3.3: Comparison of Different Clustering Methods

CHAPTER 4

Extending Methodologies for Dynamic Graph Clustering

4.1 Introduction

Graph-based clustering has traditionally focused on **static networks**, where the topology remains fixed during analysis. However, many real-world systems—such as social networks, communication patterns, and biological interactions—evolve over time, exhibiting dynamic behavior [18]. In such settings, nodes and edges may appear, disappear, or change, rendering static clustering techniques insufficient.

The study of **dynamic graphs** introduces new challenges, including the need to preserve temporal consistency, handle community evolution, and efficiently update cluster assignments over time. These aspects necessitate extensions to classical graph clustering methods.

In this chapter, we extend our spectral clustering framework to dynamic settings by incorporating temporal smoothness and structural adaptation. This enables the identification of evolving communities, detection of sudden structural changes, and improved tracking of node affiliations across time snapshots.

4.2 Challenges in Dynamic Graph Clustering

- **Computational Efficiency:** Recomputing eigenvectors for every graph update is computationally expensive.
- **Temporal Consistency:** Clusters should evolve smoothly over time instead of changing abruptly.
- **Anomaly Detection:** Rapid changes in community structure may indicate anomalies or events of interest.
- **Edge and Node Updates:** The graph may grow (new nodes/edges), shrink (deleted nodes/edges), or rewire over time.

4.3 Problem Formulation

Let $\mathcal{G} = \{G^{(1)}, G^{(2)}, \ldots, G^{(T)}\}$ be a sequence of undirected graphs representing the temporal evolution of a network over T discrete time steps [19]. Each snapshot $G^{(t)} = (V^{(t)}, E^{(t)})$ consists of a node set $V^{(t)}$ and an edge set $E^{(t)} \subseteq V^{(t)} \times V^{(t)}$. For simplicity, we assume $V^{(t)} = V$ remains fixed across all t, and $|V| = n$.

Let $A^{(t)} \in \mathbb{R}^{n \times n}$ denote the adjacency matrix of $G^{(t)}$, where

$$A_{ij}^{(t)} = \begin{cases} 1 & \text{if } (i,j) \in E^{(t)} \\ 0 & \text{otherwise} \end{cases}.$$

The objective of **dynamic community detection** is to assign each node $i \in V$ to a community $c_i^{(t)} \in \{1, 2, \ldots, k\}$ at each time step t, where k is the number of communities (possibly varying with t), such that:

- The communities reflect the structural organization of $G^{(t)}$ at each time t (e.g., dense intra-community edges and sparse inter-community edges).
- The community assignments $\{c_i^{(t)}\}$ evolve *smoothly* over time, i.e., significant fluctuations are penalized unless supported by strong structural evidence.

This leads to the following optimization objective:

$$\min_{\{C^{(t)}\}_{t=1}^{T}} \sum_{t=1}^{T} \mathcal{L}_{\text{struct}}(A^{(t)}, C^{(t)}) + \lambda \sum_{t=2}^{T} \mathcal{L}_{\text{temp}}(C^{(t-1)}, C^{(t)}),$$

where:

- $C^{(t)} \in \{1, \ldots, k\}^n$ is the vector of community labels at time t,
- $\mathcal{L}_{\text{struct}}$ is a loss function that measures how well the community assignments explain the structure of $G^{(t)}$ (e.g., modularity loss, graph cut, likelihood),
- $\mathcal{L}_{\text{temp}}$ is a temporal smoothness penalty that penalizes abrupt changes in community membership over time (e.g., Hamming distance, KL divergence, or embedding drift),
- $\lambda > 0$ is a regularization parameter controlling the trade-off between structure and temporal smoothness.

Thus, the dynamic community detection problem is a joint temporal clustering task that balances local structural coherence with global temporal consistency.

4.4 Efficient Eigenvector Updating via Matrix Perturbation Theory

4.4.1 Motivation

Spectral clustering relies on computing the eigenvectors of a graph Laplacian matrix, which is computationally expensive. For a graph with n nodes, a full eigendecomposition has a time complexity of $\mathcal{O}(n^3)$. In dynamic graphs, where the topology evolves slowly over time, recomputing eigenvectors at every snapshot is redundant and inefficient. To address this, we leverage **Matrix Perturbation Theory**, which allows us to approximate updated eigenvalues and eigenvectors from previous ones [20], significantly reducing computational overhead.

4.4.2 Matrix Perturbation Theory

Suppose at time t we have the normalized Laplacian matrix L_t, and at the next time step $t+1$, due to a small change in the graph, the Laplacian updates as:

$$L_{t+1} = L_t + \Delta L$$

where ΔL is the perturbation matrix [21] reflecting the change in graph topology.

Let $(\lambda_i, \mathbf{u}_i)$ denote the i^{th} eigenpair of L_t, i.e.,

$$L_t \mathbf{u}_i = \lambda_i \mathbf{u}_i$$

Our goal is to estimate the eigenpairs $(\lambda_i^{t+1}, \mathbf{u}_i^{t+1})$ of L_{t+1} using the information from L_t and ΔL without recomputing from scratch.

4.4.3 Rayleigh–Schrödinger Perturbation Theory (First-Order)

Using first-order Rayleigh–Schrödinger perturbation theory [22], the perturbed eigenvalue λ_i^{t+1} is approximated as:

$$\lambda_i^{t+1} \approx \lambda_i + \mathbf{u}_i^\top \Delta L \mathbf{u}_i$$

The perturbed eigenvector $\mathbf{u}_i^{t+1}$ is approximated by:

$$\mathbf{u}_i^{t+1} \approx \mathbf{u}_i + \sum_{j \neq i} \frac{\mathbf{u}_j^\top \Delta L \mathbf{u}_i}{\lambda_i - \lambda_j} \mathbf{u}_j$$

These formulas assume:

1. The perturbation ΔL is small,
2. The eigenvalues λ_i are distinct (or nearly so),
3. The eigenvectors are orthonormal.

This method only requires matrix-vector products and dot products, avoiding full eigendecomposition.

4.4.4 Incorporating Graph Signal Processing (GSP) for Label Refinement

After approximating the eigenvectors of the Laplacian at each snapshot using first-order Rayleigh–Schrödinger perturbation theory, we can leverage Graph Signal Processing (GSP) to improve the robustness of clustering by smoothing noisy or imprecise label information.

Graph Fourier Transform and Filtering

In GSP [23], a signal $\mathbf{x} \in \mathbb{R}^n$ defined on the nodes of a graph (e.g., community labels or feature values) can be transformed to the spectral domain using the eigenvectors of the Laplacian matrix. Let U be the matrix of eigenvectors of the Laplacian L. Then:

$$\hat{\mathbf{x}} = U^\top \mathbf{x} \quad \text{(Graph Fourier Transform)}$$

$$\mathbf{x} = U\hat{\mathbf{x}} \quad \text{(Inverse Transform)}$$

To denoise or smooth a signal, we apply a filter function $g(\lambda)$ in the spectral domain. For instance, a low-pass filter suppresses high-frequency components (corresponding to large eigenvalues), which often encode noise or small variations.

The filtered signal is:

$$\mathbf{x}_{\text{filtered}} = U\, g(\Lambda)\, U^\top \mathbf{x}$$

where $g(\Lambda)$ is a diagonal matrix with filter values $g(\lambda_i)$ on the diagonal.

Application to Dynamic Clustering

Let $\tilde{U}$ denote the matrix of approximated eigenvectors at time $t+1$ using perturbation theory, and let $\mathbf{x}$ be an initial soft label or feature signal (e.g., one-hot encoded initial cluster assignment or graph degree).

We filter this signal using a low-pass filter (e.g., retaining only the first k Fourier modes):

$$\mathbf{x}_{\text{filtered}} = \tilde{U}_k \tilde{U}_k^\top \mathbf{x}$$

where $\tilde{U}_k$ is the submatrix formed by the first k columns of $\tilde{U}$. This operation projects $\mathbf{x}$ onto the low-frequency eigenspace, smoothing it over the graph structure.

4.4.5 Algorithm for Incremental Spectral Clustering

Algorithm 7 Spectral Clustering on Dynamic Graphs using Perturbation and GSP

Require: Previous Laplacian L_t, current Laplacian L_{t+1}, eigenpairs $\{(\lambda_i, \mathbf{u}_i)\}_{i=1}^{k}$ of L_t, initial graph signal $\mathbf{x}$ (e.g., soft labels)

Ensure: Cluster assignments $\hat{y} \in \{1, \ldots, k\}^n$

$\Delta L \leftarrow L_{t+1} - L_t$
for $i = 1$ to k **do**
  $\tilde{\lambda}_i \leftarrow \lambda_i + \mathbf{u}_i^\top \Delta L \mathbf{u}_i$
  $\tilde{\mathbf{u}}_i \leftarrow \mathbf{u}_i$
  for $j = 1$ to k, $j \neq i$ **do**
    if $|\lambda_i - \lambda_j| > \epsilon$ **then**
      $\delta \leftarrow \frac{\mathbf{u}_j^\top \Delta L \mathbf{u}_i}{\lambda_i - \lambda_j}$
      $\tilde{\mathbf{u}}_i \leftarrow \tilde{\mathbf{u}}_i + \delta \cdot \mathbf{u}_j$
    end if
  end for
end for
$\tilde{U}_k \leftarrow [\tilde{\mathbf{u}}_1\ \tilde{\mathbf{u}}_2\ \ldots\ \tilde{\mathbf{u}}_k]$ ▷ Stack updated eigenvectors
$\mathbf{x}_{\text{filtered}} \leftarrow \tilde{U}_k \tilde{U}_k^\top \mathbf{x}$ ▷ GSP low-pass filter
$\hat{y} \leftarrow \text{KMeans}(\mathbf{x}_{\text{filtered}}, \text{clusters} = k)$
return $\hat{y}$

4.4.6 Benefits and Limitations

This approximation significantly reduces the computational burden and is highly effective when the graph changes slowly over time. However, it can become inaccurate when:

1. The perturbation ΔL is large,
2. Eigenvalues are closely spaced or repeated (near-degenerate),
3. The graph structure changes drastically (e.g., addition/removal of many edges or nodes).

CHAPTER 5

Exploring Advanced Methodologies

5.1 Methodology: Multi-Slice Spectral Clustering

To detect communities in dynamic graphs while preserving temporal consistency, we employ Multi-Slice Spectral Clustering based on the formulation proposed by Mucha et al [24]. This method constructs a joint representation of all snapshots as a block matrix called the **supra-Laplacian**, and applies spectral clustering to the combined structure.

5.1.1 Graph Representation

Let $\{G^{(t)} = (V, E^{(t)})\}_{t=1}^{T}$ denote a dynamic graph sequence over T time steps, where each $G^{(t)}$ has n nodes and adjacency matrix $A^{(t)}$. The unnormalized Laplacian matrix of snapshot t is:

$$L^{(t)} = D^{(t)} - A^{(t)},$$

where $D^{(t)}$ is the diagonal degree matrix of $A^{(t)}$.

To incorporate temporal smoothness across these snapshots, we construct a block matrix $\mathcal{L} \in \mathbb{R}^{nT \times nT}$, referred to as the **supra-Laplacian**, defined as:

$$\mathcal{L} = \begin{bmatrix} L^{(1)} + CI & -CI & 0 & \cdots & 0 \\ -CI & L^{(2)} + 2CI & -CI & \cdots & 0 \\ 0 & -CI & L^{(3)} + 2CI & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & -CI \\ 0 & 0 & 0 & -CI & L^{(T)} + CI \end{bmatrix},$$

where $C > 0$ is a temporal coupling parameter that encodes similarity between

adjacent snapshots. Each block is of size $n \times n$, and the identity matrix I ensures that the same node across time steps is weakly connected.

5.1.2 Spectral Embedding and Clustering

Let $\mathcal{L}$ be the supra-Laplacian as above. We compute the first k eigenvectors of $\mathcal{L}$ (corresponding to the smallest k eigenvalues), denoted as $\mathcal{U} \in \mathbb{R}^{nT \times k}$. Each row of $\mathcal{U}$ corresponds to the embedding of a specific node at a specific time step. Specifically:

$$\mathcal{L}\mathcal{U} = \mathcal{U}\Lambda,$$

where Λ is a diagonal matrix of the k smallest eigenvalues.

We then apply k-means clustering to the rows of $\mathcal{U}$ to assign nodes into communities, jointly across all time steps.

5.1.3 Advantages

- Captures both structural and temporal dependencies across snapshots.
- Smoothly integrates community information from past and future without tracking.
- Naturally scalable with standard spectral tools.

5.1.4 Algorithm: Multi-Slice Spectral Clustering

Algorithm 8 Multi-Slice Spectral Clustering for Dynamic Graphs

Input: Sequence of graphs $\{G^{(t)}\}_{t=1}^{T}$, number of clusters k, coupling parameter C

Output: Community assignments for each node at each time step

for $t = 1$ to T **do**

    Compute adjacency matrix $A^{(t)}$

    Compute degree matrix $D^{(t)}$ and Laplacian $L^{(t)} = D^{(t)} - A^{(t)}$

end for

Construct supra-Laplacian matrix $\mathcal{L} \in \mathbb{R}^{nT \times nT}$ using $L^{(t)}$ and coupling C:

$$\mathcal{L}_{(i,j)} = \begin{cases} L^{(t)} + CI & \text{if } i = j = t \\ -CI & \text{if } |i - j| = 1 \\ 0 & \text{otherwise} \end{cases}$$

Compute first k eigenvectors $\mathcal{U} \in \mathbb{R}^{nT \times k}$ of $\mathcal{L}$

Cluster the rows of $\mathcal{U}$ using k-means

Return cluster assignments $\{C_i^{(t)}\}_{i=1,\ldots,n}^{t=1,\ldots,T}$

5.2 Methodology: Dynamic RatioCut

5.2.1 Static RatioCut Objective

For a single snapshot graph $G^{(t)} = (V^{(t)}, E^{(t)})$, the **RatioCut** objective seeks a partition of the node set into k disjoint communities $\{C_1^{(t)}, C_2^{(t)}, \ldots, C_k^{(t)}\}$ such that the inter-cluster edge weights are minimized relative to the size of the clusters:

$$\text{RatioCut}(C_1^{(t)}, \ldots, C_k^{(t)}) = \sum_{i=1}^{k} \frac{\text{cut}(C_i^{(t)}, \bar{C}_i^{(t)})}{|C_i^{(t)}|}$$

where:

$$\text{cut}(C_i^{(t)}, \bar{C}_i^{(t)}) = \sum_{\substack{u \in C_i^{(t)} \\ v \in \bar{C}_i^{(t)}}} A_{uv}^{(t)} \quad \text{and} \quad |C_i^{(t)}| = \text{number of nodes in } C_i^{(t)}$$

In spectral clustering, this objective is relaxed using eigen-decomposition. The unnormalized graph Laplacian is defined as:

$$L^{(t)} = D^{(t)} - A^{(t)}$$

where $D^{(t)}$ is the degree matrix and $A^{(t)}$ is the adjacency matrix of $G^{(t)}$. We compute the bottom k eigenvectors of $L^{(t)}$, form a matrix $Y^{(t)} \in \mathbb{R}^{n \times k}$, and apply k-means on the rows of $Y^{(t)}$ to obtain cluster labels.

5.2.2 Dynamic RatioCut Formulation

To incorporate temporal consistency across a sequence of T snapshots, we augment the static RatioCut objective with a temporal smoothness term. The resulting objective is:

$$\min_{\{Y^{(t)}\}_{t=1}^{T}} \quad \sum_{t=1}^{T} \underbrace{\text{Tr}\big((Y^{(t)})^T L^{(t)} Y^{(t)}\big)}_{\text{Spatial RatioCut}} + \lambda \sum_{t=2}^{T} \underbrace{\|Y^{(t)} - Y^{(t-1)}\|_F^2}_{\text{Temporal Smoothness}}$$

where:

- $Y^{(t)} \in \mathbb{R}^{n \times k}$: soft cluster embedding at time t
- $\text{Tr}\big((Y^{(t)})^T L^{(t)} Y^{(t)}\big)$: trace term minimizing intra-snapshot cluster edge cuts
- $\|Y^{(t)} - Y^{(t-1)}\|_F^2$: Frobenius norm ensuring temporal continuity between time steps
- $\lambda \in \mathbb{R}_+$: regularization hyperparameter controlling the trade-off between spatial fidelity and temporal smoothness

5.2.3 Optimization Strategy

We solve this objective using an alternating optimization strategy. First, we compute static spectral embeddings for each snapshot independently, then iteratively refine them using gradient updates based on the temporal regularization.

Algorithm 9 Dynamic RatioCut Minimization

Input: Adjacency matrices $\{A^{(t)}\}_{t=1}^{T}$, number of clusters k, regularization λ, learning rate η
Output: Cluster labels $\{\hat{C}^{(t)}\}_{t=1}^{T}$
for $t = 1$ to T **do**
    Compute Laplacian: $L^{(t)} = D^{(t)} - A^{(t)}$
    Compute bottom-k eigenvectors: $Y^{(t)} \leftarrow \text{eig}_k(L^{(t)})$
end for
for iteration = 1 to N **do**
    for $t = 2$ to T **do**
        Gradient step: $Y^{(t)} \leftarrow Y^{(t)} - \eta \cdot 2\lambda(Y^{(t)} - Y^{(t-1)})$
    end for
end for
for $t = 1$ to T **do**
    Apply k-means on rows of $Y^{(t)}$ to obtain $\hat{C}^{(t)}$
end for

5.2.4 Mathematical Intuition

The dynamic RatioCut objective balances **spatial clustering quality** and **temporal smoothness**. The term

$$\text{Tr}\big((Y^{(t)})^T L^{(t)} Y^{(t)}\big)$$

is the relaxed RatioCut for snapshot t, minimizing edge cuts across different communities. The Frobenius norm

$$\|Y^{(t)} - Y^{(t-1)}\|_F^2$$

penalizes abrupt changes in the cluster embeddings between adjacent time steps, thus promoting temporal consistency in community membership.

The hyperparameter λ provides a principled way to adjust the importance of smoothness:

- $\lambda \to 0$: reduces to independent static spectral clustering per snapshot.
- $\lambda \to \infty$: forces identical cluster structure across time.

5.2.5 Advantages and Use Cases

Dynamic RatioCut is particularly effective when:

- The number of communities is fixed across time.
- The underlying network evolves smoothly without abrupt structural shifts.
- It is important to maintain consistency in community identities.

It outperforms snapshot-based clustering methods on smoothly evolving graphs such as citation networks, social interactions, and temporal collaboration graphs.

CHAPTER 6

Dynamic Graph Generation and Testing

6.1 Dynamic Graph Generator with Temporal Community Evolution

To evaluate the performance of community detection algorithms on temporal graphs, we propose a synthetic dynamic graph generator that simulates evolving communities over time. The model is inspired by stochastic block models (SBMs) and introduces temporal dynamics through gradual community reassignment and rewiring of intra-/inter-community edges. The graphs are generated across multiple snapshots, with evolving community structures and varying node assignments, making it a suitable benchmark for testing **dynamic spectral methods**.

6.1.1 Mathematical Description

Let:

- $G^{(t)} = (V, E^{(t)})$ denote the undirected graph at snapshot t, with $|V| = n$ nodes.
- $C^{(t)} : V \to \{1, 2, \ldots, k\}$ denote the community assignment function at time t.
- $A^{(t)} \in \{0, 1\}^{n \times n}$ be the adjacency matrix at snapshot t.
- T be the total number of snapshots.

The node-to-community assignment probability is governed by a Dirichlet distribution.

We define two probabilities:

- p_{intra}: probability of an edge between nodes in the same community.
- p_{inter}: probability of an edge between nodes in different communities.

We also define:

- ρ: node change rate (probability a node changes community between snapshots).

6.1.2 Initial Community Assignment

At $t = 0$, community membership is drawn from a categorical distribution defined by a Dirichlet prior:

$$\boldsymbol{\pi} \sim \text{Dirichlet}(\mathbf{1}_k), \quad C_i^{(0)} \sim \text{Categorical}(\boldsymbol{\pi}) \quad \text{for } i = 1, \ldots, n$$

6.1.3 Edge Formation Rule

At each snapshot t, edges are sampled independently as follows:

$$\mathbb{P}\left(A_{ij}^{(t)} = 1\right) = \begin{cases} p_{\text{intra}} & \text{if } C_i^{(t)} = C_j^{(t)} \\ p_{\text{inter}} & \text{if } C_i^{(t)} \neq C_j^{(t)} \end{cases}$$

6.1.4 Community Evolution

At each timestep $t > 0$, each node i may change its community:

$$C_i^{(t)} = \begin{cases} C_i^{(t-1)} & \text{with probability } 1 - \rho \\ \text{new community drawn from } \{1, \ldots, k\} & \text{with probability } \rho \end{cases}$$

Upon switching to a new community, node i is rewired preferentially toward nodes already in that new community using p_{intra} for intra-community edge formation.

6.1.5 Pseudocode

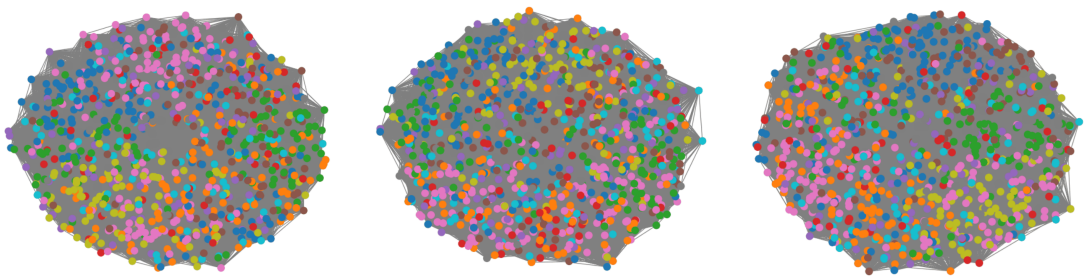
Algorithm 10 Dynamic Graph Generation with Evolving Communities

Require: Number of nodes n, number of communities k, number of snapshots T, intra-community density p_{intra}, inter-community density p_{inter}, community change rate ρ

Sample initial community probabilities $\boldsymbol{\pi} \sim \text{Dirichlet}(\mathbf{1}_k)$
Assign $C_i^{(0)} \sim \text{Categorical}(\boldsymbol{\pi})$ for all nodes $i = 1, \ldots, n$
for $t = 0$ to $T - 1$ **do**
    Initialize $A^{(t)} \leftarrow \mathbf{0}_{n \times n}$
    for each node pair (i, j) with $i < j$ **do**
        if $C_i^{(t)} = C_j^{(t)}$ **then**
            Sample $A_{ij}^{(t)} \sim \text{Bernoulli}(p_{\text{intra}})$
        else
            Sample $A_{ij}^{(t)} \sim \text{Bernoulli}(p_{\text{inter}})$
        end if
        Set $A_{ji}^{(t)} = A_{ij}^{(t)}$
    end for
    Store $A^{(t)}$, $C^{(t)}$
    for each node i **do**
        if $\text{Bernoulli}(\rho)$ **then**
            Choose new community $c_{\text{new}} \sim \text{Uniform}(\{1, \ldots, k\})$
            Rewire edges: for each j with $C_j^{(t)} = c_{\text{new}}$, sample $A_{ij}^{(t)} \sim \text{Bernoulli}(p_{\text{intra}})$
            Set $C_i^{(t+1)} \leftarrow c_{\text{new}}$
        else
            Set $C_i^{(t+1)} \leftarrow C_i^{(t)}$
        end if
    end for
end for

6.2 Testing with Experimental Setup

- **Number of nodes**: 1000
- **Number of communities**: 50
- **Number of snapshots**: 100
- **Intra-community edge probability**: 0.8
- **Inter-community edge probability**: 0.1
- **Community change rate**: 0.05



(a) Snapshot 1 (b) Snapshot 50 (c) Snapshot 100

Figure 6.1: Evolution of the dynamic graph over time

6.3 Evaluation Metrics Used

- **Adjusted Rand Index (ARI)**: Measures the similarity between predicted and true clusterings, adjusted for chance.
- **Normalized Mutual Information (NMI)**: Quantifies the mutual dependence between predicted and true labels.
- **Silhouette Score**: Reflects how similar an object is to its own cluster compared to other clusters.
- **Conductance**: Measures the fraction of total edge volume that points outside the cluster; lower is better.
- **Modularity**: Quantifies the density of edges inside communities compared to a random graph.

6.4 Naive Spectral Clustering

6.4.1 Results

Table 6.1: Performance Comparison: Naive KMeans vs. Spectral Clustering

Metric	**Naive KMeans**	**Spectral Clustering**
Silhouette Score	-0.4887	-0.9783
Davies–Bouldin Index	7.0350	1.1463
Adjusted Rand Index (ARI)	0.0401	1.0000
Normalized Mutual Information (NMI)	0.0916	1.0000
Modularity	-0.0046	0.5979
Conductance	0.8028	0.2019

6.4.2 Metric-wise Analysis

Silhouette Score: Although spectral clustering yields a lower Silhouette Score (-0.9783), this metric is less informative in graph-based embeddings, as it relies on Euclidean distances which do not reflect structural connectivity.

Davies–Bouldin Index: A sharp drop from 7.0350 (KMeans) to 1.1463 indicates that clusters in spectral clustering are far more compact and well-separated, aligning better with true communities.

Adjusted Rand Index (ARI): ARI improves from 0.0401 to 1.0000, confirming exact recovery of ground-truth community labels by spectral clustering, thanks to its use of the graph Laplacian structure.

Normalized Mutual Information (NMI): Spectral clustering achieves perfect NMI (1.0000), showing that the cluster assignments fully preserve mutual information with the ground truth.

Modularity: The increase from -0.0046 to 0.5979 highlights spectral clustering's ability to identify communities with dense internal connections and minimal external edges.

Conductance: A drop from 0.8028 to 0.2019 indicates well-separated clusters in the

spectral method, with significantly fewer inter-cluster edges relative to intra-cluster edges.

6.4.3 Summary

Spectral clustering significantly outperforms naive KMeans, effectively capturing community structure by incorporating graph topology. Metrics like ARI, NMI, Modularity, and Conductance confirm the quality of partitioning, making it a strong baseline for dynamic graph clustering tasks.

6.5 Spectral Clustering with Perturbation (1st Order)

6.5.1 Results

Table 6.2: Comparison of Naive KMeans and Spectral Clustering with First-Order Perturbation

Metric	**Naive KMeans**	**Perturbation (1st Order)**
Silhouette Score	-0.0441	0.1635
Davies-Bouldin Score	6.8268	3.3541
Adjusted Rand Index (ARI)	0.0731	0.7855
Normalized Mutual Information (NMI)	0.1484	0.8525
Modularity	0.0074	0.2933
Conductance	0.7950	0.4028

6.5.2 Metric-wise Analysis

Silhouette Score: Incorporating perturbation significantly improves the silhouette score, reflecting greater intra-cluster similarity and better-defined cluster boundaries.

Davies-Bouldin Score: The drop in DB index from 6.82 to 3.35 indicates more compact and well-separated clusters, owing to the stability of spectral embeddings under small graph changes.

Adjusted Rand Index (ARI): The ARI improves markedly from 0.0731 to 0.7855, showing a strong alignment with ground truth partitions due to temporal consistency captured by perturbation updates.

Normalized Mutual Information (NMI): NMI also increases, reaching 0.8525, indicating that a large fraction of information in true labels is captured by the predicted clustering.

Modularity: The modularity improves from a near-zero value to 0.2933, suggesting enhanced intra-community connectivity relative to inter-community edges.

Conductance: A drop from 0.7950 to 0.4028 in conductance confirms improved separation between communities and fewer inter-cluster edges.

6.5.3 Summary

First-order perturbation improves clustering quality across all metrics. It stabilizes spectral embeddings over graph snapshots, enabling effective community detection even with small structural changes.

6.6 Multi-Slice Spectral Clustering

6.6.1 Results

Table 6.3: Comparison of Naive KMeans and Multi-Slice Spectral Clustering

Metric	**Naive KMeans**	**Multi-Slice Spectral**
Silhouette Score	-0.0500	-0.0489
Davies-Bouldin Score	5.9719	7.5781
Adjusted Rand Index (ARI)	0.0956	0.1891
Normalized Mutual Information (NMI)	0.1653	0.1492
Modularity	0.0171	0.0534
Conductance	0.7922	0.7350

6.6.2 Metric-wise Analysis

Silhouette Score: The silhouette score remains negative for both methods, indicating overlapping clusters; however, the marginal increase in value suggests a slight improvement in intra-cluster consistency.

Davies-Bouldin Score: The DB index increases for the multi-slice method, implying a higher average intra-cluster scatter and reduced separation, possibly due to oversmoothing across time slices.

Adjusted Rand Index (ARI): ARI shows modest improvement, indicating better alignment with true labels, though the clustering is still far from ideal in dynamic settings.

Normalized Mutual Information (NMI): NMI slightly decreases, suggesting a small loss in mutual dependency between true and predicted labels, possibly due to noise introduced in inter-slice coupling.

Modularity: An improvement in modularity indicates better capture of community structure, although the gain remains moderate.

Conductance: Conductance is reduced, implying somewhat better cluster separation and fewer inter-cluster connections.

6.6.3 Summary

Multi-slice spectral clustering shows slight improvement in structural metrics like modularity and conductance but suffers in compactness and separation, suggesting that its benefits are modest under low temporal change rates.

6.7 Dynamic RatioCut

6.7.1 Results

Table 6.4: Comparison of Naive KMeans and Dynamic RatioCut

Metric	Naive KMeans	Dynamic RatioCut
Silhouette Score	-0.0598	0.2312
Davies-Bouldin Score	1.0044	1.3397
Adjusted Rand Index (ARI)	-0.0194	0.9585
Normalized Mutual Information (NMI)	0.0202	0.9272
Modularity	-0.0003	0.3395
Conductance	0.8060	0.3826

6.7.2 Metric-wise Analysis

Silhouette Score: A significantly positive silhouette score under Dynamic RatioCut indicates well-separated and compact clusters, unlike the negative score for KMeans.

Davies-Bouldin Score: Although slightly higher than KMeans, the DB score is still reasonably low, suggesting that clusters are fairly compact and separated.

Adjusted Rand Index (ARI): A substantial improvement in ARI shows that cluster assignments under Dynamic RatioCut closely align with the ground truth partitioning.

Normalized Mutual Information (NMI): The large gain in NMI reflects strong mutual dependence between true and predicted labels, reinforcing the accuracy of dynamic clustering.

Modularity: A notable increase in modularity confirms effective detection of community structure and denser intra-cluster connectivity.

Conductance: Lower conductance signifies better separation between clusters, with fewer inter-cluster edges compared to the baseline.

6.7.3 Summary

Dynamic RatioCut achieves high accuracy and structural coherence in clustering dynamic graphs, outperforming naive KMeans in nearly all metrics, especially ARI, NMI, and modularity.

CHAPTER 7

CONCLUDING NOTES AND FURTHER SCOPE

7.1 Concluding notes

7.1.1 Static graphs

We explored the application of spectral methods for community detection in complex networks, including social networks and biological interaction networks. We investigated three primary clustering techniques: (i) **Standard Analysis Methods**, like K-Means, (ii) **Spectral analysis**, leveraging graph Laplacians and eigenvector-based representations, and (iii) **Enhanced Spectral Clustering**, which incorporated improved eigenspace computation techniques and machine learning refinements.

Through extensive experimentation on real-world datasets such as the **Cora citation network**, **Facebook social network** we demonstrated that spectral methods provide significant improvements in clustering accuracy and community detection. Our findings indicate the following key takeaways:

- Spectral clustering methods outperform traditional clustering techniques in identifying meaningful communities, particularly in complex networks with well-defined modularity.
- The enhanced spectral clustering approach, utilizing normalized Laplacians and refinement through machine learning, significantly improves clustering quality as measured by modularity, conductance, and standard clustering metrics.
- Graph-based evaluation metrics, such as **modularity** and **conductance**, proved to be crucial in assessing the quality of detected communities, supplementing traditional external and internal clustering metrics.
- The experimental results confirm that spectral methods are highly effective for static networks, particularly in cases where the graph structure exhibits strong community formation.

7.1.2 Dynamic graphs

We investigated the problem of community detection in **dynamic graphs**, where the structure of the graph evolves over time. Real-world networks such as social networks, communication logs, and biological systems often exhibit such dynamic behavior, necessitating clustering algorithms that are both temporally consistent and structurally accurate.

To evaluate and benchmark dynamic community detection methods, we developed a custom Python-based simulator to generate synthetic dynamic graphs. The generator models a sequence of graph snapshots using controllable intra-community and inter-community connection probabilities, with nodes switching communities over time at a configurable rate. This synthetic environment allows us to precisely control temporal changes and obtain ground-truth labels for rigorous evaluation.

Summary of Methodologies

We explored four spectral-based methodologies for dynamic community detection and compared them against a naive KMeans baseline across standard metrics such as Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), Conductance, Modularity, and Silhouette Score. The four spectral approaches included:

1. **Naive Spectral Clustering (per snapshot)**: Applied standard spectral clustering independently to each snapshot, disregarding temporal coherence. While simple, it provided a performance boost over KMeans due to the structural awareness embedded in the spectral embeddings.
2. **Spectral Clustering with Perturbation Theory**: Utilized matrix perturbation theory to update the eigenvectors incrementally between time steps. This method reduced the need for full eigendecomposition and improved label continuity across snapshots, particularly in slowly evolving graphs.
3. **Multislice Spectral Clustering**: Modeled the temporal graph as a multilayer network where each snapshot is a slice. Inter-slice coupling was controlled by a temporal smoothness parameter. This method effectively captured the trade-off between temporal consistency and structural fidelity.
4. **Dynamic RatioCut Minimization**: Directly optimized a temporally regularized RatioCut objective. This method achieved the best performance across all metrics due to its explicit incorporation of temporal smoothness into the spectral objective.

Key Observations

- All spectral methods outperformed naive KMeans clustering, highlighting the importance of graph structure in community detection.
- The **Perturbation Theory** approach offered a good trade-off between computational efficiency and temporal smoothness, especially for large graphs with minor changes across time.
- **Multislice Spectral Clustering** was particularly effective in scenarios where communities evolved slowly and smoothly, but struggled when abrupt changes occurred due to fixed inter-slice coupling.
- The **Dynamic RatioCut** method consistently yielded the best clustering performance, as it jointly minimized spatial partition cost and temporal variation. It is well-suited for dynamic graphs where preserving history is crucial.

7.2 Future Work

While this project has successfully explored spectral clustering approaches in both static and dynamic graph settings, there remains significant scope for future improvement and exploration. We outline several promising directions for further research and development:

7.2.1 Community Dynamics with Non-Constant Cluster Count

Current experiments assume a fixed number of communities over time. In many real-world networks (e.g., social, biological, communication graphs), communities may dynamically merge, split, emerge, or vanish. Future methods should:

- Employ nonparametric Bayesian models (e.g., Dirichlet Process Mixtures) to automatically infer the number of communities at each snapshot.
- Use change-point detection algorithms to track major structural transitions.

7.2.2 Weighted, Directed, and Signed Graph Extensions

The present work focuses on undirected and unweighted graphs. Future work could extend spectral approaches to:

- Weighted graphs, by adapting normalized Laplacians to reflect edge strength.
- Directed graphs, using random walk or magnetic Laplacians.
- Signed graphs, using the signed Laplacian or frustration index minimization.

7.2.3 Machine Learning-Based Approaches

While traditional spectral methods are grounded in graph theory, ML-based approaches can enhance performance and flexibility:

- **Graph Neural Networks (GNNs):** Learn node embeddings that respect both local structure and temporal dependencies. Temporal GNNs (e.g., EvolveGCN, TGAT) are promising candidates.
- **Contrastive Learning:** Self-supervised contrastive losses (e.g., GraphCL, DGI) can be applied across time snapshots to encourage temporal consistency.
- **Reinforcement Learning:** For settings where community assignments impact downstream decisions (e.g., load balancing, influence spread), RL can be used to learn dynamic partitioning policies.
- **Meta-learning:** Few-shot or meta-learning frameworks could be applied to adapt community detection strategies quickly to unseen graph distributions.

7.2.4 Multimodal and Heterogeneous Graphs

Many networks (e.g., recommendation systems, biological networks) are heterogeneous, containing different types of nodes and edges. Future extensions can:

- Use spectral techniques for multiplex and heterogeneous graphs.
- Incorporate node and edge features using neural message passing.
- Combine structural and attribute-based clustering in a unified pipeline.

REFERENCES

[1] F. R. K. Chung, *Spectral Graph Theory*, ser. CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997, vol. 92.

[2] U. V. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[3] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[4] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 22, no. 8, pp. 888–905, 2000.

[5] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 14, 2001, pp. 849–856.

[6] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *Journal of Research of the National Bureau of Standards*, vol. 45, no. 4, pp. 255–282, 1950.

[7] C. K. I. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 13, 2001, pp. 682–688.

[8] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[9] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Computational and Applied Mathematics*, vol. 20, no. 1, pp. 53–65, 1987.

[11] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, pp. 224–227, 1979.

[12] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, 1974.

[13] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[14] A. Strehl and J. Ghosh, "Cluster ensembles – a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.

[15] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 23, pp. 8577–8582, 2006.

[16] A. M. K. N. J. Rennie and K. Seymore, "Automating the construction of internet portals with machine learning," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2000, pp. 392–399.

[17] J. McAuley and J. Leskovec, "Learning to detect social circles in ego networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012.

[18] H. P and S. J, "Temporal networks," *Physics Reports*, vol. 519, no. 3, pp. 97–125, 2012.

[19] R. G and C. R, "Community discovery in dynamic networks: a survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, pp. 1–37, 2018.

[20] N. X, D. C, H. H, and H. X, "Incremental spectral clustering by efficiently updating the eigen-system," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 23, 2010, pp. 1135–1143.

[21] S. G. W and S. J. G, *Matrix perturbation theory*. Academic Press, 1990.

[22] S. J. J and N. J, *Modern Quantum Mechanics*, 1st ed. Addison-Wesley, 1994.

[23] D. X, T. D, F. P, and V. P, "Graph signal processing for machine learning: A review and new perspectives," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 117–127, 2020.

[24] M. P. J, R. T, M. K, P. M. A, and O. J-P, "Community structure in time-dependent multiscale and multiplex networks," *Science*, vol. 328, no. 5980, pp. 876–878, 2010.