University of California
Los Angeles

Graph Based Models for Unsupervised High Dimensional Data Clustering and Network Analysis

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

Huiyi Hu

2015

Abstract of the Dissertation

Graph Based Models for Unsupervised High Dimensional Data Clustering and Network Analysis

by

Huiyi Hu

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2015

Professor Andrea L. Bertozzi, Chair

The demand for analyzing patterns and structures of data is growing dramatically in recent years. The study of network structure is pervasive in sociology, biology, computer science, and many other disciplines. My research focuses on network and high-dimensional data analysis, using graph based models and tools from sparse optimization. The specific question about networks we are studying is "clustering": partitioning a network into cohesive groups. Depending on the contexts, these tightly connected groups can be social units, functional modules, or components of an image.

My work consists of both theoretical analysis and numerical simulation. We first analyze some social network and image datasets using a quality function called "modularity", which is a popular model for clustering in network science. Then we further study the modularity function from a novel perspective: with my collaborators we reformulate modularity optimization as a minimization problem of an energy functional that consists of a total variation term and an L_2 balance term. By employing numerical techniques from image processing and L_1 compressive sensing, such as the Merriman-Bence-Osher (MBO) scheme, we develop a variational algorithm for the minimization problem.

Along a similar line of research, we work on a multi-class segmentation problem using the piecewise constant Mumford-Shah model in a graph setting. We propose an efficient algorithm for the graph version of Mumford-Shah model using the MBO scheme. Theoretical analysis is developed and a Lyapunov functional is proven to decrease as the algorithm proceeds. Furthermore, to reduce the computational cost for large datasets, we incorporate the Nyström extension method to efficiently approximates eigenvectors of the graph Laplacian based on a small portion of the weight matrix. Finally, we implement the proposed method on the problem of chemical plume detection in hyper-spectral video data. These graph based clustering algorithms we proposed improve the time efficiency significantly for large scale datasets. In the last chapter, we also propose an incremental reseeding strategy for clustering, which is an easy-to-implement and highly parallelizable algorithm for multiway graph partitioning. We demonstrate experimentally that this algorithm achieves state-of-the-art performance in terms of cluster purity on standard benchmark datasets. Moreover, the algorithm runs an order of magnitude faster than the other algorithms.

The dissertation of Huiyi Hu is approved.

Stanley Osher

Luminita Vese

P. Jeffrey Brantingham

Andrea L. Bertozzi, Committee Chair

University of California, Los Angeles

2015

To my parents . . .

who have always loved me unconditionally

and raised me to be an independent woman.

To my dearest friend Daniel Murfet . . .

for being a constant source of encouragement and support

during the challenges of graduate school and life.

Table of Contents

List of Figures

List of Tables

Acknowledgments

My research work is supervised by my Ph.D. advisor Dr. Andrea L. Bertozzi. I am very grateful to her kind support and guidance during all these years in graduate school. She has been an inspiring role model for me. I also thank all of my committee members, Dr. Stanley Osher, Dr. Luminita Vese and Dr. Jeffrey Brantingham for their useful advises and support. This thesis presents the collaborative work with Mason A. Porter, Yves van Gennip, Blake Hunter, Thomas Laurent, James von Brecht, Arthur Szlam and Xavier Bresson. I thank them for mentoring and guiding me during my Ph.D. research.

Chapter 3 is a version of [47, 91] where I implement the network analysis tests under the mentoring of Y. van Gennip, B. Hunter, M. A. Porter and my advisor. The text of [47] is primarily written by me. Chapter 4 is a version of [45] where my contribution is to develop the reformulation of modularity with T. Laurent, to derive the proof in the section of Γ- convergence, and to implement the numerical experiments under the supervision of M. A. Porter and my advisor. The text is primarily written by me. Chapter 5 is a version of [46] where my contribution is all of the analytical derivation and performing the numerical experiments under my advisor's supervision. One of the coauthors J. Sunu computes the eigenvectors using the Nyström method. The text is written by me. Chapter 6 is a version of the submitted manuscript [13], where my contribution is experimenting and developing the proposed INCRES algorithm, and collecting tests results on the benchmark datasets.

My Ph.D. research work is supported by: NSF grants DMS-1109805, DMS-1118971, DMS-1045536, DMS-0968309 and DMS-1417674, UC Lab Fees Research grant 12-LR-236660, ONR grants N000141010221, N000141210040 and N0001412-10838, AFOSR MURI grant FA9550-10-1-0569, and the W. M. Keck Foundation.

Vita

2006–2009	First Class Academic Excellence Award (three times), Zhejiang University, China
2009–2010	Chu Kochen Fellowship, Zhejiang University, China
2010 Spring	B. S. in Mathematics, Zhejiang University, China
2011– 2012	Teaching Assistant, Department of Mathematics, University of California, Los Angeles, USA
2012 Summer	Undergraduate Research Project Mentor, REU, University of California, Los Angeles, USA
2012– 2014	Research Assistant, Department of Mathematics, University of California, Los Angeles, USA
2013 Summer	Research Intern, Los Alamos National Laboratory, USA
2014 Summer	Research Intern, Imagination Laboratory, Adobe Systems, USA
2014– 2015	Dissertation Year Fellowship, University of California, Los Angeles, USA

Publications

- (Preprint) X. Bresson, H. Hu, T. Laurent, A. Szlam, J. von Brecht, "An Incremental Reseeding Strategy for Clustering", arXiv:1406.3837, 2014.

- H. Hu, J. Sunu, A. L. Bertozzi, "Multi-class Graph Mumford-Shah Model for Plume Detection Using the MBO scheme", Proceedings of the International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition, Lecture Notes in Computer Science (LNCS), Springer, 2015, pp. 209-222.

- H. Hu, B. Wohlberg, R. Chartrand, "Task-Driven Dictionary Learning and Inpainting", Proceedings of the International Conference on Acoustics, Speech and Signal Processing, 2014, pp. 3543-3547.

- H. Hu, T. Laurent, M. A. Porter, A. L. Bertozzi, "A Method Based on Total Variation for Network Modularity Optimization Using the MBO Scheme", SIAM Journal on Applied Mathematics, Vol. 73, Issue 6, 2013, pp. 2224-2246.

- H. Hu, Y. van Gennip, B. Hunter, M. A. Porter, A. L. Bertozzi, "Multislice Modularity Optimization in Community Detection and Image Segmentation", Proceedings of the IEEE International Conference on Data Mining Workshops, 2012, pp. 934-936.

- Y. van Gennip, H. Hu, B. Hunter, M. A. Porter, "Geosocial Graph-Based Community Detection", Proceedings of the IEEE International Conference on Data Mining Workshops, 2012, pp. 754-758.

- K. Han, H. Hu, E. Ko, A. O. Ozer, C. Simon, C. Tan, "A Variational Approach to Modeling Aircraft Hoses and Flexible Conduits", Mathematics-in-Industry Case Studies, Vol. 4, 2012, pp. 1-13.

CHAPTER 1

Introduction

One of the most basic unsupervised learning tasks is to automatically partition data into clusters based on pair-wise similarity. A standard scenario is to use a weighted graph to represent the data. This thesis particularly focuses on unsupervised, non-overlapping data clustering tasks.

Data clustering has been a widely studied problem in various fields. It takes different forms depending on the context and application, such as community detection in network science, segmentation problem in image processing, graph partitioning in parallel computing. My work starts from a network science point of view using graph models. Pursuing a similar line of research, we extend the developed graph tools to a multi-class segmentation model in the context of hyperspectral imagery segmentation and general high-dimensional data clustering. At last, we also study an incremental reseeding strategy for clustering that produces high performance with very low computational cost.

1.1 Network Community Detection

Networks provide a useful representation for the investigation of complex systems, and they have accordingly attracted considerable attention in sociology, biology, computer science, and many other disciplines [70, 71]. Most of the networks that people study are graphs, which consist of nodes (i.e., vertices) to represent the elementary units of a system and edges to represent pairwise connections or in-

teractions between the nodes.

Using networks makes it possible to examine intermediate-scale structure in complex systems. Most investigations of intermediate-scale structures have focused on *community structure*, in which one decomposes a network into (possibly overlapping) cohesive groups of nodes called *communities* [76]. There is a higher density of connections within communities than between them.

In some applications, communities have been related to functional units in networks [76]. For example, a community might be closely related to a functional module in a biological system [55] or a group of friends in a social system [88]. Because community structure in real networks can be very insightful [33,37,71,76], it is useful to study algorithmic methods to detect communities. Such efforts have been useful in studies of the social organization in friendship networks [88], legislation cosponsorships in the United States Congress [100], functional modules in biology networks [39, 55], and many other situations.

To perform community detection, one needs a quantitative definition for what constitutes a community, though this relies on the goal and application that one has in mind. Perhaps the most popular approach is to optimize a quality function known as *modularity* [68,72,73], and numerous computational heuristics have been developed for optimizing modularity [33, 76]. The modularity of a network partition measures the fraction of total edge weight within communities versus what one might expect if edges were placed randomly according to some null model. Modularity gives one definition of the "quality" of a partition, and maximizing modularity is supposed to yield a reasonable partitioning of a network into disjoint communities.

Community detection is related to *graph partitioning*, which has been applied to problems in numerous areas [74, 81, 95]. In graph partitioning, a network is divided into disjoint sets of nodes. Graph partitioning usually requires the number of clusters to be specified to avoid trivial solutions, whereas modularity optimiza-

tion does not require one to specify the number of clusters [76]. This is a desirable feature for applications such as social and biological networks.

The idea of modularity is generalized to a "multi-slice network" version in [64], which consists of layers of ordinary networks in which vertex x in one slice is connected to the corresponding vertex in other slices, via a coupling constant. Each layer can represent relationships induced from a different type of feature of the agents, the status of the network at a different temporal point, or the network inspected under a different scale. The multi-slice modularity model allows us to cluster multiple layers of a network simultaneously, while enforcing some consistency in clustering identical vertices similarly across slices. In the work of [47, 91], we implement the multi-slice modularity on a dataset with both geographic and social information about stops involving street gang members in the Los Angeles Police Department (LAPD) Division of Hollenbeck [92]. Without prior knowledge of the number of gangs of the members, we examined network diagnostics over slices to attempt to estimate the number of gangs that is stable across multiple scales, which turns out to correspond roughly to the number expected by the LAPD. We also applied this technique to image segmentation and tried to determine the number of components in a test image through the multi-slice network model [47].

Because modularity optimization is an NP-hard problem [12], efficient algorithms are necessary to find good locally optimal network partitions with reasonable computational costs. Numerous methods have been proposed [33, 76]. These include greedy algorithms [23, 67], extremal optimization [11, 28], simulated annealing [40, 51], spectral methods (which use eigenvectors of a modularity matrix) [68, 79], and more. The locally greedy algorithm by Blondel et al. [9] is arguably the most popular computational heuristic; it is a very fast algorithm, and it also yields high modularity values [33, 53].

In [45], we interpret modularity optimization (using the Newman-Girvan null

model [71, 73]) from a novel perspective. Inspired by the connection between graph cuts and the total variation (TV) of a graph partition, we reformulate the problem of modularity optimization as a minimization of an energy functional that consists of a graph cut (i.e., TV) term and an L_2 balance term. By employing numerical techniques from image processing and L_1 compressive sensing—such as convex splitting and the Merriman-Bence-Osher (MBO) scheme [62,63]we propose a variational algorithm to perform the minimization on the new formula. (The MBO was originally introduced to approximate motion by mean curvature of an interface in Euclidean space.) We apply this method to both synthetic benchmark networks and real data sets, and we achieve performance that is competitive with the state-of-the-art modularity optimization algorithms.

1.2 Hyper-spectral Image Segmentation

Multi-class segmentation has been studied as an important problem for many years in various areas, such as computer science and machine learning. For imagery data in particular, the Mumford-Shah model [65] is one of the most extensively used model in the past decade. This model approximates the true image by an optimal piecewise smooth function through solving a energy minimization problem. More detailed review of the work on Mumford-Shah model can be found in the references of [20]. A simplified version of Mumford-Shah is the piecewise constant model (also known as the "minimal partition problem"), which is widely used due to its reduced complexity compared to the original one. For a given contour Φ which segments an image region Ω into $\hat{n}$ many disjoint sub-regions $\Omega = \cup_{r=1}^{\hat{n}} \Omega_r$, the *piecewise constant Mumford-Shah* energy is defined as:

$$E^{\mathrm{MS}}(\Phi, \{c_r\}_{r=1}^{\hat{n}}) = |\Phi| + \lambda \sum_{r=1}^{\hat{n}} \int_{\Omega_r} (u_0 - c_r)^2 \, , \tag{1.1}$$

where u_0 is the observed image data, $\{c_r\}_{r=1}^{\hat{n}}$ is a set of constant values, and $|\Phi|$ denotes the length of the contour Φ. By minimizing the energy E^{MS} over

Φ and $\{c_r\}_{r=1}^{\hat{n}}$, one obtains an optimal function which is constant within each sub-region to approximate u_0, along with a segmentation given by the optimal Φ. In [21], a method of active contours without edges is proposed to solve for the two-class piecewise constant Mumford-Shah model ($\hat{n} = 2$), using a level set method introduced in [75]. The work in [21] is further generalized to a multi-class scenario in [93]. The method developed in [21,93] is well known as the Chan-Vese model, which is a popular and representative method for image segmentation. The Chan-Vese method has been widely used due to the model's flexibility and the great success it achieves in performance.

In the work of [46], we study the multi-class segmentation problem using the Mumford-Shah model in the context of imagery or general hight-dimensional datasets, in particular hyper-spectral data. It can be viewed as a clustering problem with Mumford-Shah model being the quality function (in contrast to modularity). We formulate the piecewise constant MS problem in a graph setting instead of a continuous one, and propose an efficient algorithm to solve it. Recently the authors of [8] introduced a binary semi-supervised segmentation method based on minimizing the Ginzburg-Landau functional on a graph. Inspired by [8], a collection of work has been done on graph-based high-dimensional data clustering problems posed as energy minimization problems, such as semi-supervised methods studied in [35, 59, 60] and an unsupervised network clustering method [45] known as modularity optimization. These methods make use of graph tools [22] and efficient graph algorithms, and our work pursues similar ideas. Note that unlike the Chan-Vese model which uses $log_2(\hat{n})$ many level set functions and binary representations to denote multiple classes, our model uses simplex constrained vectors for class assignments representation. To solve the multi-class piecewise constant MS variational problem in the graph setting, we propose an efficient algorithm adopting the idea of the MBO scheme [62,63]. The MBO scheme is used on the continuous MS model [29,85] motivated by level set methods. The authors

of [35, 45, 59, 60] implement variants of the MBO scheme applied to segmentation problems in a graph setting. Rigorous proofs of convergence of the original MBO scheme in continuous setting can be found in [5, 31] for the binary case, and [30] for the multi-class case. An analogous discussion in a graph setting is given in [90]. Inspired by the work of [30, 90], we develop a Lyapunov functional for our proposed variant of the MBO algorithm, which approximates the graph MS energy. Theoretical analysis is given to prove that this Lyapunov energy decreases at each iteration of our algorithm, until it converges within finitely many steps.

In order to solve for each iteration of the MBO scheme, one needs to compute the weight matrix of the graph as well as the eigenvectors of the corresponding graph Laplacian. However, the computational cost can become prohibitive for large datasets. To reduce the numerical expenses, we implement the Nyström extension method [34] to approximately compute the eigenvectors, which only requires computing a small portion of the weigh matrix. Thus the proposed algorithm is efficient even for large datasets, such as the hyper-spectral video data considered in [46].

The proposed method can be implemented on general high-dimensional data clustering problems. However, in this work the numerical experiment is focused on the detection of chemical plumes in hyper-spectral video data. Detecting harmful gases and chemical plumes has wide applicability, such as in environmental study, defense and national security. However, the diffusive nature of plumes poses challenges and difficulties for the problem. One popular approach is to take advantage of hyper-spectral data, which provides much richer sensing information than ordinary visual images. The hyper-spectral images used in this work were taken from video sequences captured by long wave infrared (LWIR) spectrometers at a scene where a collection of plume clouds is released. Over 100 spectral channels at each pixel of the scene are recorded, where each channel corresponds to a particular frequency in the spectrum ranging from 7,820 nm to 11,700 nm.

The data is provided by the Applied Physics Laboratory at Johns Hopkins University, (see more details in [18]). Prior analysis of this dataset can be found in the works [36, 61, 84, 86]. The authors of [61] implement a semi-supervised graph model using a similar MBO scheme. In this work, each pixel is considered as a node in a graph, upon which the proposed unsupervised segmentation algorithm is implemented. Competitive results are achieved as demonstrated below.

1.3 An Incremental Reseeding Strategy

We propose an easy-to-implement and highly parallelizable algorithm for multiway graph partitioning. Different from the methods studied in the previous chapters, it does not have an explicit global quality function. Instead, this heuristic strategy takes advantage of random sampling in order to reduce the chances that the solutions get stuck at local minima.

The algorithm proceeds by alternating three simple routines in an iterative fashion: diffusion, thresholding, and random sampling. We demonstrate experimentally that the proper combination of these ingredients leads to an algorithm that achieves state-of-the-art performance in terms of cluster purity on standard benchmark datasets. Moreover, the algorithm runs an order of magnitude faster than the other algorithms that achieve comparable results in terms of accuracy.

1.4 Outlines

The rest of this thesis is organized as follows. Chapter 2 introduces the background and preliminaries of the graph setting, the definition and properties of the graph Laplacians. A few representative quality functions such as the graph cuts and the modularity are presented for the general clustering problem. In Chapter 3, a specific method called the multi-slice modularity optimization is implemented on

social and imagery data to study the network structure, and its performance is evaluated with statistical network diagnostics.

In Chapter 4, the techniques of variational methods in partial differential equation and image processing are discussed, so that an alternative approach to the graph-based clustering problem can be developed. More specifically, the Ginzburg-Landau functional and the MBO scheme are introduced in Euclidean space. Then we derive an equivalent formula of modularity optimization as a minimization problem of an energy functional that consists of a total variation term and an L_2 balance term. A graph version of the MBO scheme is presented for solving the minimization problem; and numerical tests with our algorithms on several benchmark and real-world networks are demonstrated.

Chapter 5 introduces the graph formula for the multi-class piecewise constant Mumford-Shah model and relevant notations. A Mumford-Shah MBO scheme is presented as well as the theoretical analysis for a Lyapunov functional which is proven to decrease as the algorithm proceeds; techniques such as Nyström method are also introduced for the purpose of numerical efficiency. At last, the proposed algorithm is tested on the hyper-spectral video data for plume detection problem. The results are then presented and discussed.

Chapter 6 introduces an incremental reseeding strategy for clustering. Comparisons on computational time and performance between the proposed algorithm and several state-of-art methods are presented.

CHAPTER 2

Preliminary

In this chapter we firstly introduce the basic notations for the graph model and some useful properties of spectral graph theory [22, 95]. Then we briefly present several commonly used clustering models based on graphs. The advantage of graph-based methods is that it can reduce the high-dimensionality of the feature space via a similarity metric; It can also reflect non-local properties of the data.

2.1 Graph Basics

A graph is a commonly used mathematical tool to represent a network and relationships between data samples, as illustrated in Figure 2.1. Consider an N-node graph (G, E), which consists of a node set $G = \{n_1, n_2, \ldots, n_N\}$ and an edge set $E = \{w_{ij}\}_{i=1}^{N}$. Each node n_i corresponds to an agent in a given dataset, such as a pixel in an image, or a person in a social network. In a weighted graph, each edge is assigned with a non-negative quantity w_{ij}, representing the similarity between a pair of nodes n_i and n_j. If an edge is not included in the edge set E, it is assumed to be equivalent to having zero similarity. Let $\mathbf{W} = [w_{ij}]$ denote the graph's $N \times N$ *similarity matrix* (or *weight matrix*).

In this work we only focus on undirected graph, i.e. $\mathbf{W}$ is symmetric, $w_{ij} = w_{ji}$. Although the directed graph (i.e. $w_{ij} \neq w_{ij}$) is also studied in the literature for relationships that are not mutual, (such as one user "following" another on Twitter but not the other way around), it is beyond the realm of this work. If not specified

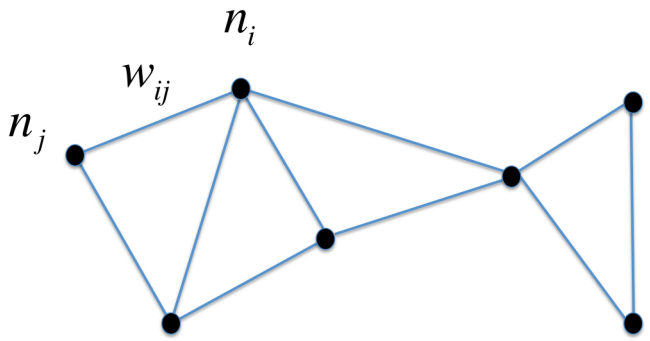


Figure 2.1: Illustration of an undirected weighted graph.

otherwise, the similarity weight w_{ij} is always nonnegative. For unweighted graph, simply assign w_{ij} with value of either one or zero.

2.1.1 Feature vector & similarity metric

In order to build a graph that represents the relationships between agents in the network, one needs a similarity metric to compute the weight w_{ij}, i.e. how to quantify the similarity between node n_i and node n_j. In a dataset, usually each agent (node) comes with a so-called "feature vector" which contains feature information relevant to the problem. In a social network, a feature vector can contain entries about age, gender, occupation, etc. Note that each entry of the feature vector should be normalized accordingly, for example, one person's age and salary have quite different ranges of values and it is reasonable to normalize them to a similar range. A descriptive feature such as gender can be treated as a binary value in the feature vector.

Let vector v_i denotes the feature vector of node n_i. A similarity metric is a mapping from (v_i, v_j) to w_{ij}. A common choice for the similarity metric is

$$w_{ij} = e^{-\frac{\text{dist}^2(v_i,v_j)}{2\sigma^2}}$$

where $\text{dist}(v_i, v_j)$ can be any suitable metric distance between v_i and v_j in Euclidean space, such as L_1 or L_2 norm, or the angle between the two vectors. The

choice of the similarity metric highly depends on the data and the application.

Note that if the dimension of feature vectors is too high, some algorithms may not scale well. In that case, one can reduce the dimensionality using principal component analysis (PCA) [49]. A basic PCA dimension reduction process can be performed as following.

Given a set of data samples with high-dimensional $d \times 1$ features vectors $x_1, x_2, \ldots, x_N \in \mathbb{R}^d$, we want to reduce the dimension from d to $\hat{d}$:

1. Let μ denote the empirical mean of the feature vectors: $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$. Build a $N \times d$ matrix

$$\hat{\mathbf{X}} = (x_1 - \mu, x_2 - \mu, \ldots, x_N - \mu)^T \ .$$

2. Compute the $d \times d$ *scatter matrix* $\mathbf{S} = \hat{\mathbf{X}}^T\hat{\mathbf{X}}$.

3. Because $\mathbf{S}$ is positive semidefinite, one can compute the eigenvectors of $\mathbf{S}$:

$$\mathbf{S} = U^T \Lambda U \ .$$

4. Let $V = (u_1, u_2, \ldots, u_{\hat{d}})$, a $N \times \hat{d}$ matrix, where u_i is i-th eigenvector of $\mathbf{S}$ associated with the largest eigenvalues.

5. The $1 \times \hat{d}$ rows $\{v_i\}_{i=1}^{N}$ of V are the new feature vectors with reduced dimensions.

2.1.2 Graph functions

Graph functions and operators applied on them are important components when studying graph models for clustering problems. In order to represent a partition on the graph, consider a set of graph functions $f = (f_1, f_2, \ldots, f_{\hat{n}}) : G \to \mathbb{R}^{\hat{n}}$:

$$\mathbb{B} := \left\{ f |\ f : G \to \{0,1\}^{\hat{n}}, \sum_{r=1}^{\hat{n}} f_r(n_i) = 1 \right\} \ .$$

The simplex constrained vector value taken by $f \in \mathbb{B}$ can indicate class assignment, i.e. if $f_r(n_i) = 1$ for some r, then n_i belongs to the r-th class. Thus for each $f \in \mathbb{B}$, it corresponds to a partition of the graph G with at most $\hat{n}$ classes; and every partition can be represented by a unique function in $\mathbb{B}$ accordingly. One can think of $\mathbb{B}$ as a a set of generalized binary indicator functions. Indeed, the r-th entry f_r of a function $f \in \mathbb{B}$ is a 1-0 binary indicator function of the r-th class. Thus, a set $\mathbb{B}$ covers all the relvant functions concerning a clustering problem.

However, the strict constraints on $\mathbb{B}$ makes it difficult to apply useful operators. A more general admissible set for graph functions $f = (f_1, f_2, \ldots, f_{\hat{n}}) : G \to \mathbb{R}^{\hat{n}}$ is defined as:

$$\mathbb{K} := \left\{ f |\ f : G \to [0,1]^{\hat{n}}, \sum_{r=1}^{\hat{n}} f_r(n_i) = 1 \right\} ,$$

which is a compact and convex set. Note that the set $\mathbb{B}$ is a subset of $\mathbb{K}$. The graph functions in $\mathbb{K}$ take values on the boundary of a high dimensional simplex, while those in $\mathbb{B}$ take values on the vertex of a simplex.

2.1.3 Graph Laplacian

One of the most commonly used operator on graph functions is the so-called (unnormalized) *graph Laplacian* matrix $\mathbf{L} := \mathbf{D} - \mathbf{W}$ [22], where $\mathbf{D}$ is a diagonal matrix with the i-th entry being the node *strength* (degree) $d_i = \sum_{j=1}^{N} w_{ij}$. A variant of the Laplacian is the symmetric normalized Laplacian $\mathbf{L}_{sym} := \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{I}$ is the identity matrix and assuming every node has nonzero degree (i.e. $d_i > 0$). The Laplacian operators have many useful properties which can give insights to the graph's structure, as studied in depth in spectral graph theory [22]. Here we introduce several commonly used observations of the Laplacian operators.

For a real valued graph function $a : G \to \mathbb{R}$, observe that

$$\langle a, \mathbf{L}a \rangle = \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} \left(a(n_i) - a(n_j) \right)^2 , \tag{2.1}$$

where $\langle \cdot, \cdot \rangle$ is the inner product. For vector-valued $f = (f_1, f_2, \ldots, f_{\hat{n}}) : G \to \mathbb{R}^{\hat{n}}$, let $\langle f, \mathbf{L}f \rangle = \sum_{r=1}^{\hat{n}} \langle f_r, \mathbf{L}f_r \rangle$. One can see from (2.1) that $\mathbf{L}$ is positive semidefinite, so is the operator $\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$:

$$\langle a, \mathbf{L}_{sym} a \rangle = \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} \left(\frac{a(n_i)}{\sqrt{d_i}} - \frac{a(n_j)}{\sqrt{d_j}} \right)^2 . \tag{2.2}$$

The right side of (2.1) is analogous to $\int |\nabla a|^2 dw$ in Euclidean space, with w_{ij} corresponding to the measure dw. Note that the first order derivative of $\int |\nabla a|^2 dw$ yields the negative Laplacian operator $-\Delta$ in heat equations, according to the calculous of variation.

Similar to the Laplacian operator Δ in partial differential equations (PDE), the eigen-space of the graph Laplacians $\mathbf{L}$ and $\mathbf{L}_{sym}$ can reflect a lot of information about the graph. People have been using the leading eigenvectors of the graph Laplacians, (which correspond to the smallest eigenvalues), to partition the graph into meaningful subgroups [95].

Because the Laplacians are positive semidefinite, the smallest possible eigenvalue is zero. Consider a graph with exactly k connected components:

$$G = G_1 \cup G_2 \cup \ldots \cup G_k \,,$$

where any pair of nodes that belong to two different components have zero edge weight between them. Observe that the indicator function χ_{G_s} of each connected component G_s is an eigenvector of $\mathbf{L}$ associated with eigenvalue zero. In fact, for Laplacian $\mathbf{L}$, the eigenvalue zero has the exact algebraic multiplicity of k. In other words, $\mathbf{L}$'s eigen-space of eigenvalue zero has k dimensions, with the k indicator functions $\chi_{G_1}, \chi_{G_2}, \ldots, \chi_{G_k}$ being a set of orthogonal basis. On the one hand, it is trivial to see that the zero eigenvalue has a multiplicity of at least k; on the other hand, Proposition 1 shows that the eigen-space of eigenvalue zero is spanned by the k indicator functions $\chi_{G_1}, \chi_{G_2}, \ldots, \chi_{G_k}$, and thus of dimension k.

Proposition 1. *Every eigenvector of the graph Laplacian* $\mathbf{L}$ *associated with eigenvalue zero must have constant value over each connected component of the graph.*

Proof. Suppose a graph G has k connected components $G = G_1 \cup G_2 \cup \ldots \cup G_k$. Given an eigenvector a that satisfies $\mathbf{L}a = 0$, we want to show that a is constant over G_s, for any $s \in \{1, 2, \ldots, k\}$.

For fixed $s \in \{1, 2, \ldots, k\}$, without loss of generality, one can assume

$$n_1 = \text{argmax}_{n_i \in G_s} \{a(n_i)\} .$$

Because $\mathbf{L}a = 0$ and $w_{1j} = 0$ for $n_j \notin G_s$, we have

$$\begin{aligned} d_1 a(n_1) &= \sum_{j=1}^{N} a(n_j) w_{1j} = \sum_{n_j \in G_s} a(n_j) w_{1j} \\ &\leq \sum_{n_j \in G_s} a(n_1) w_{1j} = a(n_1) \sum_{j=1}^{N} w_{1j} = d_1 a(n_1) . \end{aligned} \tag{2.3}$$

The equality holds if and only if $w_{1j} = 0$ or $a(n_j) = a(n_1)$. It implies that every node n_j in G_s that has non-zero edge connection to n_1 must satisfy $a(n_j) = a(n_1)$. Hence a takes the constant value $a(n_1)$ on G_s, because G_s is a connected component. Thus a is constant on every connected component of the graph G.

□

Therefore, by looking at the orthogonal eigenvectors associated with eigenvalue zero one can obtain the number of connected components in the graph. However, for real world clustering problems, people are more concerned with less ideal situations, where the clusters of the graph are not necessarily fully disconnected from each other. Thus, the goal is to identify the "loosely" connected components in the graph.

In practice, people find the leading eigenvectors associated with small eigenvalues (close to zero) very useful in determining such components [7,22,24,68,81].

One way of interpretation is that these leading eigenvectors approximate the zero-eigen-space in the case discussed above where clusters are fully disconnected, and thus approximate the indicator functions of the clusters. Another way of understanding the use of leading eigenvectors is through the equation (2.1). For a connected graph, the first eigenvector ϕ_1 of the graph Laplacian $\mathbf{L}$ is the constant vector $\mathbf{1}$ (up to a constant), with value one at each entry. The second eigenvector ϕ_2 satisfies:

$$\phi_2 = \text{argmin}_{a\perp\mathbf{1}} \frac{\langle a, \mathbf{L}a\rangle}{\langle a, a\rangle} = \text{argmin}_{\|a\|=1, a\perp\mathbf{1}} \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}\left(a(n_i) - a(n_j)\right)^2 . \qquad (2.4)$$

To minimize the righthand side of equation (2.4), ϕ_2 takes similar value on node n_i and n_j when w_{ij} has a large value. Therefore intuitively speaking, the values of ϕ_2 are grouping together nodes with strong connections, and separating the ones with loose connections. The relation between the graph Laplacian and a quality function called graph cuts is mentioned in Section 2.2.

In many cases the normalized Laplacian $\mathbf{L}_{sym}$ is preferred because it is more regularized, (see normalized cuts in Section 2.2). For non-zero eigenvalues, the corresponding eigenvectors of $\mathbf{L}$ denoted by $\{\phi_s\}$ can not be directly derived from the eigenvectors of $\mathbf{L}_{sym}$ denoted by $\{\hat{\phi}_s\}$, nor the other way around. However, for the zero eigenvalue, one can derive:

$$\begin{aligned}
& \mathbf{L}_{sym}\hat{\phi}_s = \hat{\phi}_s - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\hat{\phi}_s = 0 \\
\Rightarrow\ & \mathbf{D}^{-\frac{1}{2}}\hat{\phi}_s - \mathbf{D}^{-1}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\hat{\phi}_s = 0 \\
\Rightarrow\ & \mathbf{L}(\mathbf{D}^{-\frac{1}{2}}\hat{\phi}_s) = \mathbf{D}(\mathbf{D}^{-\frac{1}{2}}\hat{\phi}_s) - \mathbf{W}(\mathbf{D}^{-\frac{1}{2}}\hat{\phi}_s) = 0 \\
\Rightarrow\ & \phi_s = \mathbf{D}^{-\frac{1}{2}}\hat{\phi}_s . \qquad (2.5)
\end{aligned}$$

Therefore, $\mathbf{L}$'s eigen-space of eigenvalue zero can be derived directly from that of $\mathbf{L}_{sym}$, and vice versa. Results similar to Proposition 1 can be shown for $\mathbf{L}_{sym}$.

2.2 Quality Functions

Various approaches have been studied to perform clustering on datasets represented by graphs. A more detailed review can be found in [33] and its references. The main work of this thesis (Chapter 3-5) focuses on the type of methods which is based on a global quality function.

To be more specific, in order to cluster or partition a graph, one needs a quality function to quantify and measure how "good" a clustering is. The quality function models how clusters are defined and what properties of a clustering is favored, which varies depending on different applications at hands. Once a quality function is chosen, the remaining task is to optimize it over all admissive clusterings, either maximization or minimization depending on the specific quality function.

Normally a quality function favors clusters that have tight connections within, but are loosely connected with each other. How to balance these criteria is the core of a quality function. This section introduces several representative quality functions: the family of *graph cuts* and its connection to the *graph total variation*, and the *modularity* function. Note that there are many other quality functions in the literature other than what is covered in this section, but the graph cuts and the modularity function can give a good picture of this type of methods.

2.2.1 Graph Cuts

The most common concept in graph based models is perhaps the *graph cuts*. Consider a partitioning of the graph $G = A_1 \cup A_2 \cup \dots A_{\hat{n}}$ where $\{A_r\}_{r=1}^{\hat{n}}$ are disjoint subsets of G. Let $g : G \to \{1, 2, \dots, \hat{n}\}$ be the group assignment of each node, i.e. a node n_i belongs to A_r if and only if $g_i := g(n_i)$ takes the value r. The graph cuts of this partitioning is defined as the sum of all the edge links one needs

to cut in order to separate $\{A_r\}_{r=1}^{\hat{n}}$ from each other:

$$\mathrm{Cut}(A_1, A_2, \ldots, A_{\hat{n}}) := \sum_{g_i \neq g_j} w_{ij} \,. \tag{2.6}$$

The graph cuts in the case of $\hat{n} = 2$ is illustrated in Figure 2.2 (b), where a red dash line is splitting the graph into two groups of nodes. The corresponding graph cuts for this splitting is the sum of the two edges being cut through by the dash line.

From the definition one can see that the graph cuts is measuring the inter-connections between clusters. Therefore, if one wants to partition a graph into loosely connected subgroups, the corresponding graph cuts is preferred to be small. A general principle of clustering can thus be finding a clustering that minimizes the graph cuts over all admissive ones.

However, if the quality function is set to be the graph cuts only, it tends to yield solutions where most clusters contains only one single node while one cluster contains most of the nodes in the graph. Such solutions are trivial and can not provide useful information. To resolve this issue, people therefore add certain balance terms into the graph cuts so that the sizes of clusters are favored to be similar.

One popular version of modified graph cuts is the *normalized cuts* [81], usually in the two-cluster form (i.e. $\hat{n} = 2$, $G = A_1 \cup A_2$):

$$\mathrm{Ncut}(A_1, A_2) := \frac{\mathrm{Cut}(A_1, A_2)}{\mathrm{vol}(A_1)} + \frac{\mathrm{Cut}(A_1, A_2)}{\mathrm{vol}(A_2)} \,, \tag{2.7}$$

where the quantity $\mathrm{vol}(A)$ denotes the *volume* of a subset $A \in G$ defined as:

$$\mathrm{vol}(A) := \sum_{n_i \in A} d_i \,.$$

In the normalized cuts, the volume terms in the denominators serve the purpose of balancing the cluster sizes. Because it favors equal volumes on A_1 and A_2,

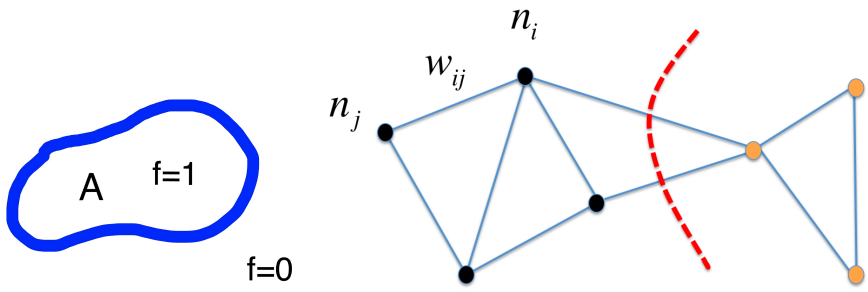
trivial solutions are likely to be avoided. Therefore, using the normalized cuts as the quality function is more well-posed than the graph cuts itself.

Finding the minimizer of the normalized cuts is NP hard [96]. Therefore relaxing the problem and solving it approximately is a common option. Among many approaches explored in the literature, *spectral clustering* [95] is a popular one to solve the relaxed version of the normalized cuts. Analysis in [81] shows that the eigenvectors of the normalized Laplacian $\mathbf{L}_{sym}$ is an approximate solution to the minimization of Ncut. More specifically, to bipartition the graph the normalized spectral clustering first computes the second eigenvector

$$\hat{\phi}_2 = \text{argmin}_{a \perp \mathbf{1}} \frac{\langle a, \mathbf{L}_{sym} a \rangle}{\langle a, a \rangle}, \tag{2.8}$$

and then cluster the nodes into two groups according to the signs of $\hat{\phi}_2$'s value. This method has been widely used due to its efficiency and relatively good performance.

There are other variants of the graph cuts such as the ratio cut [41] and the Cheeger cut [15]; and alternative relaxations are explored [15, 16].



(a) Indicator function (b) Graph cuts

Figure 2.2: (a). An indicator function f of a set A, illustrating that the length of $A's$ perimeter is the total variation of f. (b). Illustration of the graph cuts.

2.2.2 Total Variation

In Euclidean space, the total variation (TV) norm of a real valued function f is defined as the L_1 norm of f's gradient:

$$|f|_{TV} := \int |\nabla f| \,. \tag{2.9}$$

The TV norm has been used as a very useful regularizer in image processing and L_1 compressive sensing, such as in the Rudin-Osher-Fatemi denoising model [80]. An intuitive interpretation of TV is illustrated in Figure 2.2 (a): given a subset A in the problem domain and an indicator function $f = \chi_A$, the length of region A's perimeter equals the TV norm of f:

$$|f|_{TV} = \text{length}(\Gamma_A) \,,$$

where Γ_A denotes the boundary of the subset A. Therefore by adding the total variation term into the energy (quality function) of a minimization problem, one can enforce a more regular boundary in a segmentation, or a smoother intensity variation in an image.

Analogously, people define a *graph total variation* for $f : G \to \mathbb{R}$

$$|f|_{TV} := \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} \left| f(n_i) - f(n_j) \right| \,. \tag{2.10}$$

Interestingly, the total variation on a graph function has very close connection to the graph cuts introduced in the last section. In fact, we have

$$\text{Cut}(A, A^c) = |\chi_A|_{TV} \,.$$

In another words, the graph cuts of a partition can be seen as the boundaries of the subsets.

Comparing the equations (2.1) and (2.10), one can observe that when $f = \chi_A$, the equality holds:

$$\langle f, \mathbf{L}f \rangle = |f|_{TV} \,. \tag{2.11}$$

Therefore, $\langle f, \mathbf{L}f \rangle$ is an L_2 relaxation of $|f|_{TV}$ and hence the graph cuts $\mathrm{Cut}(A, A^c)$. This is an intuitive interpretation of why the spectral clustering method is an approximate solution to the family of graph cuts problems.

2.2.3 Modularity

From the perspective of network science, the clustering problem is posed as "community detection" [76]. Communities are functional units of a network which have close links within but loose connections with the outside. Usually the number of clusters (communities) is not pre-assigned for community detection.

A popular method for community detection is the *modularity* optimization [69,71–73]. The quality function modularity is to be maximized to achieve a good clustering of a network. The modularity of a partition g is defined as

$$Q(g) = \frac{1}{2m} \sum_{i,j=1}^{N} \left(w_{ij} - \gamma \frac{d_i d_j}{2m} \right) \delta(g_i, g_j) \,, \tag{2.12}$$

where $g_i \in \{1, 2, \ldots, \hat{n}\}$ is the community assignment of n_i, and γ is a *resolution parameter* [78]. The term $\delta(g_i, g_j) = 1$ if $g_i = g_j$ and $\delta(g_i, g_j) = 0$ otherwise. The resolution parameter can change the scale at which a network is clustered [33,76]. A network breaks into more communities as one increases γ. The resolution parameter implicitly controls the number of clusters. In modularity optimization, there is no fixed number of clusters being enforced directly in the quality function.

The modularity of a graph partition measures the fraction of total edge weight within each community minus the edge weight that would be expected if edges were placed randomly using some null model [76]. The most common null model is the Newman-Girvan (NG) model [73], which assigns the expected edge weight between n_i and n_j to be $\frac{d_i d_j}{2m}$, recalling that $d_i = \sum_{s=1}^{N} w_{is}$ is the strength (i.e., weighted degree) of a node n_i and the quantity $2m = \sum_{i=1}^{N} d_i$ is the total volume (i.e., total edge weight) of the graph (G, E). An advantage of the NG null model is that it preserves the expected strength distribution of the network. As shown

in the definition (2.12), we adopt the term $\frac{d_i d_j}{2m}$ as the null model in this work. The literature on algorithms of modularity optimization is briefly introduced in Section 4.1.

2.2.4 Number of clusters

The number of clusters is an important factor of clustering algorithms. In some methods such as k-means, there is a pre-assigned number of clusters which serves as an explicit constraint in the quality function. For some other methods such as the modularity, the number of clusters is not specified, but rather induced implicitly by other parameters in the quality function. Sometimes a method can only do a bipartitioning at a time (i.e. partitioning into two subgroups), and people therefore implement it recursively to achieve a desirable number of clusters.

These methods suit different purposes depending on the applications. For tasks such as partitioning computer work loads for parallel computing, the number of clusters is fixed. However, for community detection in social networks, usually there is no "true" number of clusters in the clustering problem. Because it all depends on at what scale one examines the network: coarse or fine, high level structure or low level detail. In network science, it makes more sense to look at a range of scales in order to understand a fuller picture of the data, rather than a single fixed scale or a fixed number of clusters.

CHAPTER 3

Network Analysis using Multi-slice Modularity Optimization

In this chapter, a generalized modularity function is discussed and implemented on both social network and imagery datasets. Diagnostic analysis of the data is performed accordingly to understand the network structure.

3.1 Multi-slice Modularity

As introduced in the previous chapter, the modularity Q defined in the equation (2.12) is a quality function that measures the "goodness" of a clustering. Finding a network partition that attempts to maximize Q allows one to probe a network's community structure. In contrast to traditional forms of spectral clustering, modularity optimization requires no knowledge of the number or sizes of communities, and it also allows one to segment a network into communities of disparate sizes [69, 76].

Optimization of modularity was recently generalized to a "multi-slice" network framework [64]. It consists of layers of ordinary networks, which share the same node-set but may have different edge-set. In another words, for the s-th slice there is a similarity matrix $\mathbf{W}_s$ representing the intra-slice connections between nodes $\{n_{is}\}_{i=1}^{N}$. Additionally there are also inter-slice connections linking the corresponding nodes across slices, such as between nodes n_{ir} and n_{is} from the r-th and the s-th slice respectively. This framework of the multi-slice networks can

thereby be used to represent time-dependent or multi-scale networks.

Under this framework, the generalized modularity function proposed in [64] is defined as:

$$Q_{\text{multi}}(g) = \frac{1}{\mu} \sum_{ijsr} \left[(w_{ijs} - \gamma_s \frac{d_{is} d_{js}}{2m_s}) \delta_{sr} + \delta_{ij} C_{jsr} \right] \delta(g_{is}, g_{jr}) \,, \tag{3.1}$$

where g_{jr} indicates that community assignment of node j from slice r, the intra-slice edge strength of node j in the s-th slice is $d_{js} = \sum_i w_{ijs}$, the corresponding inter-slice edge strength is $c_{js} = \sum_r C_{jsr}$, and $2\mu = \sum_{jr} d_{jr} + c_{jr}$. In (3.1), one can use a different resolution parameter γ_s in each slice. For a given slice s, the quantity w_{ijs} gives the edge weight between nodes n_{is} and n_{js}. For a given node j, the quantity C_{jsr} gives the inter-slice coupling between the rth and sth slices.

In Fig. 3.1, we show a schematic of the multi-slice network.

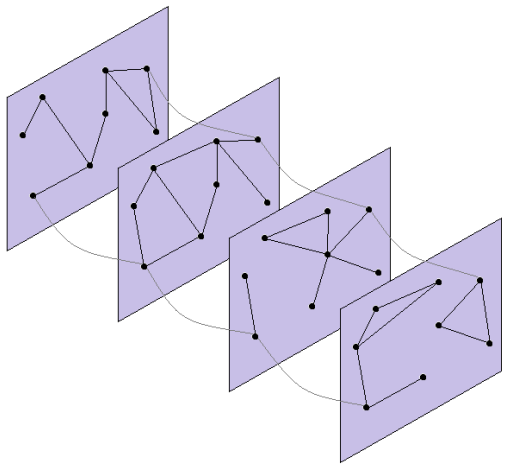


Figure 3.1: Schematic of a multislice network.

Optimization of the ordinary modularity function (2.12) has been used to study community structure in myriad networks [76], and it has also been used in the analysis of hyperspectral images [58] recently. In this work, we optimize multislice modularity (3.1) to examine community structure in social networks and segmentation of images. In each case, we start with a static graph, and each layer of the multi-slice network uses the same adjacency matrix but associates it with

a different resolution-parameter value γ_s. We include inter-slice edges between each node j in adjacent slices only, so $C_{jsr} = 0$ unless $|r - s| = 1$. We set all nonzero inter-slice edges to a constant value ω. This setup, which was illustrated using the infamous Zachary Karate Club network in [64], allows one to detect communities using a range of resolution parameter values while enforcing some consistency in clustering identical nodes similarly across slices. The strength of this enforcement becomes larger as one increases ω. To optimize the multi-slice modularity (3.1), we use a Louvain-like locally-greedy algorithm [9, 50], which is explained in Section 4.1. In our tests, we use the GenLouvain code (in Matlab) from Ref. [50]. The GenLouvain code is a modified implementation of the Louvain locally greedy algorithm [9] so that it applies to the cases with arbitrary values of resolution parameter, but it was not optimized for speed.

3.2 LAPD Field Interview Data

In [91], we use data with both geographic and social information about stops involving street gang members in the Los Angeles Police Department (LAPD) Division of Hollenbeck [92]. We optimize multi-slice modularity (3.1) as a means of unsupervised clustering of individual gang members without prior knowledge of the number of gangs or affiliation of the members. We subsequently examine network diagnostics over slices to attempt to estimate the number of gangs that is stable across multiple resolution-parameter values and that also corresponds roughly to the number expected by the LAPD.

3.2.1 Data description

The numerical experiment is tested on the field interview data from Hollenbeck provided by LAPD. It records the incidents of gang members being stopped while meeting with each other. In total there are 748 individuals involved, who belong

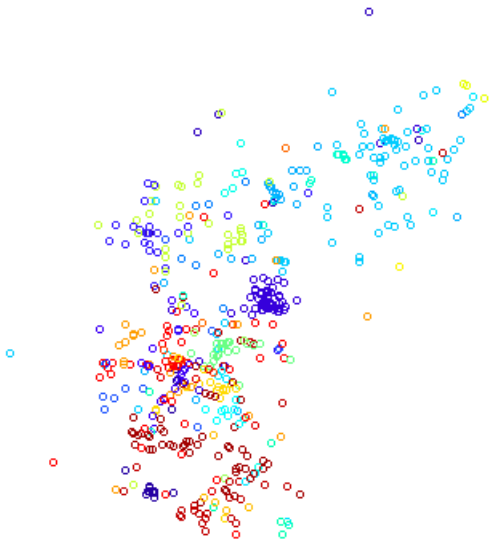


Figure 3.2: Hollenbeck gang member distribution.

to 31 different gangs. As shown in Figure 3.2, the 748 colored dots represent all the interviewed members distributed in Hollenbeck according to their average location across all the recorded events, with the color indicating their affiliation.

Consider a graph with each node representing a member interviewed. We have a social-connection adjacency matrix $\mathbf{S}$ of size 748×748 satisfying $\mathbf{S}[i,j] = 1$ when the members associated with nodes n_i and n_j have been recorded meeting with each other at some point, and $\mathbf{S}[i,j] = 0$ otherwise, ($\mathbf{S}[i,i] = 1$). Additionally there is a geographical matrix $\mathbf{G}$ of the same size satisfying:

$$\mathbf{G}[i,j] = exp(-dist^2(i,j)/\sigma^2)\,,$$

which is derived from the location information of each individual. The quantity $dist(i,j)$ denotes the physical distance between the individual n_i's average location across all the recorded events and that of n_j. The term σ is a normalizing parameter which is hand picked empirically. The gang-affiliation information, i.e. the ground truth (GT), of each gang member is also known. Each individual belongs to a unique gang. The goal is to find the community structures of these 748 people based on the matrixes $\mathbf{S}$ and $\mathbf{G}$, without any prior knowledge about

the their affiliation information or the number of gangs.

3.2.2 Geographical and social matrix

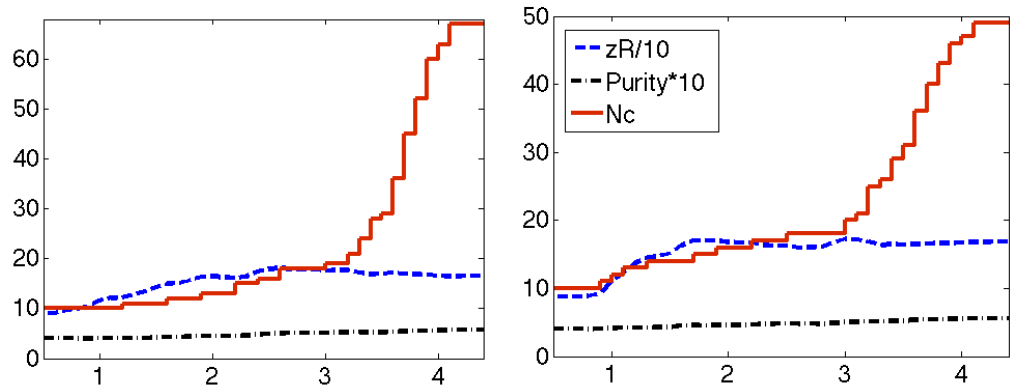
Define a similarity (weight) matrix $\mathbf{W}$ which linearly combines both the social connection $\mathbf{S}$ and the geographical information $\mathbf{G}$:

$$\mathbf{W} := \alpha\mathbf{S} + (1-\alpha)\mathbf{G}\,.$$

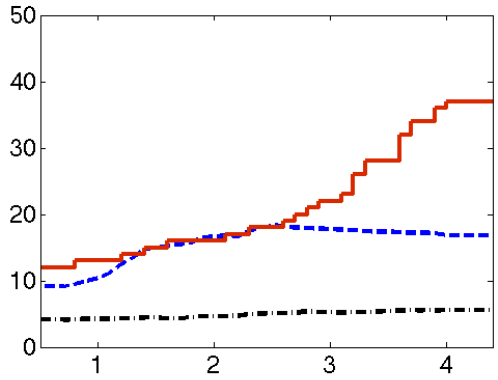
To implement the multi-slice modularity framework, let the subgraph $\mathbf{W}_s$ for the s-th slice to be the same as $\mathbf{W}$ but associated with a different value of resolution parameter $\gamma_s = 0.4 + 0.1s$, s=1,2,...,40. In other words, the multi-slice network we are considering consists of 40 slices, and each one of them is a copy of $\mathbf{W}$ indexed by a different γ_s. The slices are ordered according to the values of γ_s. We connect each node in every slice to the corresponding nodes in neighboring slices, i.e. we set $C_{jsr} \in \{0, \omega\}$ and it equals to ω if and only if s and r are consecutive slice index ($|s-r| = 1$). This way one can detect communities simultaneously over a range of resolution parameters, while still enforcing some consistency in clustering identical nodes similarly across slices.

For a specific fixed α, the resulting partition of the multi-slice network using a Louvain-like algorithm [9, 50] straightforwardly yields partitions on each slice. Each such partition is a clustering on the group of 748 interviewed members. Network diagnostics, such as the number of clusters, *purity*, and *zRand-score*, are examined on these clusterings to understand the network structure by comparing to the ground truth (GT) gang affiliation.

As mentioned before, the modularity optimization determines the number of clusters as part of its output. The quantities purity and zRand-score are used to measure how "similar" one clustering is compared to the ground truth. To compute purity [42], we assign to all nodes in a given community the label of the gang that appears the most often in that group (in case of a tie between two or



(a) α=0 (b) α=0.4



(c) α=0.8

Figure 3.3: Partitioning results using the multi-slice modularity optimization, with $\omega = 1$ and $\mathbf{W} = \alpha\mathbf{S} + (1-\alpha)\mathbf{G}$. The number of clusters (Nc), zRand-score (zR) and purity (Prt) for the partition of each slice are plotted as functions of the resolution parameter γ_s. Different values of α is used: (a) $\alpha = 0$, (b) $\alpha = 0.4$, (c) $\alpha = 0.8$.

more gangs, the label of one of these gangs is arbitrarily chosen for all the nodes in that group). The purity is then given by the fraction of the correctly labeled individuals in the whole data set. The value of purity ranges from zero to one, with one indicating a good match.

The quantity zRand-score is a standardized pair counting method measuring

the similarity of two partitions. To obtain the zRand-score, one computes the number of pairs w of individuals who belong to the same gang and who are placed in the same community by a clustering algorithm. One can then compare this number to its expected value under a hypergeometric distribution with the same number and sizes of communities. The zRand-score, which is normalized by the standard deviation from the mean, indicates how far the actual w value lies in the tail of a distribution [87]. There is no upper bound for the value of zRand-score, and higher value indicates a good clustering.

In Figure 3.3, the number of clusters (Nc), zRand-score (zR) and purity (Prt) for the partition of each slice are plotted as functions of the resolution parameter γ_s. The inter-slice coupling parameter is set to be a constant $\omega = 1$. Small variations in the value of ω did not qualitatively change the result.

A plateau in the number of clusters curve may indicate a relatively stable status of the clustering and hence yields a reasonable number of gangs. Therefore, from Figure 3.3 we seek plateaus in the number of clusters that are near a local maximum of the zRand-score. The details differ slightly for different α, but the general picture that arises is that the optimal number of clusters for our data lies around 18 clusters with a resulting z-Rand score of about 180. Purity is again roughly constant and again near 0.5. Note, however, that comparing the purity scores of two different partitions with different numbers of clusters is not very meaningful, as purity is biased to favor partitions with more clusters.

3.2.3 Ground truth & geographical matrix

From the previous experiment shown in Figure 3.3, we observe that changing the value of $\alpha \in [0, 1]$ does not have a big influence on the resulting purity and zRand-score. This suggests that the social component of the data is too sparse to significantly improve the clustering.

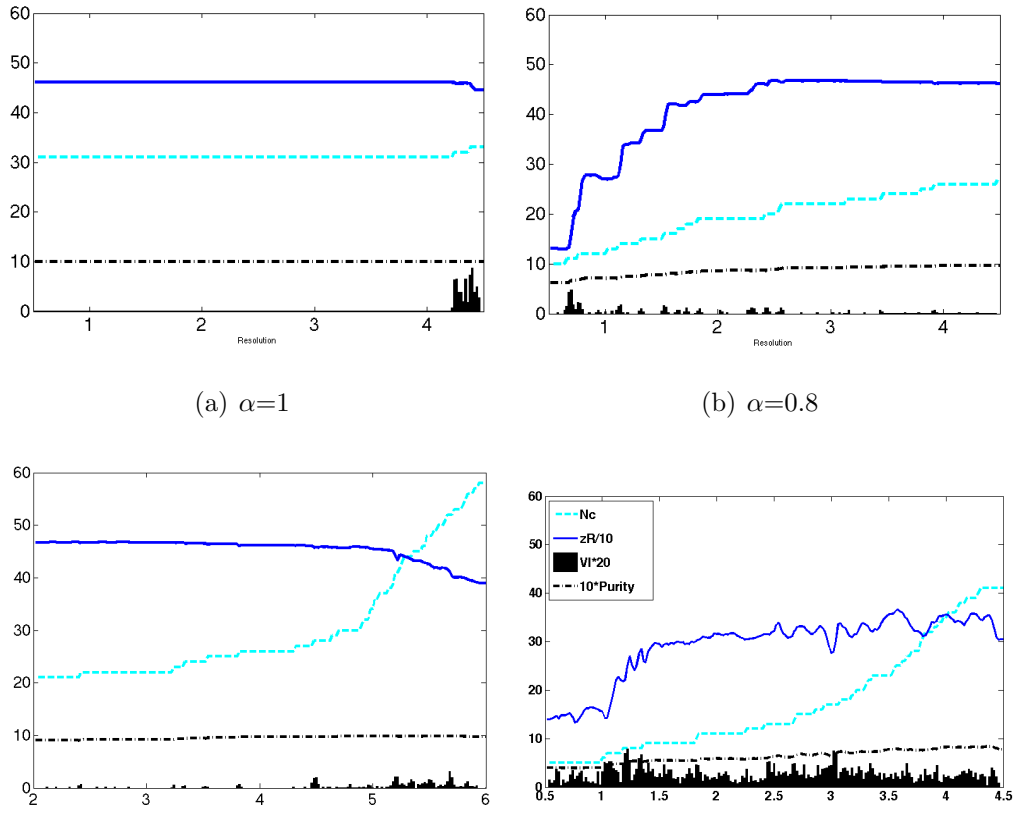
To test whether an improvement might be expected at all if more information on social interactions becomes available in the future, we replace the social matrix $\mathbf{S}$ by the ground truth matrix $\mathbf{T}$ in formulating the similarity matrix:

$$\mathbf{W}_{\mathrm{GT}} = \alpha\mathbf{T} + (1-\alpha)\mathbf{G}\,.$$

The ground true matrix $\mathbf{T}$ is derived from the known affiliation of each interviewed member: $\mathbf{T}[i,j] = 1$ if and only if n_i and n_j belong to the same gang or $i = j$, and $\mathbf{T}[i,j] = 0$ otherwise. Additionally, the nonzero entry of $\mathbf{T}$ is randomly switched to zero with probability 0.1. Therefore we are actually using 90% of the ground truth in the probability sense.

The multi-slice network is subsequently constructed similar to the previous section. It consists of 200 slices, and each of them is a copy of $\mathbf{W}_{\mathrm{GT}}$ associated with a different value of the resolution parameter γ_s, $s = 1, 2, \ldots, 200$. The slices are ordered according to the value of γ_s. In the tests shown in Figure 3.4 (a), (b) and (d), the resolution parameter takes value $\gamma_s = 0.5 + 0.02s$, while in (c) $\gamma_s = 2 + 0.02s$.

In Figure 3.4, the number of clusters (Nc), zRand-score (zR) and purity (Prt) are plotted as functions of the resolution parameter for partitions on each slice. One can observe that when $\alpha = 1$, which means only $\mathbf{T}$ (90% GT) contributes, the Nc perfectly picked up the number 31. As α decreases, more geographical information is involved instead of the ground truth, and there are increasing unstableness in the plots, similar to the real data which is sparse and noisy. For $\alpha = 0.8$, one can see that there is a turning point in the Nc curve around $\gamma_s = 4.9$ which corresponds to Nc=31. The number of clusters increases progressively until $\gamma_s = 4.9$, where it starts increasing dramatically. The zRand-score also starts to decline right after that point. This suggests that the network breaks into small pieces which no longer have significant structure meaning after $\gamma_s = 4.9$. Hence the partition corresponding to this point may give us interesting information.



(a) α=1 (b) α=0.8

(c) α=0.8 (d) α=0.4

Figure 3.4: Partitioning results using the multi-slice modularity optimization, with $\omega = 1$ and $\mathbf{W}_{\mathrm{GT}} = \alpha\mathbf{T} + (1-\alpha)\mathbf{G}$. The number of clusters (Nc), zRand-score (zR) and purity (Prt) for the partition of each slice are plotted as functions of the resolution parameter γ_s. Different values of α is used: (a) $\alpha = 1$, (b) $\alpha = 0.8$, (c) $\alpha = 0.8$, and (d) $\alpha = 0.4$.

The specific partition corresponding to $\alpha = 0.8, \gamma_s = 4.9$ is visualized in Figure 3.5. Each pie corresponds to a gang, and the location of each pie is the true location of that gang (on average). The color indicates the community assignment of the partition. Most of the gangs are successfully picked up by the partition. Therefore, the resolution parameter value around $\gamma_s = 4.9$ yields a good clustering with the number of clusters close to the ground truth. This experiment shows that

more social information indeed helps to improve the performance of the method.

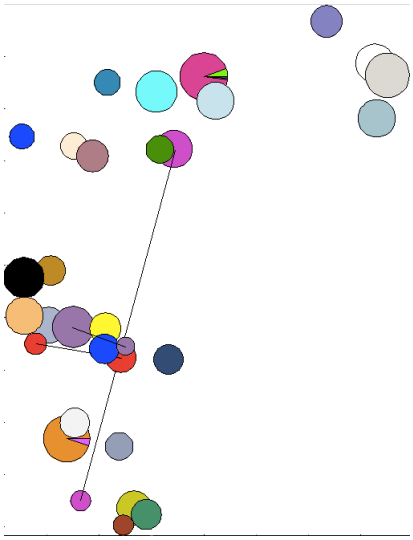


Figure 3.5: Visualization of the partitioncorresponding to $\alpha = 0.8, \gamma_s = 4.9$. Each pie represents one gang, placed according to actual geographical information. The color indicates partition assignments. Lines connect pies of the same color.

3.3 Cow Image

The modularity function is mostly studied in the network science for community detection, rarely in image processing. In the few literatures where it is implemented on imaging, such as in [58], only a fixed scale ($\gamma = 1$) has been briefly studied. In this section, the multi-slice modularity is implemented on the task of unsupervised image segmentation to explore the components of an image at a range of multiple scales. This work is presented in the paper [47].

The experiment is performed on a cow image showed in Figure 3.6, which contains about 3×10^4 pixels. The goal is to segment the image without specifying the number of image components. A graph is built for this image in which each node corresponds to a pixel and each edge indicates the similarity between a pair of pixels. We associate a 3×3 pixel-neighbor patch with each pixel i in the image.

Figure 3.6: Image of a pair of cows, which we downloaded from the Microsoft Research Cambridge Object Recognition Image Database (copyright © 2005 Microsoft Corporation). It is cropped from the original image to produce the segmentation in Figure 3.7.

By stacking together the three channels (RGB) of the pixel-neighbor patch, one obtains a 27-dimensional feature vector denoted by v_i for each pixel i. Let $p_D(i,j)$ denote the L_2 norm of the difference of patches corresponding to the pixels i and j, i.e.:

$$p_D(i,j) := \|v_i - v_j\|_2 \,.$$

The similarity matrix $\mathbf{W}_s$ that is used in each layer of the multi-slice network is set to be the same with elements:

$$\mathbf{W}_s[i,j] = w_{ij} = \exp\left\{\frac{-p_D^2(i,j)}{\tau(i)\tau(j)}\right\}, \tag{3.2}$$

where $\tau(i)$ is the 30th smallest p_D between pixel i and other pixels [99], known as local scaling. By using the pixel-neighbor patch in building the graph, more non-local and texture information is covered.

We construct a multislice network that consists of six copies of $\mathbf{A}$. We associate the resolution parameter value $\gamma_s = 0.04s - 0.03$ with slice $s \in \{1,\ldots,6\}$. We then optimize multislice modularity and obtain the image segmentations shown in Fig. 3.7. (Color indicates group assignments.) With this procedure, we are

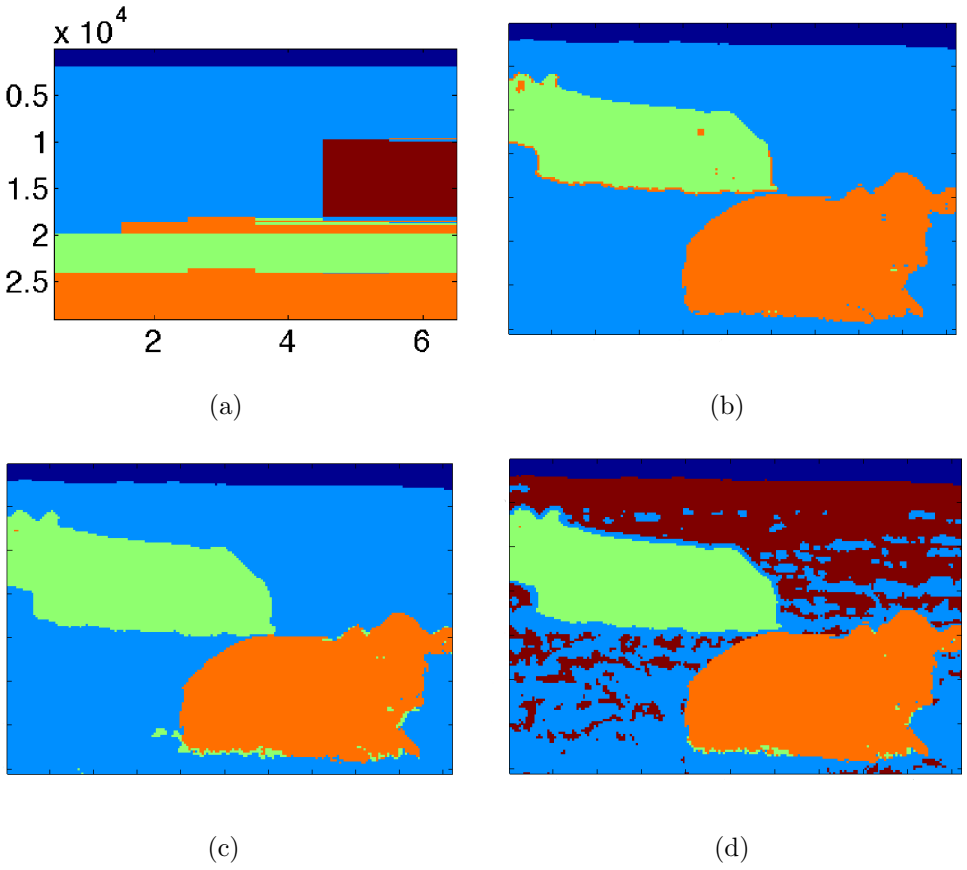


Figure 3.7: (a) Segmentation of the cow image in Fig. 3.6 obtained using optimization of multislice modularity with interslice coupling parameter $\omega = 0.3$. The horizontal axis shows the slice index $s \in \{1, \ldots, 6\}$, which has an associated resolution-parameter value of $\gamma_s = 0.04s - 0.03$. The vertical axis gives the sorted pixel index. Color in each vertical stripe indicates the community assignments of the pixels in the corresponding network slice. We also show the segmentation that we obtain in the images for (b) $\gamma_s = 0.05$, (c) $\gamma_s = 0.13$, and (d) $\gamma_s = 0.21$.

able to identify all four components of the image. As indicated in panel (a), we obtain smaller-scale communities (i.e., groups of pixels) as we increase the value of the resolution parameter. Importantly (see the discussion in Section 3.1), the coupling between slices enforces some consistency in clustering identical nodes similarly across slices. In panels (b) and (c), we observe a good segmentation of the two cows, the sky, and the background grass. As indicated in panel (d), the three groups corresponding to the two cows and the sky stay relatively stable, but the group corresponding to the grass breaks down by the sixth slice. The computational time using the GenLouvain is about 3 hours. This inspired us to develop more efficient and scalable algorithms in Chapter 4 and 5.

When building a graph for an image with each pixel being a node, the size of the graph can easily get very large. This poses challenges upon both computing the similarity matrix and storing it in computer memory. To deal with the memory limits, people usually consider a sparse similarity matrix which has nonzero entries at the order of $O(N)$, instead of a full one which is $O(N^2)$. To make a similarity matrix sparse, one common choice known as the *K-Nearest-Neighbor* (KNN) method is to keep K edges with the largest weight for each node and set the others to zero, where K$<<N$.

To compute a KNN similarity matrix straightforwardly is still a computationally intensive task. Because it needs to compute $O(N^2)$ many pair-wise similarity to find the K largest ones for each node. Thus it does not scale well when N becomes large. For example, it takes 60 hours to straightforwardly build the KNN similarity matrix for the cow image test. To tackle this difficulty, one option is to randomly sample a small subset of all the pixels, and compute the K largest similarity among them instead of the whole set of pixels. In the cow image test, the time cost can be reduced to about 2 hours by such sub-samplying without affecting the performance significantly. Empirically if the randomly sampled subset of pixels can reasonably represent the whole pixel set (in terms of their associ-

ated neighbor patch), then the KNN similarity matrix produced this way would not affect the performance much. A more sophisticated heuristic for finding the KNN pixel-patch by random sampling is proposed in [6], which achieved good performance in industrial application. In Chapter 5, a Nyström method is used to improve the efficiency of constructing large graph.

This application of multi-slice modularity on image segmentation is computationally expensive in general due to the large number of pixels. It takes a lot of computational memory and time to run the optimization using more slices, which we would like to do in order to investigate how the segmentation evolves over a larger range of resolution values. Computational improvements will be necessary to conduct more detailed analysis.

CHAPTER 4

A Variational Method for Modularity Optimization

Modularity optimization is a widely adopted method in network science for the task of community detection. By maximizing modularity, one expects to obtain a reasonable partitioning of a network. Recall that the modularity of a partition g is defined as

$$Q(g) = \frac{1}{2m} \sum_{i,j=1}^{N} \left(w_{ij} - \gamma \frac{d_i d_j}{2m} \right) \delta(g_i, g_j) \,. \tag{4.1}$$

One can refer to Chapter 2 for detailed definitions and an interpretation of the modularity function.

In Chapter 3, we have explored several interesting applications of the multi-slice modularity model. In general, the modularity optimization method has achieved a lot of success in network science. However, this maximization problem is NP hard [12], hence considerable effort has been put into the development of computational heuristics to obtain network partitions with high values of Q. The background for modularity optimization algorithms is briefly introduced in Section 4.1.

However, these approaches are either heuristic and not mathematically well founded (such as the Louvain-like algorithm), or not efficient and accurate enough for large scale datasets (such as the spectral methods). This chapter is aiming at developing an alternative approach to solving the modularity optimization inspired by variational methods from PDE and image processing, such that the

proposed approach is both efficient and mathematically well founded. In Section 4.2 certain variational methods in Euclidean space relating to clustering problems are introduced. Starting from Section 4.3, the modularity optimization problem is posed as an energy minimization problem in the graph setting; then a variational scheme is proposed to solve for the minimization problem using a graph version of the MBO scheme. Numerical experiments are presented after that.

4.1 Background for Modularity Optimization Algorithms

Numerous methods have been proposed [33,76] to approximately solve modularity optimization. These include greedy algorithms [23, 67], extremal optimization [11, 28], simulated annealing [40, 51], spectral methods (which use eigenvectors of a modularity matrix) [68,79], and more. The locally greedy Louvain-like algorithm by Blondel et al. [9] is arguably the most popular computational heuristic; it is a very fast algorithm, and it also yields high modularity values [33, 53].

Section 4.6.3 will discuss more details about the spectral method, where the leading eigenvectors (associated with the largest eigenvalues) of a so-called *modularity matrix* $\mathbf{B} = \mathbf{W} - \mathbf{P}$ are used to approximate the modularity function Q. In the modularity matrix, $\mathbf{P}$ is the probability null model and $P_{ij} = \frac{d_i d_j}{2m}$ is the NG null model with $\gamma = 1$. Ref. [68] gives a very neat mathematical derivation of the spectral method for bipartitioning, while the authors of [79] generalized it to the tripartitioning scenario.

The remaining of this section gives a brief description of the Louvain-like algorithm [9]. Note that the original version of this algorithm only applies to the particular scenario of $\gamma = 1$. Some part of the algorithm efficiency depends on the fact $\gamma = 1$. The GenLouvain code from [50] generalizes it to scenarios with arbitrary values of γ.

The algorithm alternates between the following two phases:

1. Assume we have a weighted graph with N nodes at the current stage. Let each node to be a distinctive cluster by itself. For each node n_i, merge it to its neighbor n_j's cluster that gives the maximum positive increase in modularity. If none of n_i's neighbor nodes can give a positive increase in modularity via this process, then leave n_i's group assignment unchanged. Perform this process for every node in a random order and repeat it until no more change can be made. Note that the order of nodes in this process would affect the outcome, but according to [9] it is empirically insignificant.

2. At this phase a new network is constructed where each new node corresponds to a cluster from the previous phase. The link between a pair of new nodes is the sum of all the edge weights between the two associated clusters. Self-loop may occur in this new network. After this phase, the total number of clusters is reduced.

While iterating between the above two phases, the number of clusters decreases and the modularity score increases. The algorithm terminates when no more change can be made and the modularity achieves a (local) maximum value.

4.2 Variational Method

The optimization problem of a graph quality function concerning clustering is usually NP hard. In order to develop efficient algorithms with reasonable computational costs for the optimization, people have explored various approaches to approximately solve it. To further study the clustering problem, we explore an alternative approach to the optimization problem which is mathematically well founded as well as computationally efficient. In this section, the background of certain variational methods in Euclidean space relating to clustering problems is introduced; and in the following sections, we present a graph version variational method for modularity optimization.

In the analysis of PDE inn Euclidean space, a variational method refers to using calculus of variation techniques to find the minimizer (or maximizer) of a functional (energy) over certain function spaces. Two variational methods that are closely related to the clustering problem studied in this work are introduced: the Ginzburg-Landau functional and the MBO scheme.

4.2.1 Ginzburg-Landau functional

The Ginzburg-Landau functional is a widely used energy functional for diffuse interface models in Euclidean space. In physics, given a region of particles, the Ginzburg-Landau functional models a competition between two phases of the particles. The minimizer of the Ginzburg-Landau functional indicates a stable status of particles' phases. At this status, the region is separated into two subgroups according to the two phases, between which there is a phase transition region (soft boundary). This induced separation is essentially a segmentation of the region and therefore the Ginzburg-Landau functional is related to our clustering problem.

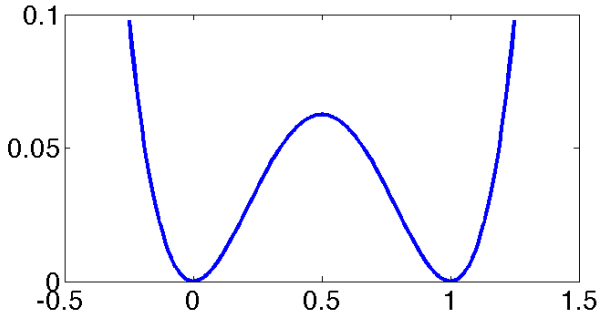


Figure 4.1: Illustration of a double well function $W(f) = f^2(f-1)^2$ with two equilibrium points at zero and one.

The definition of the Ginzburg-Landau functional in Euclidean space is given as:

$$GL(f) = \frac{\epsilon}{2}\int |\nabla f|^2 dx + \frac{1}{4\epsilon}\int W(f)dx\,, \tag{4.2}$$

where $W(f)$ is a double well potential function satisfying $W(f) = f^2(f-1)^2$. The goal is to find a minimizer f^* of $GL(f)$. As illustrated in Figure 4.1, the double well functional $\int W(f)dx$ has two minimizers at $f = 0$ and $f = 1$. Therefore, by minimizing the double well function, the value of f is enforced to move towards either zero or one. Thus a soft phase separation is induced. The first term $\int |\nabla f|^2 dx$ is a regularizer, upon minimizing which the smoothness and regularity of the function f is favored. The small parameter ϵ represents the diffuse interface spacial scale, which relates to how wide the transition region is.

The differentiability of the Ginzburg-Landau functional allows simple algorithms for minimization. To minimizer the energy $GL(f)$, one can use a gradient descent method in the L_2 sense:

$$f_\tau = -\frac{d}{df} GL(f) = \epsilon \Delta f - \frac{1}{\epsilon} W'(f)\,, \tag{4.3}$$

where the quantity τ is the time evolving parameter. Note that the PDE equation (4.3), the L_2 gradient flow of the Ginzburg-Landau energy, is the Allen-Cahn equation. According to [48], the Allen-Cahn equation converges to motion by mean curvature, which is a gradient flow of the total variation semi-norm. The Ginzburg-Landau functional has been used as an alternative for the total variational semi-norm in image processing (see the references in Ref. [8]) due to its Γ-convergence to the TV of characteristic functions in Euclidean space [52]. Because the Ginzburg-Landau functional is not convex, one can only achieve local minima using the gradient descent method.

4.2.2 MBO scheme

An alternative approach to approximating motion by mean curvature of an interface in Euclidean space is introduced by the authors of [62, 63]: an efficient algorithm known as the Merriman-Bence-Osher (MBO) scheme using threshold dynamics. The general procedure of the MBO scheme alternates between solving

a linear heat equation and thresholding. Because motion by mean curvature is a gradient flow of the TV semi-norm, the goal of the MBO scheme is to approximately solve:

$$\frac{\delta f}{\delta \tau} = -\frac{d}{df}|f|_{TV} = -\frac{d}{df}\int |\nabla f| dx \,. \tag{4.4}$$

An L_2 relaxation of the equation (4.4) is:

$$\frac{\delta f}{\delta \tau} = -\frac{d}{df}\int |\nabla f|^2 dx = \Delta f \,, \tag{4.5}$$

which is the heat equation. The equation (4.5) is a tight approximation when f is close to a indicator function, because $|\nabla f| = |\nabla f|^2$ when f only takes value zero or one.

Starting from the indicator function $f = \chi_A$, the MBO scheme first propagates f according to the heat equation (4.5) by a small time step. After that, the function f is no longer a binary indicator function and the L_2 relaxation is no longer tight. Therefore, the MBO scheme subsequently performs a hard threshold on f such that the value of f switches to one if $f > 0.5$, and zero otherwise. Consequently, f becomes an indicator function again, with the corresponding subset modified slightly.

In summary, the MBO scheme alternates between the two steps: propagating f according to the heat equation by a small time step; and thresholding f to become an indicator function. Empirically, the MBO scheme converges very quickly, and the computation cost for each iteration is low because the thresholding step is a simple operation. Mathematical analysis has been explored to understand the theoretical connection between the MBO scheme and motion by mean curvature. It turns out the MBO scheme can be proven to converge to motion by mean curvature, in the sense of Γ-convergence. See references [5, 30, 31] for discussions of the convergence of the MBO scheme.

The MBO scheme has a close connection to minimizing the Ginzburg-Landau functional with a gradient descent (4.3). After all, they are both approximating

motion by mean curvature, i.e. a gradient flow of the total variation semi-norm. One interpretation of the connection is that the MBO scheme replaces the non-linear term of the Allen-Cahn equation with thresholding [29]. The non-linear term $\frac{1}{\epsilon}W'(f)$ in (4.3) is induced by the double well function, which enforces the minimizer f^* to take binary values, and thus poses a soft constraint for f to be close to an indicator function. In the MBO scheme however, instead of using the double well term, the thresholding serves as a hard constraint to enforce f to be an exact indicator function at the end of each iteration.

4.3 Reformulation of Modularity Optimization

In this subsection, we reformulate the problem of modularity optimization by deriving a new expression for Q that bridges the network-science and compressive-sensing communities. This formula makes it possible to use techniques from the latter to tackle the modularity-optimization problem with low computational cost.

We start by recalling the total variation (TV) and defining a weighted L_2-norm, and weighted mean of a graph function $f : G \to \mathbb{R}$:

$$\begin{aligned} |f|_{TV} &:= \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}\,|f(n_i) - f(n_j)|\ , \\ \|f\|_{L_2}^2 &:= \sum_{i=1}^{N} d_i\,|f(n_i)|^2\ , \\ \mathrm{mean}(f) &:= \frac{1}{2m}\sum_{i=1}^{N} d_i f(n_i)\,. \end{aligned} \tag{4.6}$$

For a vector-valued function $f = (f_1, f_2, \ldots, f_{\hat{n}})$: $G \to \mathbb{R}^{\hat{n}}$, we define

$$\begin{aligned} |f|_{TV} &:= \sum_{l=1}^{\hat{n}} |f_l|_{TV}\,, \\ \|f\|_{L_2}^2 &:= \sum_{l=1}^{\hat{n}} \|f_l\|_{L_2}^2\,, \end{aligned} \tag{4.7}$$

and $\text{mean}(f) := (\text{mean}(f_1), \text{mean}(f_2), \ldots, \text{mean}(f_{\hat{n}}))$.

Given a partition $g = \{g_i\}_{i=1}^{N}$ defined in Section 2.2.1, let $A_l = \{n_i \in G, g_i = l\}$, where $l \in \{1, 2, \ldots, \hat{n}\}$ ($\hat{n} \leq N$). Thus, $G = \cup_{l=1}^{\hat{n}} A_l$ is a partition of the network (G, E) into disjoint communities. Note that every A_l is allowed to be empty, so g is a partition into at most $\hat{n}$ communities. Let $f_l : G \to \{0, 1\}$ be the indicator function χ_{A_l} of community l; in other words, $f_l(n_i)$ equals one if $g_i = l$, and it equals zero otherwise. The function $f = (f_1, f_2, \ldots, f_{\hat{n}})$ is then called the *partition function* (associated with g). Because each set A_l is disjoint from all of the others, it is guaranteed that only a single entry of f_i equals one for any node n_i. Therefore, $f : G \to V^{\hat{n}} \subset \mathbb{R}^{\hat{n}}$, where

$$V^{\hat{n}} := \{(1, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)\} = \{\vec{e}_l\}_{l=1}^{\hat{n}}$$

is the standard basis of $\mathbb{R}^{\hat{n}}$. It is equivalent to $f \in \mathbb{B}$.

The key observation that bridges the network-science and compressive-sensing communities is the following:

Theorem 1. *Maximizing the modularity functional Q over all partitions that have at most $\hat{n}$ communities is equivalent to minimizing*

$$|f|_{TV} - \gamma \|f - \text{mean}(f)\|_{L_2}^2 \tag{4.8}$$

over all functions $f : G \to V^{\hat{n}}$.

Proof. In the language of graph partitioning, $\text{vol}(A_l) = \sum_{n_i \in A_l} d_i$ denotes the volume of the set A_l, and $\text{Cut}(A_l, A_l^c) = \sum_{n_i \in A_l, n_j \in A_l^c} w_{ij}$ is the graph cut of A_l

and A_l^c. Therefore,

$$\begin{aligned} Q(g) &= \frac{1}{2m}\left[\left(2m - \sum_{g_i \neq g_j} w_{ij}\right) - \frac{\gamma}{2m}\sum_{l=1}^{\hat{n}}\left(\sum_{n_i \in A_l, n_j \in A_l} d_i d_j\right)\right] \\ &= 1 - \frac{1}{2m}\left(\sum_{l=1}^{\hat{n}} \mathrm{Cut}(A_l, A_l^c) + \frac{\gamma}{2m}\sum_{l=1}^{\hat{n}} \mathrm{vol}(A_l)^2\right) \\ &= 1 - \gamma - \frac{1}{2m}\left(\sum_{l=1}^{\hat{n}} \mathrm{Cut}(A_l, A_l^c) - \frac{\gamma}{2m}\Big(\sum_{l=1}^{\hat{n}} \mathrm{vol}(A_l)\cdot \mathrm{vol}(A_l^c)\Big)\right), \end{aligned} \tag{4.9}$$

where the sum $\sum_{g_i \neq g_j} w_{ij}$ includes both w_{ij} and w_{ji}. As mentioned in Section 2.2.2, one has $|\chi_A|_{TV} = \mathrm{Cut}\,(A, A^c)$. Additionally, one can observe that:

$$\begin{aligned} \|\chi_A - \mathrm{mean}(\chi_A)\|_{L_2}^2 &= \sum_{i=1}^{N} d_i \left|\chi_A(n_i) - \frac{\mathrm{vol}(A)}{2m}\right|^2 \\ &= \mathrm{vol}(A)\left(1 - \frac{\mathrm{vol}(A)}{2m}\right)^2 + \mathrm{vol}\,(A^c)\left(\frac{\mathrm{vol}\,(A)}{2m}\right)^2 \\ &= \frac{\mathrm{vol}(A)\cdot \mathrm{vol}\,(A^c)}{2m}. \end{aligned} \tag{4.10}$$

Because $f^{(l)} = \chi_{A_l}$ is the indicator function of A_l, it follows that:

$$\begin{aligned} |f|_{TV} - \gamma\|f - \mathrm{mean}(f)\|_{L_2}^2 &= \sum_{l=1}^{\hat{n}} \left\{|f_l|_{TV} - \gamma\|f_l - \mathrm{mean}(f_l)\|_{L_2}^2\right\} \\ &= \sum_{l=1}^{\hat{n}} \left\{\mathrm{Cut}(A_l, A_l^c) - \gamma\frac{\mathrm{vol}(A_l)\cdot \mathrm{vol}(A_l^c)}{2m}\right\}. \end{aligned} \tag{4.11}$$

Combining (4.9) and (4.11), we conclude that maximizing Q is equivalent to minimizing (4.8). □

With the above argument, we have reformulated the problem of modularity maximization as the minimization problem (4.8), which corresponds to minimizing the total variation (TV) of the function f along with a balance term. This yields a novel view of modularity optimization that uses the perspective of compressive sensing (see the references in [57]). In the context of compressive sensing, one seeks a solution of function f that is compressible under the transform of a linear

operator Φ. That is, we want Φf to be well-approximated by sparse functions. (A function is considered to be "sparse" when it is equal to or approximately equal to zero on a "large" portion of the whole domain.) Minimizing $\|\Phi f\|_{L_1}$ promotes sparsity in Φf. When Φ is the gradient operator (on a continuous domain) or the finite-differencing operator (on a discrete domain) ∇, then the object $\|\Phi f\|_{L_1} = \|\nabla f\|_{L_1}$ becomes the total variation $|f|_{TV}$ [57,66]. The minimization of TV is also common in image processing and computer vision [21, 57, 66, 80].

The expression in equation (4.9) is interesting because its geometric interpretation of modularity optimization contrasts with existing interpretations (e.g., probabilistic ones or in terms of the Potts model from statistical physics [68, 76]). For example, we see from (4.9) that finding the bipartition of the graph $G = A \cup A^c$ with maximal modularity is equivalent to minimizing

$$\mathrm{Cut}(A, A^c) - \frac{\gamma}{2m}\mathrm{vol}(A) \cdot \mathrm{vol}\,(A^c)\ .$$

Note that the term $\mathrm{vol}(A) \cdot \mathrm{vol}\,(A^c)$ is maximal when $\mathrm{vol}(A) = \mathrm{vol}\,(A^c) = m$. Therefore, the second term is a balance term that favors a partition of the graph into two groups of roughly equal size. In contrast, the first term favors a partition of the graph in which few links are severed. This is reminiscent of the normalized cuts problem (Section 2.2.1) in which the objective is to minimize the ratio

$$\frac{\mathrm{Cut}\,(A, A^c)}{\mathrm{vol}(A) \cdot \mathrm{vol}\,(A^c)}\,. \tag{4.12}$$

In recent papers Refs. [14, 16, 17, 43, 44, 77], various TV-based algorithms are proposed to minimize ratios similar to (4.12).

4.4 Algorithm

Directly optimizing (4.8) over all partition functions $f : G \to V^{\hat{n}}$ is difficult due to the discrete solution space. A continuous relaxation is thus needed to simplify the optimization problem.

4.4.1 Ginzburg-Landau relaxation of the discrete problem

Recall that

$$\mathbb{B} = \{f \mid f : G \to V^{\hat{n}}\}$$

denotes the space of partition functions. Minimizing (4.8) over $\mathbb{B}$ is equivalent to minimizing

$$H(f) = \begin{cases} |f|_{TV} - \gamma \|f - \text{mean}(f)\|_{L_2}^2 , & \text{if } f \in \mathbb{B} \\ +\infty , & \text{otherwise} \end{cases} \tag{4.13}$$

over all $f : G \to \mathbb{R}^{\hat{n}}$.

As discussed in Section 4.2.1, the Ginzburg-Landau (GL) functional has been used as an alternative for the TV term in image processing. Reference [8] developed a graph version of the GL functional and used it for graph-based high-dimensional data segmentation problems. The authors of reference [35, 59] generalized the two-phase graphical GL functional to a multi-phase one.

Following the idea in references [8,35,59], we define the graph *Ginzburg-Landau relaxation* of H as follows:

$$H_\epsilon(f) = \frac{1}{2} \sum_{l=1}^{\hat{n}} \langle f_l, \mathbf{L} f_l \rangle + \frac{1}{\epsilon^2} \sum_{i=1}^{N} W_{\text{multi}} \left(f(n_i) \right) - \gamma \|f - \text{mean}(f)\|_{L_2}^2 , \tag{4.14}$$

where $\epsilon > 0$. In equation (4.14), $W_{\text{multi}} : \mathbb{R}^{\hat{n}} \to \mathbb{R}$ is a multi-well potential (see [35, 59]) with equal-depth wells. The minima of W_{multi} are spaced equidistantly, take the value zero, and correspond to the points of $V^{\hat{n}}$. The specific formula for W_{multi} does not matter for this work, because we will discard it when we implement the MBO scheme. Note that the purpose of this multi-well term is to force $f(n_i)$ to go to one of the minima, so that one obtains an approximate phase separation.

Our next theorem states that modularity optimization with an upper bound on the number of communities is well-approximated (in terms of Γ-convergence) by minimizing H_ϵ over all $f : G \to \mathbb{R}^{\hat{n}}$. Therefore, the *discrete* modularity optimization problem (4.8) can be approximated by a *continuous* optimization problem.

We give the mathematical definition and relevant proofs of Γ-convergence in Section 4.5.

Theorem 2 (Γ–convergence of H_ϵ towards H)**.** *The functional H_ϵ Γ-converges to H on the space $X = \{f \mid f : G \to \mathbb{R}^{\hat{n}}\}$.*

Proof. As shown in Theorem 3 (in Section 4.5), $H_\epsilon + \gamma\|f - \text{mean}(f)\|_{L_2}^2$ Γ-converges to $H + \gamma\|f - \text{mean}(f)\|_{L_2}^2$ on X. Because $\gamma\|f - \text{mean}(f)\|_{L_2}^2$ is continuous on the metric space X, it is straightforward to check that H_ϵ Γ-converges to H according to the definition of Γ-convergence. □

By the definition of Γ-convergence, Theorem 2 directly implies the following:

Corollary 1. *Let f^ϵ be the global minimizer of H_ϵ. Any convergent subsequence of f^ϵ then converges to a global maximizer of the modularity Q with at most $\hat{n}$ communities.*

4.4.2 MBO scheme, convex splitting, and spectral approximation

In this subsection, we use techniques from the compressive-sensing and image-processing literatures to develop an efficient algorithm that (approximately) optimizes H_ϵ.

In Section 4.2.2, we discussed a fast variational method called the MBO scheme. Inspired by the MBO scheme, the authors of reference [29] developed a method using a PDE framework to minimize the piecewise-constant Mumford-Shah functional (introduced in [65]) for image segmentation. Their algorithm was motivated by the Chan-Vese level-set method [21] for minimizing certain variants of the Mumford-Shah functional. Note that the Chan-Vese method is related to our reformulation of modularity, because it uses the TV as a regularizer along with L_2 based fitting terms. The authors of references [35, 59, 60] applied the MBO scheme to graph-based problems.

The gradient-descent equation of (4.14) is

$$\frac{\partial f}{\partial t} = -(\mathbf{L}f_1, \ldots, \mathbf{L}f_{\hat{n}}) - \frac{1}{\epsilon^2}\frac{d}{df}W_{\text{multi}}(f) + \frac{d}{df}\left(\gamma\|f - \text{mean}(f)\|_{L_2}^2\right), \quad (4.15)$$

where $\frac{d}{df}W_{\text{multi}}(f) : G \to \mathbb{R}^{\hat{n}}$ is the composition of the functions $\frac{d}{df}W_{\text{multi}}$ and f. Thus, one can follow the idea of the original MBO scheme to split (4.15) into two parts and replace the forcing part $\frac{\partial f}{\partial t} = -\frac{1}{\epsilon^2}\frac{d}{df}W_{\text{multi}}(f)$ by an associated thresholding.

We propose a *Modularity MBO scheme* that alternates between the following two primary steps to obtain an approximate solution $f^n : G \to V^{\hat{n}}$:

Step 1.

A gradient-descent process of temporal evolution consists of a diffusion term and an additional balance term:

$$\frac{\partial f}{\partial t} = -(\mathbf{L}f_1, \ldots, \mathbf{L}f_{\hat{n}}) + \frac{\delta}{\delta f}\left(\gamma\|f - \text{mean}(f)\|_{L_2}^2\right). \quad (4.16)$$

We apply this process on f^n with time step τ_n, and we repeat it for η time steps to obtain $\hat{f}$.

Step 2.

We threshold $\hat{f}$ from $R^{\hat{n}}$ into $V^{\hat{n}}$:

$$f^{n+1}(n_i) = \vec{e}_{g_i} \in V^{\hat{n}}, \text{ where } g_i = \text{argmax}_{\{1\le l\le \hat{n}\}}\{\hat{f}_l(n_i)\}.$$

This step assigns to $f^{n+1}(n_i)$ the node in $V^{\hat{n}}$ that is the closest to $\hat{f}(n_i)$.

To solve (4.16), we implement a *convex-splitting* scheme [32, 94]. Equation (4.16) is the gradient flow of the energy $H_1 + H_2$, where $H_1(f) := \frac{1}{2}\sum_{l=1}^{\hat{n}}\langle f_l, \mathbf{L}f_l\rangle$ is convex and $H_2(f) := -\gamma\|f - \text{mean}(f)\|_{L_2}^2$ is concave. In a discrete-time stepping scheme, the convex part is treated implicitly in the numerical scheme, whereas the concave part is treated explicitly. Note that the convex-splitting scheme for gradient-descent equations is an unconditionally stable time-stepping scheme.

The discretized time-stepping formula is

$$\begin{aligned}\frac{\hat{f}-f^n}{\tau_n} &= -\frac{d}{df}H_1(\hat{f}) - \frac{d}{df}H_2(f^n) \\ &= -(\mathbf{L}\hat{f}_1,\ldots,\mathbf{L}\hat{f}_{\hat{n}}) + 2\gamma\vec{d}\odot\left(f^n - \text{mean}(f^n)\right), \end{aligned} \tag{4.17}$$

where $(\vec{d}\odot f)(n_i) := d_i f(n_i)$, $\hat{f}: G \to \mathbb{R}^{\hat{n}}$, ($d_i$ is the strength of node n_i), and $f^n : G \to V^{\hat{n}}$. At each step, we thus need to solve

$$\left((1+\tau_n\mathbf{L})\hat{f}_1,\ldots,(1+\tau_n\mathbf{L})\hat{f}_{\hat{n}}\right) = f^n + 2\gamma\tau_n\vec{d}\odot\left[f^n - \text{mean}(f^n)\right]. \tag{4.18}$$

For the purpose of computational efficiency, we utilize the low-order (leading) eigenvectors (associated with the smallest eigenvalues) of the graph Laplacian $\mathbf{L}$ to approximate the operator $\mathbf{L}$. The eigenvectors with higher order are more oscillatory, and resolve finer scale. Leading eigenvectors provide a set of basis to approximately represent graph functions. The more leading eigenvectors are used, the finer scales can be resolved. In the graph-clustering literature, scholars usually use a small portion of leading eigenvectors of $\mathbf{L}$ to find useful structural information in a graph [7, 22, 24, 68, 81], (note however that some recent work has explored the use of other eigenvectors [25]). In contrast, one typically uses many more modes when solving partial differential equations numerically (e.g., consider a psuedospectral scheme), because one needs to resolve the solution at much finer scales.

Motivated by the known utility and many successes of using leading eigenvectors (and discarding higher-order eigenvectors) in studying graph structure, we project f onto the space of the N_{eig} leading eigenvectors to approximately solve (4.18). Assume that $f^n = \sum_s \phi_s \mathbf{a}_s^n$, $\hat{f} = \sum_s \phi_s \hat{\mathbf{a}}_s$, and $2\gamma\tau_n\vec{d}\odot(f^n - \text{mean}(f^n)) = \sum_s \phi_s \mathbf{b}_s^n$, where $\{\lambda_s\}$ are the N_{eig} smallest eigenvalues of the graph Laplacian $\mathbf{L}$. We denote the corresponding eigenvectors (eigenfunctions) by $\{\phi_s\}$. Note that $\mathbf{a}_s^n$, $\hat{\mathbf{a}}_s$, and $\mathbf{b}_s^n$ all belong to $\mathbb{R}^{\hat{n}}$. With this representation, we obtain

$$\hat{\mathbf{a}}_s = \frac{\mathbf{a}_s^n + \mathbf{b}_s^n}{1+\tau_n\lambda_s}, \quad s \in \{1,2,\ldots,N_{\text{eig}}\} \tag{4.19}$$

from (4.18) and are able to solve (4.18) more efficiently.

Algorithm 1 The Modularity MBO scheme.

Set values for γ, $\hat{n}$, η, and $\tau_n = dt$.

Input: $\leftarrow$ an initial function $f^0 : G \to V^{\hat{n}}$ and the eigenvalue-eigenvector pairs $\{(\lambda_s, \phi_s)\}$ of the graph Laplacian $\mathbf{L}$ corresponding to the N_{eig} smallest eigenvalues.

Initialization: for $n = 0$

$\mathbf{a}_s^0 = \langle f^0, \phi_s \rangle$;

while $f^n \neq f^{n-1}$ and $n \leq 500$: **do**

Diffusion:

for $i = 1 \to \eta$ **do**

$\mathbf{b}_s^n = \langle 2\gamma dt \vec{d} \odot (f^n - \text{mean}(f^n)), \phi_s \rangle$;

$\mathbf{a}_s^n \leftarrow \frac{\mathbf{a}_s^n + \mathbf{b}_s^n}{1 + dt\lambda_s}$, for $s \in \{1, 2, \ldots, N_{\text{eig}}\}$;

$f^n \leftarrow \sum_s \phi_s \mathbf{a}_s^n$;

i=i+1;

end for

Thresholding:

$$f^{n+1}(n_i) = \vec{e}_{g_i} \in V^{\hat{n}}, \text{ where } g_i = \text{argmax}_{\{1 \leq l \leq \hat{n}\}} \{\hat{f}_l(n_i)\}.$$

$\mathbf{a}_s^{n+1} = \langle f^{n+1}, \phi_s \rangle$;

$n = n + 1$;

end while

Output $\leftarrow$ the partition function f^n.

We summarize our Modularity MBO scheme in Algorithm 1. Note that the time complexity of each MBO iteration step is $O(N)$. The choice of the initial function f^0 can have significant impact on the outcome of the algorithm, because the modularity function is not convex and different initiation may lead to different local minima. Unless specified otherwise, the numerical experiments in this work

using a random initial function f^0.

4.4.3 Two implementations of the Modularity MBO scheme

Given an input value of the parameter $\hat{n}$, the Modularity MBO scheme partitions a graph into at most $\hat{n}$ communities. In many applications, however, the number of communities is usually not known in advance [33, 76], so it can be difficult to decide what values of $\hat{n}$ to use. Accordingly, we propose two implementations of the Modularity MBO scheme. The *Recursive Modularity MBO (RMM)* scheme is particularly suitable for networks that one expects a large number of communities, whereas the *Multiple Input-$\hat{n}$ Modularity MBO (Multi-$\hat{n}$ MM)* scheme is particularly suitable for networks that one expects to have a small number of communities.

Implementation 1. The RMM scheme performs the Modularity MBO scheme recursively, which is particular suitable for networks that one expects to have a large number of communities. In practice, we set the value of $\hat{n}$ to be large in the first round of applying the scheme, and we then let it be small for the rest of the recursion steps. In the experiments that we report in this work, we use $\hat{n} = 50$ for the first round and $\hat{n} = \min(10, |S|)$ thereafter, where $|S|$ is the size of the subnetwork that one is partitioning in a given step. (We also tried $\hat{n} = 10$, 20 or 30 for the first round and $\hat{n} = \min(10, |S|)$ thereafter. The results are similar.)

Importantly, the minimization problem (4.8) needs a slight adjustment for the recursion steps. Assume for a particular recursion step that we perform the Modularity MBO partitioning with parameter $\hat{n}$ on a network $S \subset G$ containing a subset of the nodes of the original graph. Our goal is to increase the modularity for the global network instead of the subnetwork S. Hence, the target energy to minimize is

$$H^{(S)}(f) := |f|_{TV}^{(S)} - \gamma \frac{m^{(S)}}{m} \left\| f - \text{mean}^{(S)}(f) \right\|_{L_2}^2 ,$$

where $f : S \to V^{\hat{n}} \subset \mathbb{R}^{\hat{n}}$, the TV norm $|\cdot|_{TV}^{(S)}$ is defined as:

$$|f|_{TV}^{(S)} := \frac{1}{2} \sum_{n_i, n_j \in S} w_{ij} |f(n_i) - f(n_j)|_{L_1} ;$$

the total edge weight of S is $2m^{(S)} = \sum_{n_i \in S} d_i$, and

$$\text{mean}^{(S)}(f) = \frac{1}{2m^{(S)}} \sum_{n_i \in S} d_i f(n_i) .$$

The rest of the minimization procedures are the same as described previously.

Note that this recursive scheme is adaptive in resolving the network structure scale. The eigenvectors of the subgroups are recalculated at each recursive step, so the scales being resolved get finer as the recursion step goes. Therefore N_{eig} need not to be very large.

Implementation 2. For the Multi-$\hat{n}$ MM scheme, one sets a search range T for $\hat{n}$, runs the Modularity MBO scheme for each $\hat{n} \in T$, and then chooses the resulting partition with the highest modularity score. It works well if one knows the approximate maximum number of communities and that number is reasonably small. One can then set the search range T to be all integers between two and the maximum number. Even though the Multi-$\hat{n}$ MM scheme allows partitions with fewer than $\hat{n}$ clusters, it is still necessary to include small values of $\hat{n}$ in the search range to better avoid local minimums. (See the discussion of the MNIST "4-9" digits network in Section 4.6.2.1.) For different values of $\hat{n}$, one can reuse the previously computed eigenvectors because $\hat{n}$ does not affect the graph Laplacian. Inputting multiple choices for the random initial function f^0 (as described at the end of Section 4.4) also helps to reduce the chance of getting stuck in a minimum and thereby to achieve a good optimal solution for the Modularity MBO scheme. Because this initial function is used after the computation of eigenvectors, it only takes a small amount of time to rerun the MBO steps. In Section 4.6, we test these two schemes on several real and synthetic networks.

4.5 Γ-convergence

The notion of Γ-convergence of functionals is now commonly used for minimization problems. See reference [26] for detailed introduction. In this section, we briefly review the definition of Γ-convergence and then prove the claim that the graphical multi-phase Ginzburg-Landau functional Γ-converges to the graph TV. This proof is a straightforward extension of the work in [89] for the two-phase graph GL functional.

Definition 1. *Let X be a metric space and let $\{F_n : X \to \mathbb{R} \cup \{\pm\infty\}\}_{n=1}^{\infty}$ be a sequence of functionals. The sequence F_n Γ-converges to the functional $F : X \to R \cup \{\pm\infty\}$ if, for all $f \in X$, the following lower and upper bound conditions hold:*

(lower bound condition) *for every sequence $\{f_n\}_{n=1}^{\infty}$ such that $f_n \to f$, we have*

$$F(f) \leq \liminf_{n\to\infty} F_n(f_n)\,;$$

(upper bound condition) *there exists a sequence $\{f_n\}_{n=1}^{\infty}$ such that*

$$F(f) \geq \limsup_{n\to\infty} F_n(f_n)\,.$$

Reference [35, 59] proposed the following multi-phase graph GL functional:

$$GL_{\epsilon}^{\text{multi}}(\hat{f}) = \frac{1}{2}\sum_{l=1}^{\hat{n}} \langle \hat{f}_l, \mathbf{L}\hat{f}_l\rangle + \frac{1}{\epsilon^2}\sum_{i=1}^{N} W_{\text{multi}}(\hat{f}(n_i))$$

where $\hat{f} : G \to \mathbb{R}^{\hat{n}}$ and $W_{\text{multi}}(\hat{f}(n_i)) = \prod_{l=1}^{\hat{n}} \|\hat{f}(n_i) - \vec{e}_l\|_{L_1}^2$.

Let $X = \{\hat{f} \mid \hat{f} : G \to \mathbb{R}^{\hat{n}}\}$ and $F_\epsilon = GL_{\epsilon}^{\text{multi}}$ for all $\epsilon > 0$, (we have $\mathbb{B} = \{f \mid f : G \to V^{\hat{n}}\} \subset X$). Because $\hat{f}$ can be viewed as a matrix in $\mathbb{R}^{N\times\hat{n}}$, the metric for space X can be defined naturally using the L_2 norm.

Theorem 3. *(Γ-convergence). The sequence F_ϵ Γ-converges to F_0 as $\epsilon \to 0^+$, where*

$$F_0(\hat{f}) := \begin{cases} |\hat{f}|_{TV} = \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}|\hat{f}(n_i) - \hat{f}(n_j)|_{L_1}\,, & \text{if } \hat{f} \in \mathbb{B}\,, \\ +\infty\,, & \text{otherwise}\,. \end{cases}$$

Proof. Consider the functional $W_\epsilon(f) = \frac{1}{\epsilon^2}\sum_{i=1}^{N} W_{\text{multi}}(f(n_i))$ and

$$W_0(f) := \begin{cases} 0\,, & \text{if } f \in \mathbb{B}\,, \\ +\infty\,, & \text{otherwise}\,. \end{cases}$$

First, we show that W_ϵ Γ-converges to W_0 as $\epsilon \to 0^+$. Let $\{\epsilon_n\}_{n=1}^{\infty} \subset (0,\infty)$ be a sequence such that $\epsilon_n \to 0$ as $n \to \infty$. For the lower bound condition, suppose that a sequence $\{f^n\}_{n=1}^{\infty}$ satisfies $f^n \to f$ as $n \to \infty$. If $f \in \mathbb{B}$, then it follows that $W_0(f) = 0 \leq \liminf_{n\to\infty} W_{\epsilon_n}(f^n)$ because $W_\epsilon \geq 0$. If f does not belong to $\mathbb{B}$, then there exists $i \in \{1,2,\ldots,N\}$ such that $f(n_i) \notin V^{\hat{n}}$ and $f^n(n_i) \to f(n_i)$. Therefore, $\liminf_{n\to\infty} W_{\epsilon_n}(f^n) = +\infty \geq W_0(f) = +\infty$. For the upper bound condition, assume that $f \in \mathbb{B}$ and $f^n = f$ for all n. It then follows that $W_0(f) = 0 \geq \limsup_{n\to\infty} W_{\epsilon_n}(f^n) = 0$. Thus, W_ϵ Γ-converges to W_0.

Because $Z(f) := \frac{1}{2}\sum_{l=1}^{\hat{n}} \langle f_l, \mathbf{L} f_l\rangle$ is continuous on the metric space X, it is straightforward to check that the functional $F_{\epsilon_n} = Z + W_{\epsilon_n}$ satisfies the lower and upper bound condition and therefore Γ-converges to $Z + W_0$.

Finally, note that $Z(f) = |f|_{TV}$ for all $f \in \mathbb{B}$. Therefore, $Z + W_0 = F_0$ and one can conclude that F_{ϵ_n} Γ-converges to F_0 for any sequence $\epsilon_n \to 0^+$. □

Theorem 4. *(Compactness). Let $\{\epsilon_n\}_{n=1}^{\infty} \subset R_+$ be a sequence such that $\epsilon_n \to 0$ as $n \to \infty$. $\{\hat{f}^n\}_{n=1}^{\infty} \subset X$ is a sequence such that $F_{\epsilon_n}(\hat{f}^n)$ is uniformly bounded. Then there exists a subsequence $\{\hat{f}^{n'}\}_{n'=1}^{\infty} \subset \{\hat{f}^n\}_{n=1}^{\infty}$ and $f^\infty \in \mathbb{B}$ such that $\hat{f}^{n'} \to f^\infty$ as $n \to \infty$.*

Proof. Since $F_{\epsilon_n}(\hat{f}^n)$ is uniformly bounded, we have

$$\sum_{i=1}^{N} W_{multi}(\hat{f}^n(n_i)) \leq C\epsilon_n, \ \forall \ n;$$

for some constant C. Hence for large enough n, $\{\hat{f}^n\}_{n=1}^{\infty}$ is bounded and therefore has a convergent subsequence $\{\hat{f}^{n'}\}_{n'=1}^{\infty}$ with limit f^{∞}.

W_{multi} is continuous $\Rightarrow \sum_{i=1}^{N} W_{multi}(\hat{f}^{n'}(n_i))$ converges to $\sum_{i=1}^{N} W_{multi}(f^{\infty}(n_i))$ $\Rightarrow \sum_{i=1}^{N} W_{multi}(f^{\infty}(n_i)) = 0$. Therefore $f^{\infty} \in \mathbb{B}$. □

Since $\{f \in X : F_{\epsilon}(f) \leq C\}$ is bounded for any fixed ϵ and C, Theorem 4 proves the equi-coerciveness of the sequence F_{ϵ_n}. Combined with the Γ-convergence, one can conclude the following theorem stated in Chapter 7 of [26].

Theorem 5. *Let X be a metric space and $\{F_n : X \to R \cup \{\pm\infty\}\}_{n=1}^{\infty}$ be a sequence of equi-coerciveness functionals. F_n Γ-converges to $F : X \to R \cup \{\pm\infty\}$. Then there exists a minimizer of F in X and $min\{F(f) : f \in X\} = \liminf_{n\to\infty}\{F_n(f) : f \in X\}$. Furthermore, if $\{\hat{f}^n\}_{n=1}^{\infty} \subset X$ is a precompact sequence such that*

$$\lim_{n\to\infty} F_n(\hat{f}^n) = \liminf_{n\to\infty}\{F_n(f) : f \in X\},$$

then every cluster point of this sequence is a minimizer of F.

4.6 Numerical Results

In this section, we present the numerical results of experiments that we conducted using both synthetic and real network data sets. Unless otherwise specified, our Modularity MBO schemes are all implemented in Matlab, (which are not optimized for speed). In the following tests, we set the parameters of the Modularity MBO scheme to be $\eta = 5$ and $\tau_n = 1$.

4.6.1 LFR benchmark

In Ref. [54], Lancichinetti, Fortunato, and Radicchi (LFR) introduced an eponymous class of synthetic benchmark graphs to provide tougher tests of community-detection algorithms than previous synthetic benchmarks. Many real networks have heterogeneous distributions of node degree and community size, so the LFR benchmark graphs incorporate such heterogeneity. They consist of unweighted networks with a predefined set of non-overlapping communities. As described in Ref. [54], each node is assigned a degree from a power-law distribution with power ξ; additionally, the maximum degree is given by $d_{\max}$ and mean degree is $\langle d \rangle$. Community sizes in LFR graphs follow a power-law distribution with power β, subject to the constraint that the sum of the community sizes must equal the number of nodes N in the network. Each node shares a fraction $1-\mu$ of its edges with nodes in its own community and a fraction μ of its edges with nodes in other communities. (The quantity μ is called the *mixing parameter*.) The minimum and maximum community sizes, $q_{\min}$ and $q_{\max}$, are also specified. We label the LFR benchmark data sets by $(N, \langle d \rangle, d_{\max}, \xi, \beta, \mu, q_{\min}, q_{\max})$. The code used to generate the LFR data is publicly available provided by the authors in [54].

The LFR benchmark graphs has become a popular choice for testing community detection-algorithms, and Ref. [53] uses them to test the performance of several community-detection algorithms. The authors concluded, for example, that the locally greedy Louvain algorithm [9] is one of the best heuristics for maximizing modularity based on the evaluation of the *normalized mutual information* (NMI) (discussed below in this section). Note that the time complexity of this Louvain algorithm is $O(M)$ [33], where M is the number of nonzero edges in the network. In our tests, we use the GenLouvain code (in Matlab) from Ref. [50]; this is an implementation of a Louvain-like algorithm. The GenLouvain code is a modification of the Louvain locally greedy algorithm [9] (introduced in Section 4.1) so that it applies to the cases with arbitrary values of resolution parameter,

but it was not optimized for speed. We implement our RMM scheme on the LFR benchmark, and we compare our results against the GenLouvain code. We use the recursive version of the Modularity MBO scheme because our LFR networks contain about $0.04N$ communities.

We implement the modularity-optimization algorithms on severals sets of LFR benchmark data. We then compare the resulting partitions with the known community assignments of the benchmarks (i.e., the ground truth) by examining the *normalized mutual information* (NMI) [27]. NMI is a similarity measure for comparing two partitions based on the information entropy, and it is often used for testing community-detection algorithms [53, 54]. The NMI equals one when two partitions are identical, and it has an expected value of zero when they are independent. For an N-node network with two partitions, $C = \{C_1, C_2, \ldots, C_K\}$ and $\hat{C} = \{\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_{\hat{K}}\}$, that consist of non-overlapping communities, the NMI is

$$\mathrm{NMI}(C, \hat{C}) = \frac{2\sum_{k=1}^{K}\sum_{\hat{k}=1}^{\hat{K}} P(k,\hat{k})\log\left[\frac{P(k,\hat{k})}{P(k)P(\hat{k})}\right]}{-\sum_{k=1}^{K} P(k)\log\left[P(k)\right] - \sum_{\hat{k}=1}^{\hat{K}} P(\hat{k})\log\left[P(\hat{k})\right]}\,, \tag{4.20}$$

where $P(k,\hat{k}) = \frac{|C_k \cap \hat{C}_{\hat{k}}|}{N}$, $P(k) = \frac{|C_k|}{N}$, and $P(\hat{k}) = \frac{|\hat{C}_{\hat{k}}|}{N}$.

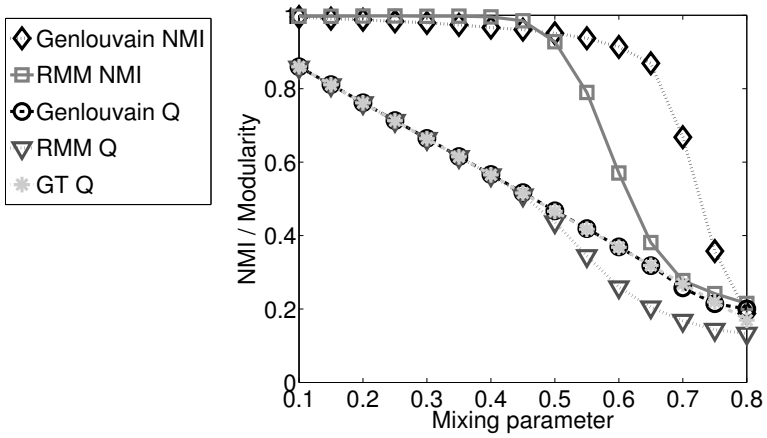
We examine two types of LFR networks. One is the 1000-node ensembles used in Ref. [53]:

$$\mathrm{LFR1k} : (1000, 20, 50, 2, 1, \mu, 10, 50)\,,$$

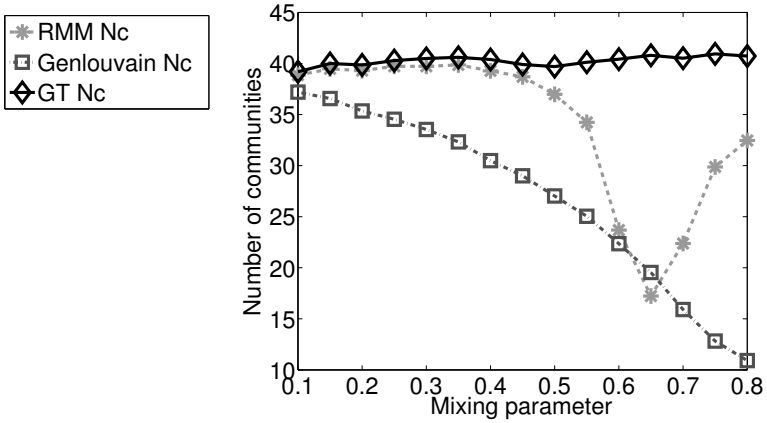
where $\mu \in \{0.1, 0.15, \ldots, 0.8\}$. The other is a 50,000-node network, which we call "LFR50k" and construct as a composition of 50 LFR1k networks. (See the detailed description below.)

4.6.1.1 LFR1k networks

We use the RMM scheme (with $N_{\mathrm{eig}} = 80$) and the GenLouvain code on ensembles of LFR1k$(1000, 20, 50, 2, 1, \mu, 10, 50)$ graphs with mixing parameters $\mu \in \{0.1, 0.15, \ldots, 0.8\}$. We consider 100 LFR1k networks for each value of μ, and we



(a) NMI and Modularity (Q).



(b) Number of Communities (N_c).

Figure 4.2: Tests on LFR1k networks with RMM and GenLouvain. The ground-truth communities are denoted by GT.

use a resolution parameter of $\gamma = 1$.

In Figure 4.2, we plot the mean maximized modularity score (Q), the number of communities (N_c), and the NMI of the partitions compared with the ground truth (GT) communities as a function of the mixing parameter μ. As one can see from panel (a), the RMM scheme performs very well for $\mu < 0.5$. Both its NMI score and modularity score are competitive with the results of GenLouvain. However, for $\mu \geq 0.5$, its performance drops with respect to both NMI and the modularity

scores of its network partitions. From panel (b), we see that RMM tends to give partitions with more communities than GenLouvain, and this provides a better match to the ground truth. However, it is only trustworthy for $\mu < 0.5$, when its NMI score is very close to one.

The mean computational time for one ensemble of LFR1k, which includes 15 networks corresponding to 15 values of μ, is 22.7 seconds for the GenLouvain code and 17.9 seconds for the RMM scheme. As we will see later when we consider large networks, the Modularity MBO scheme scales very well in terms of its computational time.

4.6.1.2 LFR50k networks

To examine the performance of our scheme on larger networks, we construct synthetic networks (LFR50k) with 50,000 nodes. To construct an LFR50k network, we start with 50 different LFR1k networks $N_1, N_2, \ldots, N_{50}$ with mixing parameter μ, and we connect each node in N_s ($s \in \{1, 2, \ldots, 50\}$) to 20μ nodes in N_{s+1} uniformly at random (where we note that $N_{51} = N_1$). We thereby obtain an LFR50k network of size $50,000$. Each community in the original $N_s, s = 1, 2, \ldots, 50$ is a new community in the LFR50k network. We build four such LFR50k networks for each value of $\mu = 0.1, 0.15, \ldots, 0.8$, and we find that all such networks contain about 2000 communities. The mixing parameter of the LFR50k network constructed from LFR1k(μ) is approximately $\frac{2\mu}{1+\mu}$.

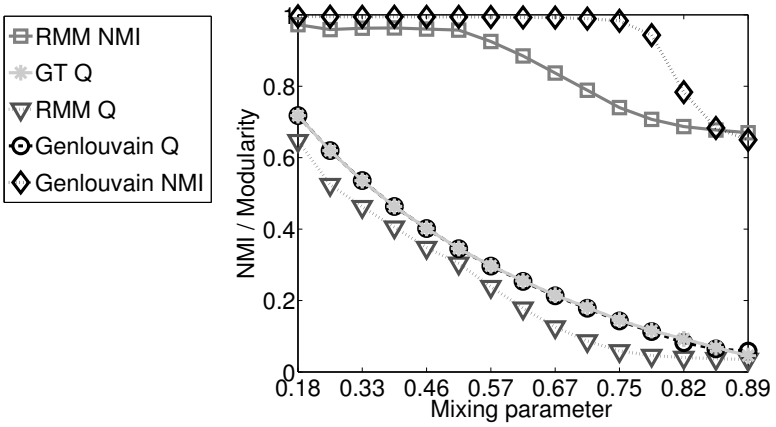
By construction, the LFR50k network has a similar structure as LFR1k. Importantly, simply increasing N in LFR($N, \langle k \rangle, k_{\max}, \xi, \beta, \mu, q_{\min}, q_{\max}$) to 50,000 is insufficient to preserve similarity of the network structure. A large N results in more communities, so if the mixing parameter μ is held constant, then the edges of each node that are connected to nodes outside of its community will be distributed more sparsely. In other words, the mixing parameter does not entirely

reflect the balance between a node's connection within its own community versus its connections to other communities, as there is also a dependence on the total number of communities.

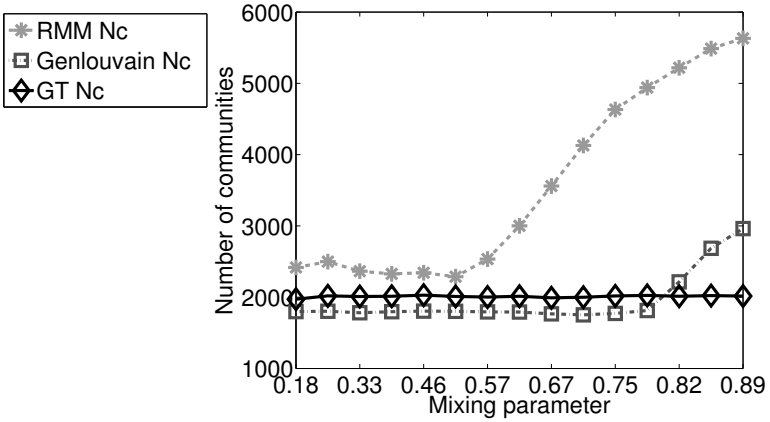
The distribution of node strengths in LFR50k is scaled approximately by a factor of $(1+2\mu)$ compared to LFR1k, while the total number of edges in LFR50k is scaled approximately by a factor of $50(1+2\mu)$. Therefore, the probability null model term $\frac{d_i d_j}{2m}$ in modularity (2.12) is also scaled by a factor of $\frac{(1+2\mu)}{50}$. Hence, in order to probe LFR50k with a resolution scale similar to that in LFR1k, it is reasonable to use the resolution $\gamma = 50$ to try to minimize issues with modularity's resolution limit [78]. We then implement the RMM scheme ($N_{\text{eig}} = 100$) and the GenLouvain code. Note that we also implemented the RMM scheme with $N_{\text{eig}} = 500$, but there is no obvious improvement in the result even though there are about 2000 communities. This is because the eigenvectors of the subgroups are recalculated at each recursive step, so the scales being resolved get finer as the recursion step goes.

We average the network diagnostics over the four LFR50k networks for each value of mixing parameter. In Fig. 4.3, we plot the network diagnostics versus the mixing parameter $\frac{2\mu}{1+\mu}$ for $\mu \in \{0.1, 0.15, \ldots, 0.8\}$. In panel (a), we see that the performance of RMM is good only when the mixing parameter is less than 0.5, though it is not as good as GenLouvain. It seems that the recursive Modularity MBO scheme has some difficulties in dealing with networks with very large number of clusters.

However the computational time of RMM is lower than that of the GenLouvain code [50]. The mean computational time for an ensemble of LFR50k networks, which includes 15 networks corresponding to 15 values of μ, is 690 seconds for GenLouvain and 220 seconds for the RMM scheme. In Table 4.1, we summarize the mean computational time (in seconds) on each ensemble of LFR data.



(a) NMI and Modularity (Q).



(b) Number of Communities (N_c).

Figure 4.3: Tests on LFR50k data with RMM and GenLouvain.

	LFR1k	LFR50k
GenLouvain	22.7 s	690 s
RMM	17.9 s	220 s

Table 4.1: Computational time on LFR data.

4.6.2 MNIST handwritten digit images

The MNIST database consists of 70,000 images of size 28×28 pixels containing the handwritten digits "0" through "9" [1]. The digits in the images have been

normalized with respect to size and centered in a fixed-size grey image. In this section, we use two networks from this database. We construct one network using all samples of the digits "4" and digit "9", which are difficult to distinguish from each other and which constitute 13782 images of the 70000. We construct the second network using all images. In each case, our goal is to separate the distinct digits into distinct communities.

We construct the adjacency matrices (and hence the graphs) $\mathbf{W}$ of these two data sets as follows. First, we project each image (a 28^2-dimensional datum) onto 50 principal components. For each pair of nodes n_i and n_j in the 50-dimensional space, we then let $w_{ij} = \exp\left(-\frac{d_{ij}^2}{3\sigma_i^2}\right)$ if n_j is among the 10 nearest neighbors of n_i; otherwise, we let $w_{ij} = 0$. To make $\mathbf{W}$ a symmetric matrix, we take $\mathbf{W} = \frac{\mathbf{W}+\mathbf{W}'}{2}$. The quantity d_{ij} is the L_2 distance between n_i and n_j, the parameter σ_i is the mean of distances between n_i and its 10 nearest neighbors.

In this data set, the maximum number of communities is two when considering only the digits "4" and "9", and it is 10 when considering all digits. We can thus choose a small search range for $\hat{n}$ and use the Multi-$\hat{n}$ Modularity MBO scheme.

4.6.2.1 MNIST "4-9" digits network

This weighted network has 13782 nodes and 194816 weighted edges. We use the labeling of each digit image as the ground truth. There are two groups of nodes: ones containing the digit "4" and ones containing the digit "9". We use these two digits because they tend to look very similar when they are written by hand. In Figure 4.4 (a), we show a visualization of this network, where we have projected the data projected onto the second and third leading eigenvectors of the graph Laplacian $\mathbf{L}$. The difficulty of separating the "4" and "9" digits has been observed in the graph-partitioning literature (see, e.g., Ref. [44]). For example, there is a near-optimal partition of this network using traditional spectral clustering [81, 95]

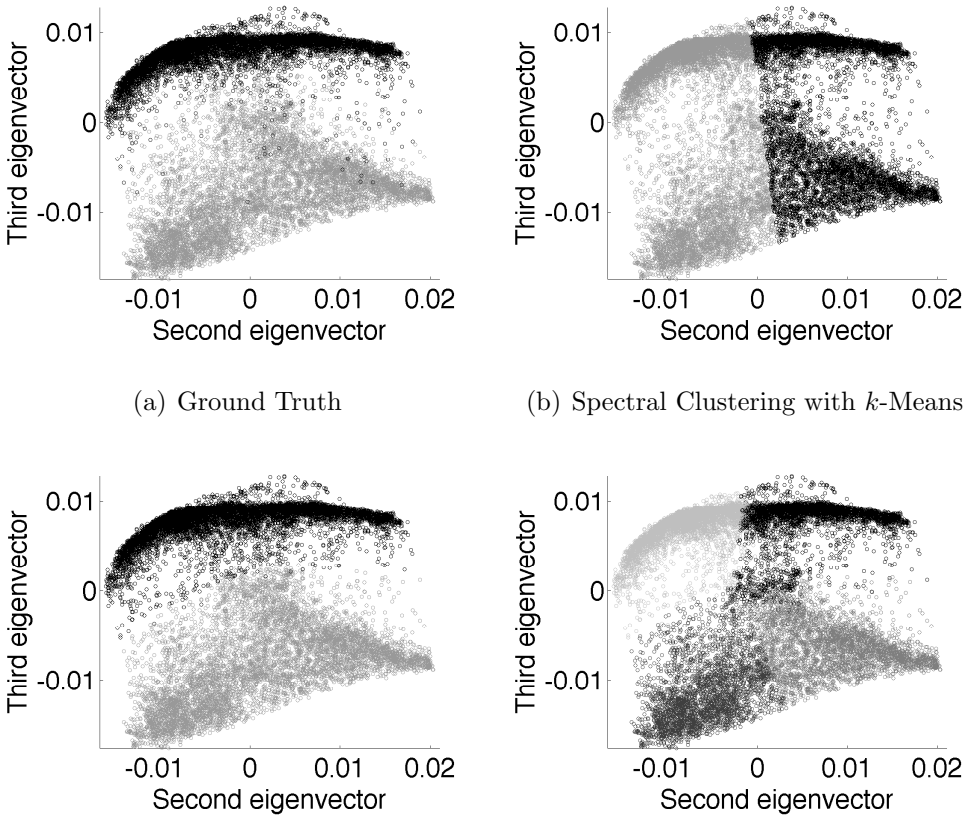
(see below) that splits both the "4"-group and the "9"-group roughly in half.

The modularity-optimization algorithms that we discuss for the "4-9" network use $\gamma = 0.1$. We choose this resolution-parameter value so that the network is partitioned into two groups by the GenLouvain code. The question about what value of γ to choose is beyond the scope of this thesis, but it has been discussed at some length in the literature on modularity optimization [33]. Instead, we focus on evaluating the performance of our algorithm with the given value of the resolution parameter. We implement the Modularity MBO scheme with $\hat{n} = 2$ and the Multi-$\hat{n}$ MM scheme, and we compare our results with that of the GenLouvain code as well as traditional spectral clustering method [81, 95].

Traditional spectral clustering is an efficient clustering method that has been used widely in computer science and applied mathematics because of its simplicity. It calculates the first k nontrivial eigenvectors $\phi_1, \phi_2, \ldots, \phi_k$ (corresponding to the smallest eigenvalues) of the graph Laplacian $\mathbf{L}$. Let $U \in \mathbb{R}^{N \times k}$ be the matrix containing the vectors $\phi_1, \phi_2, \ldots, \phi_k$ as columns. For $i \in \{1, 2, \ldots, N\}$, let $y_i \in \mathbb{R}^k$ be the ith row vector of U. Spectral clustering then applies the k-means algorithm to the points $(y_i)_{\{i=1,\ldots,N\}}$ and partitions them into k groups, where k is the number of clusters that was specified beforehand.

On this MNIST "4-9" digits network, we specify $k = 2$ and implement spectral clustering to obtain a partition into two communities. As we show in Figure 4.4 (b), we obtain a near-optimal solution that splits both the "4"-group and the "9"-group roughly in half. This differs markedly from the ground-truth partition in panel (a).

For the Multi-$\hat{n}$ MM scheme, we use $N_{\text{eig}} = 80$ and the search range $\hat{n} \in \{2, 3, \ldots, 10\}$. We show visualizations of the partition at $\hat{n} = 2$ and $\hat{n} = 8$ in Figure 4.4(c,d). For this method, computing the spectrum of the graph Laplacian takes a significant portion of the run time (9 seconds for this data set). Importantly, however, this information can be reused for multiple $\hat{n}$, which saves time. In



(a) Ground Truth

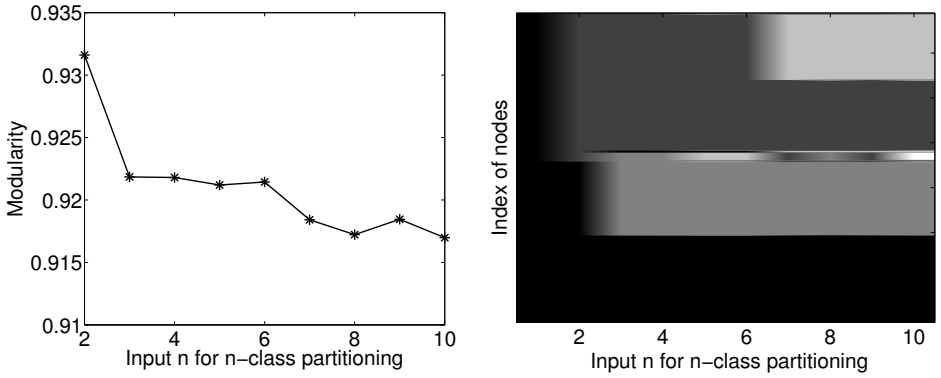
(b) Spectral Clustering with k-Means

(c) Modularity MBO Scheme with $\hat{n}$=2

(d) Modularity MBO Scheme with $\hat{n} = 8$

Figure 4.4: (a)–(d) Visualization of partitions on the MNIST "4-9" digit image network by projecting it onto the second and third leading eigenvectors of the graph Laplacian. Shading indicates the community assignment.

Figure 4.5 (a), we show a plot of this method's optimized modularity scores versus $\hat{n}$. Observe that the optimized modularity score achieves its maximum when we choose $\hat{n} = 2$, which yields the best partition that we obtain using this method. In Figure 4.5(b), we show how the partition evolves as we increase the input $\hat{n}$ from 2 to 10. At $\hat{n} = 2$, the network is partitioned into two groups (which agrees very well with the ground truth). For $\hat{n} > 2$, however, the algorithm starts to pick out worse local optima, and either "4"-group or the "9"-group gets split roughly in half. Starting from $\hat{n} = 7$, the number of communities stabilizes at about four



(a) Modularity Score (b) Group Assignments Visualization

Figure 4.5: Implementation results of the Multi-$\hat{n}$ Modularity MBO scheme on the MNIST "4-9" digit images. In panel (b), shading indicates the community assignment. The horizontal axis represents the input $\hat{n}$ (i.e., the maximum number of communities), and the vertical axis gives the (sorted) index of nodes. In panel (a), we plot the optimized modularity score as a function of the input $\hat{n}$.

instead of increasing with $\hat{n}$. This indicates that the Modularity MBO scheme allows one to obtain partitions with $N_c \leq \hat{n}$.

	N_c	Q	NMI	Purity	Time (seconds)
GenLouvain	2	0.9305	0.85	0.975	110 s
Modularity MBO ($\hat{n} = 2$)	2	0.9316	0.85	0.977	11 s
Multi-$\hat{n}$ MM ($\hat{n} \in \{2, 3, \ldots, 10\}$)	2	0.9316	0.85	0.977	25 s
Spectral Clustering (k-Means)	2	NA	0.003	0.534	1.5 s

Table 4.2: Computational time and accuracy comparison on MNIST "4-9" digits network.

The *purity* score, which we also report in Table 4.2, measures the extent to which a network partition matches ground truth. Suppose that an N-node network has a partition $C = \{C_1, C_2, \ldots, C_K\}$ into non-overlapping communities and that the ground-truth partition is $\hat{C} = \{\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_{\hat{K}}\}$. The purity of the

partition C is then defined as

$$\text{Prt}(C, \hat{C}) = \frac{1}{N} \sum_{k=1}^{K} \max_{l \in \{1,\ldots,\hat{K}\}} |C_k \cap \hat{C}_l| \in [0, 1] \,. \tag{4.21}$$

Intuitively, purity can by viewed as the fraction of nodes that have been assigned to the correct community. However, the purity score is not robust in estimating the performance of a partition. When the partition C breaks the network into communities that consist of single nodes, then the purity score achieves a value of one. hence, one needs to consider other diagnostics when interpreting the purity score. In this particular data set, a high purity score does indicate good performance because the ground truth and the partitions each consist of two communities.

Observe in Table 4.2 that all modularity-based algorithms identified the correct community assignments for more than 97% of the nodes, whereas standard spectral clustering was only correct for just over half of the nodes. The Multi-$\hat{n}$ MM scheme takes only 25 seconds. If one specifies $\hat{n} = 2$, then the Modularity MBO scheme only takes 11 seconds.

4.6.2.2 MNIST 70k network

We test our new schemes further by consider the entire MNIST network of 70,000 samples containing digits from "0" to "9". This network contains about five times as many nodes as the MNIST "4-9" network. However, the node strengths in the two networks are very similar because of how we construct the weighted adjacency matrix. We thus choose $\gamma = 0.5$ so that the modularity optimization is performed at a similar resolution scale in both networks. There are 1001664 weighted edges in this network.

We implement the Multi-$\hat{n}$ MM scheme with $N_{\text{eig}} = 100$ and the search range $\hat{n} \in \{2, 3, \ldots, 20\}$. Even if N_c is the number of communities in the true optimal solution, the input $\hat{n} = N_c$ might not give a partition with N_c groups. The

modularity landscape in real networks is notorious for containing a huge number of nearly degenerate local optima (especially for values of modularity Q near the globally optimum value) [38], so we expect the algorithm to yield a local minimum solution (such as the merging of groups) rather than a global minimum. Consequently, it is preferable to extend the search range to $\hat{n} > N_c$, so that the larger $\hat{n}$ gives more flexibility to the algorithm to try to find the partition that optimizes modularity.

The best partition that we obtained using the search range $\hat{n} \in \{2, 3, \ldots, 20\}$ contains 11 communities. All of the digit groups in the ground truth except for the "1"-group are correctly matched to those communities. In the partition, the "1"-group splits into two parts, which is unsurprising given the structure of the data. In particular, the samples of the digit "1" include numerous examples that are written like a "7". This set of samples are thus easily disconnected from the rest of "1"-group. If one considers these two parts as one community associated with "1"-group, then the partition achieves a 96% correctness in its classification of the digits. The 4% error rate distributes over each digit group almost evenly.

As we illustrate in Table 4.3, the GenLouvain code yields comparably successful partitions as those that we obtained using the Multi-$\hat{n}$ MM scheme. By comparing the running time of the Multi-$\hat{n}$ MM scheme on both MNIST networks, one can see that our algorithm scales well in terms of speed when the network size increases. While the network size increases five times ($5\times$) and the search range gets doubled ($2\times$), the computational time increases by a factor of $11.6 \approx 5 \times 2$.

The number of iterations for the Modularity MBO scheme ranges approximately from 35 to 100 for $\hat{n} \in \{2, 3, \ldots, 20\}$. Empirically, even though the total number of iterations can be as large as over a hundred, the modularity score quickly gets very close to its final value within the first 20 iteration.

The computational cost of the Multi-$\hat{n}$ MM scheme consists of two parts: the calculation of the eigenvectors and the MBO iteration steps. Because of the size

of the MNIST 70k network, the first part costs about 90 seconds in Matlab. However, one can incorporate a faster eigenvector solver, such as the Rayleigh-Chebyshev (RC) procedure of [3], to improve the computation speed of an eigen-decomposition. This solver is especially fast for producing a small portion (in this case, 1/700) of the leading eigenvectors for a sparse symmetric matrix. Upon implementing the RC procedure in C++ code, it only takes 12 seconds to compute the 100 leading eigenvector-eigenvalue pairs. Once the eigenvectors are calculated, they can be reused in the MBO steps for multiple values of $\hat{n}$ and different initial functions f^0. This allows good scalability, which is a particularly nice feature of using this MBO scheme.

	Nc	Q	NMI	Purity	Time (second)
GenLouvain	11	0.93	0.916	0.97	10900 s
Multi-$\hat{n}$ MM ($\hat{n} \in \{2, 3, \ldots, 20\}$)	11	0.93	0.893	0.96	290 s / 212 s*
Modularity MBO 3% GT ($\hat{n} = 10$)	10	0.92	0.95	0.96	94.5 s / 16.5 s*

**Calculated with the RC procedure.*

Table 4.3: Computational time and accuracy comparison on MNIST 70k network.

Another benefit of the Modularity MBO scheme is that it allows the possibility of incorporating a small portion of the ground truth in the modularity optimization process. In this work, we implement the Modularity MBO using 3% of the ground truth by specifying the true community assignments of 2100 nodes in the initial function f^0; we chose the nodes uniformly at random. We also let $\hat{n} = 10$. With the eigenvectors already computed (which took 12 seconds using the RC process), the MBO steps take a subsequent 4.5 seconds to yield a partition with exactly 10 communities and 96.4% of the nodes classified into the correct groups. The authors of Ref. [35, 59] also implemented a segmentation algorithm on this MNIST 70k data with 3% of the ground truth, and they obtained a partition with a correctness 96.9% in 15.4 seconds. In their algorithm, the ground truth

was enforced by adding a quadratic fidelity term to the energy functional (semi-supervised). The fidelity term is the L_2 distance of the unknown function f and the given ground truth. In our scheme, however, it is only used in the initial function f^0. Nevertheless, it is also possible to add a fidelity term to the Modularity MBO scheme and thereby perform semi-supervised clustering.

4.6.3 Network-science coauthorships

Another well-known graph in the community detection literature is the network of coauthorships of network scientists. This benchmark was compiled by Mark Newman and first used in [68].

In this work, we use the graph's largest connected component, which consists of 379 nodes representing authors and 914 weighted edges that indicate coauthored papers. We do not have any so-called ground truth for this network, but it is useful to compare partitions obtained from our algorithm with those obtained using more established algorithms. In this section, we use GenLouvain's result as this pseudo-ground truth. In addition to Modularity-MBO, RMM, and GenLouvain, we also consider the results of modularity-based spectral partitioning methods that allow the option of either bipartitioning or tripartitioning at each recursive stage [68,79].

In Ref. [68], Newman proposes a spectral partitioning scheme for modularity optimization by using the leading eigenvectors (associated with the largest eigenvalues) of a so-called *modularity matrix* $\mathbf{B} = \mathbf{W} - \mathbf{P}$ to approximate the modularity function Q. In the modularity matrix, $\mathbf{P}$ is the probability null model and $P_{ij} = \frac{d_i d_j}{2m}$ is the NG null model with $\gamma = 1$. Assume that one uses the first p leading eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p\}$, and let β_j denote the eigenvalue of $\mathbf{u}_j$ and $\mathbf{U} = (\mathbf{u}_1 | \mathbf{u}_2 | \ldots | \mathbf{u}_p)$. We then define N node vectors $\mathbf{r}_i \in \mathbb{R}^p$ whose jth component is

$$(\mathbf{r}_i)_j = \sqrt{\beta_j - \alpha} U_{ij} \,,$$

where $\alpha \leq \beta_p$ and $j \in \{1, 2, \ldots, p\}$. The modularity Q is therefore approximated as

$$Q \simeq \hat{Q} = N\alpha + \sum_{l=1}^{\hat{n}} \|\mathbf{R}_l\|_{L_2}^2 \,, \tag{4.22}$$

where $\mathbf{R}_l = \sum_{g_i=l} \mathbf{r}_i$ is sum of all node vectors in the lth community (where $l \in \{1, 2, \ldots, \hat{n}\}$).

A partition that maximizes (4.22) in a given step must satisfy the geometric constraints $\mathbf{R}_l \cdot \mathbf{r}_i > 0$, $g_i = l$, and $\mathbf{R}_l \cdot \mathbf{R}_h < 0$ for all $l, h \in \{1, 2, \ldots, \hat{n}\}$. Hence, if one constructs an approximation $\hat{Q}$ using p eigenvectors, a network component can be split into at most $p + 1$ groups in a given recursive step. The choice $p = 2$ allows either bipartitioning or tripartitioning in each recursive step. Reference [68] discusses the case of general p but reports results for recursive bipartitioning with $p = 1$. Reference [79] implements this spectral method with $p = 2$ and a choice of bipartitioning or tripartioning at each recursive step.

In Table 4.4, we report diagnostics for partitions obtained by several algorithms (for $\gamma = 1$). For the recursive spectral bipartitioning and tripartitioning, we use Matlab code provided by the authors of Ref. [79]. They informed us that this particular implementation was not optimized for speed, so we expect it to be slow. One can create much faster implementations of the same spectral method. The utility of this method for the present comparison is that Ref. [79] includes a detailed discussion of its application to the network of network scientists. Each partitioning step in this spectral scheme either bipartitions or tripartitions a group of nodes. Moreover, as discussed in Ref. [79], a single step of the spectral tripartitioning is by itself interesting. Hence, we specify $\hat{n} = 3$ for the Modularity MBO scheme as a comparison.

From Table 4.4, we see that the Modularity MBO scheme with $\hat{n} = 3$ gives a higher modularity than a single tripartition, and the former's NMI and purity are both significantly higher. When we do not specify the number of clusters,

	N_c	Q	NMI	Purity	Time (seconds)
GenLouvain	19	0.8500	1	1	0.5 s
Spectral Recursion	39	0.8032	0.8935	0.9525	60 s
RMM	23	0.8344	0.9169	0.9367	0.8 s
Tripartition	3	0.5928	0.3993	0.8470	50 s
Modularity MBO	3	0.6165	0.5430	0.9974	0.4 s

Table 4.4: Computational time and accuracy comparison on network-science coauthorships.

the RMM scheme achieves a higher modularity score and NMI than recursive bipartitioning/tripartitioning, though the former's purity is lower (which is not surprising due to its larger N_c). The RMM scheme and GenLouvain have similar run times. For any of these methods, one can of course use subsequent post-processing, such as Kernighan-Lin node-swapping steps [68, 76, 79], to find higher-modularity partitions.

4.6.4 A note on computational heuristics and time complexity

Numerous computational heuristics have been employed to optimize network modularity [33, 76]. We have compared our results with implementations of a small number of popular methods that others have made available. We report computation times in our discussions. Our results above demonstrate the good speed of our method, but we have not, for example, included a comparison of its speed with Blondel et al.'s C++ implementations [10] of the Louvain method [9]. Importantly, the low computational cost of our method is a direct result of its firm theoretical grounding, and our reformulation of the problem of modularity optimization offers hope for the development of even faster methods in the future.

4.7 Conclusion and discussion

In summary, we present a novel perspective on the problem of modularity optimization by reformulating it as a minimization of an energy functional involving the total variation on a graph. This provides an interesting bridge between the network science and compressive sensing communities, and it allows the use of techniques from compressive sensing and image processing to tackle modularity optimization. We propose the MBO scheme that can handle large data at very low computational cost. Our algorithms produce competitive results compared to existing methods, and they scale well in terms of speed for certain networks (such as the MNIST data). After computing the eigenvectors of the graph Laplacian, the time complexity of each MBO iteration step is $O(N)$.

One major part of our schemes is to calculate the leading eigenvector-eigenvalue pairs, so one can benefit from the fast numerical Rayleigh-Chebyshev procedure in Ref. [3] when dealing with large, sparse networks. Furthermore, for a given network (which is represented by a weighted adjacency matrix), one can reuse previously computed eigen-decompositions for different choices of initial functions, different values of $\hat{n}$, and different values of the resolution parameter γ. This provides welcome flexibility, and it can be used to significantly reduce computation time because the MBO step is extremely fast, as each step is $O(N)$ and the number of iterations is empirically small.

Importantly, our reformulation of modularity also provides the possibility to incorporate partial ground truth. This can be accomplished either by feeding the information into the initial function or by adding a fidelity term into the functional. (We only pursue the former approach in this work.) It is not obvious how to incorporate partial ground truth using previous optimization methods. This ability to use our method either for unsupervised or for semi-supervised clustering is a significant boon.

CHAPTER 5

Graph Mumford-Shah in Hyperspectral Data

In this chapter we focus on the multi-class segmentation problem using the piecewise constant Mumford-Shah model (1.1) in a graph setting. A graph variational method is developed similarly to the previous chapter. Application of the proposed method on the problem of chemical plume detection in hyper-spectral video data is presented afterwards.

Recall the definition of the continuous piecewise constant Mumford-Shah model introduced in Chapter 1:

$$E^{\mathrm{MS}}(\Phi, \{c_r\}_{r=1}^{\hat{n}}) = |\Phi| + \lambda \sum_{r=1}^{\hat{n}} \int_{\Omega_r} (u_0 - c_r)^2 \,, \tag{5.1}$$

where a given contour Φ segments an image region Ω into $\hat{n}$ many disjoint subregions $\Omega = \cup_{r=1}^{\hat{n}} \Omega_r$, u_0 is the observed imagery data, $\{c_r\}_{r=1}^{\hat{n}}$ is a set of constant values, and $|\Phi|$ denotes the length of the contour Φ.

To study the hyper-spectral data via a graph, each node (pixel) n_i is associated with a d-dimensional feature vector (spectral channels). Let $u_0 : G \to \mathbb{R}^d$ denote the raw hyper-spectral data, where $u_0(n_i)$ represents the d-dimensional spectral channels of n_i. In this setting, we present a graph version of the multi-class *piecewise constant Mumford-Shah* energy functional:

$$MS(f, \{c_r\}_{r=1}^{\hat{n}}) := \frac{1}{2} |f|_{TV} + \lambda \sum_{r=1}^{\hat{n}} \langle \|u_0 - c_r\|^2, f_r \rangle \,, \tag{5.2}$$

where $\{c_r\}_{r=1}^{\hat{n}} \subset \mathbb{R}^d$, $\|u_0 - c_r\|^2$ denotes an $N \times 1$ vector

$$\left(\|u_0(n_1) - c_r\|^2, \ldots, \|u_0(n_N) - c_r\|^2 \right)^T \,,$$

and $\langle \|u_0 - c_r\|^2, f_r \rangle = \sum_{i=1}^{N} f_r(n_i) \|u_0(n_i) - c_r\|^2$. Note that when n_i and n_j belong to different classes, we have $|f(n_i) - f(n_j)| = 2$, which leads to the coefficient in front of the term $\frac{1}{2} |f|_{TV}$.

To see the connection between (5.2) and (5.1), one first observes that f_r is the characteristic function of the r-th class, and thus $\langle \|u_0 - c_r\|^2, f_r \rangle$ is analogous to the term $\int_{\Omega_r} (u_0 - c_r)^2$ in (5.1). Furthermore, the total variation of the characteristic function of a region gives the length of its boundary contour, and therefore $|f|_{TV}$ is the graph analogy of $|\Phi|$.

In order to find a segmentation for G, we propose to solve the following minimization problem:

$$\min_{f \in \mathbb{B}, \{c_r\}_{r=1}^{\hat{n}} \subset \mathbb{R}^d} MS(f, \{c_r\}_{r=1}^{\hat{n}}) \,. \tag{5.3}$$

The resulting minimizer f yields a partition of G.

One can observe that the optimal solution of (5.3) must satisfy:

$$c_r = \frac{\langle u_0, f_r \rangle}{\sum_{i=1}^{N} f_r(n_i)} \,, \tag{5.4}$$

if the r-th class is non-empty.

Note that for the minimization problem given in (5.3), it is essentially equivalent to the K-means method when λ goes to $+\infty$. When $\lambda \to 0$, the minimizer approaches a constant.

5.1 Mumford-Shah MBO and Lyapunov functional

The authors of [62, 63] introduce an efficient algorithm (known as the MBO scheme) to approximate motion by mean curvature of an interface in Euclidean space. The general procedure of the MBO scheme alternates between solving a linear heat equation and thresholding. One interpretation of the scheme is that it replaces the non-linear term of the Allen-Cahn equation with thresholding [29].

In this section we propose a variant of the original *MBO* scheme to approximately find the minimizer of the energy $MS(f, \{c_r\}_{r=1}^{\hat{n}})$ presented in (5.2). Inspired by the work of [30, 89], we write out a Lyapunov functional $Y_\tau(f)$ for our algorithm and prove that it decreases at each iteration of the MBO scheme.

5.1.1 Mumford-Shah MBO scheme

We first introduce a "diffuse operator" $\Gamma_\tau = e^{-\tau \mathbf{L}}$, where $\mathbf{L}$ is the graph Laplacian defined above and τ is a time step size. The operator Γ_τ is analogous to the diffuse operator $e^{-\tau\Delta}$ of the heat equation in PDE (Euclidean space). It satisfies the following properties.

Proposition 2. *Given $\Gamma_\tau = e^{-\tau \mathbf{L}}$, firstly Γ_τ is strictly positive definite, i.e. $\langle f, \Gamma_\tau f\rangle > 0$ for any $f \in \mathbb{K}$, $f \neq 0$. Secondly, Γ_τ conserves the mass, i.e. $\langle 1, \Gamma_\tau f\rangle = \langle 1, f\rangle$. At last, the quantity $\frac{1}{2\tau}\langle 1 - f, \Gamma_\tau f\rangle$ approximates $\frac{1}{2}|f|_{TV}$, for any $f \in \mathbb{B}$.*

Proof. Taylor expansion gives

$$e^{-\tau \mathbf{L}} = I - \tau \mathbf{L} + \frac{\tau^2}{2!}\mathbf{L}^2 - \frac{\tau^3}{3!}\mathbf{L}^3 + \dots$$

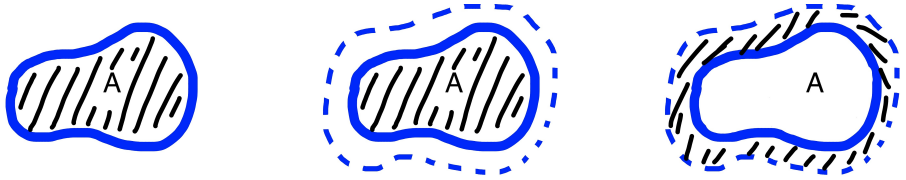
Suppose v is an eigenvector of $\mathbf{L}$ associated with the eigenvalue ξ. One then has $\Gamma_\tau v = e^{-\tau\xi} v \;\Rightarrow \langle v, \Gamma_\tau v\rangle = e^{-\tau\xi}\langle v, v\rangle > 0$. Let the eigen-decomposition (with respect to $\mathbf{L}$) for a non-zero $f : G \to \mathbb{R}$ to be $f = \sum_{i=1}^N a_i \phi_i$, where $\{\phi_i\}_{i=1}^N$ is a set of orthogonal eigenvectors of $\mathbf{L}$ (note that $\mathbf{L}$ is positive definite). Because Γ_τ is a linear operator, one therefore has $\langle f, \Gamma_\tau f\rangle = \sum_{i=1}^N a_i^2 \langle \phi_i, \Gamma_\tau \phi_i\rangle > 0$.

For the second property, $\mathbf{L1} = 0 \Rightarrow \langle \mathbf{1}, \mathbf{L}^k f\rangle = 0$, where $\mathbf{1}$ is an N-dimensional vector with one at each entry. Therefore, the Taylor expansion of Γ_τ gives $\langle \mathbf{1}, \Gamma_\tau f\rangle = \langle \mathbf{1}, f\rangle$.

At last, $\Gamma_\tau \simeq I - \tau\mathbf{L} \Rightarrow \frac{1}{2\tau}\langle 1 - f, \Gamma_\tau f\rangle \simeq \frac{1}{2\tau}\langle 1 - f, f\rangle - \frac{1}{2}\langle 1, \mathbf{L}f\rangle + \frac{1}{2}\langle f, \mathbf{L}f\rangle$. Particularly when $f \in \mathbb{B}$, we have $\frac{1}{2\tau}\langle 1 - f, f\rangle = \frac{1}{2}\langle 1, \mathbf{L}f\rangle = 0$ and $\frac{1}{2}\langle f, \mathbf{L}f\rangle =$

$\frac{1}{2}|f|_{TV}$. Hence $\frac{1}{2\tau}\langle 1-f, \Gamma_\tau f\rangle$ approximates $\frac{1}{2}|f|_{TV}$ in $\mathbb{B}$. $\square$

An intuitive interpretation of the third property of Proposition 2 is illustrated in Figure 5.1. For an indicator function $f = \chi_A$ shown in (a), the diffuse operator propagates the nonzero support set of f to leak outside of A by the scale of τ (time step), as shown in (b). The inner product $\langle 1-f, \Gamma_\tau f\rangle$ gives the shadowed margin (c) around the boundary, which is an approximation of the perimeter of set A. Note that the operator $(I+\tau\mathbf{L})^{-1}$ also satisfies the above three properties, and can serve the same purpose as $e^{-\tau\mathbf{L}}$, as far as this work concerns.



(a) f (b) $\Gamma_\tau f$ (c) $\langle 1-f, \Gamma_\tau f\rangle$

Figure 5.1: Illustration of the third property of Proposition 2: the quantity $\frac{1}{2\tau}\langle 1-f, \Gamma_\tau f\rangle$ approximates $\frac{1}{2}|f|_{TV}$, for any $f \in \mathbb{B}$.

The proposed *Mumford-Shah MBO* scheme for the minimization problem (5.3) consists of alternating between the following three steps:

For a given $f^k \in \mathbb{B}$ at the k-th iteration and $c_r^k = \frac{\langle u_0, f_r^k\rangle}{\langle \mathbf{1}, f_r^k\rangle}$,

1. Compute

$$\hat{f} = \Gamma_\tau f^k - \tau\lambda\left(\|u_0 - c_1^k\|^2, \|u_0 - c_2^k\|^2, \ldots, \|u_0 - c_{\hat{n}}^k\|^2\right), \tag{5.5}$$

2. (Thresholding)

$$f^{k+1}(n_i) = \vec{e}_r\,, \quad r = \text{argmax}_c \hat{f}_c(n_i)$$

for all $i \in \{1, 2, \ldots, N\}$, where $\vec{e}_r$ is the r-th standard basis in $\mathbb{R}^{\hat{n}}$, i.e. $f_r^{k+1}(n_i) = 1$ and $f^{k+1} \in \mathbb{B}$.

3. (Update c)

$$c_r^{k+1} = \frac{\langle u_0, f_r^{k+1}\rangle}{\langle \mathbf{1}, f_r^{k+1}\rangle}\,.$$

5.1.2 A Lyapunov functional

We introduce a Lyapunov functional Y_τ for the Mumford-Shah MBO scheme:

$$Y_\tau(f) := \frac{1}{2\tau}\langle 1-f, \Gamma_\tau f\rangle + \lambda\sum_{r=1}^{\hat{n}}\langle \|u_0 - c_r\|^2, f_r\rangle\,, \quad \text{subject to } c_r = \frac{\langle u_0, f_r\rangle}{\langle \mathbf{1}, f_r\rangle}\,. \tag{5.6}$$

According to the third property of Γ_τ in Proposition 1, the energy $Y_\tau(f)$ approximates $MS(f, \{c_r\}_{r=1}^{\hat{n}})$ for $f \in \mathbb{B}$ and $c_r = \frac{\langle u_0, f_r\rangle}{\langle \mathbf{1}, f_r\rangle}$. A similar functional for the graph total variation is shown and discussed in [90].

Pursuing similar ideas as in [30, 89], we present the following analysis which consequently shows that the Mumford-Shah MBO scheme (with time step τ) decreases Y_τ and converges to a stationary state within a finite number of iterations.

First define

$$G_\tau(f, c) := \frac{1}{2\tau}\langle 1-f, \Gamma_\tau f\rangle + \lambda\sum_{r=1}^{\hat{n}}\langle \|u_0 - c_r\|^2, f_r\rangle\,. \tag{5.7}$$

Proposition 3. *The functional $G_\tau(\cdot, c)$ is strictly concave on $\mathbb{K}$, for any fixed $\{c_r\}_{r=1}^{\hat{n}} \in \mathbb{R}^d$.*

Proof. Take $f, g \in \mathbb{K}$, $\alpha \in (0,1)$. We have $(1-\alpha)f + \alpha g \in \mathbb{K}$, because $\mathbb{K}$ is a convex set.

$$\begin{aligned} & G_\tau\left((1-\alpha)f + \alpha g, c\right) - (1-\alpha)G_\tau(f,c) - \alpha G_\tau(g,c) \\ = \;& \frac{1}{2\tau}\alpha(1-\alpha)\langle f-g, \Gamma_\tau(f-g)\rangle \geq 0\,. \end{aligned} \tag{5.8}$$

Equality only holds when $f = g$. Therefore, $G_\tau(\cdot, c)$ is strictly concave on $\mathbb{K}$.

□

Aside from the concavity of G_τ, we observe that the first order variation of $G_\tau(\cdot, c)$ is given as

$$\frac{\delta}{\delta f} G_\tau(f, c) = \frac{1}{2\tau}(1 - 2\Gamma_\tau f) + \lambda \left(\|u_0 - c_1\|^2, \|u_0 - c_2\|^2, \ldots \right) .$$

Note that since $\langle \frac{\delta}{\delta f} G_\tau(f^k, c^k), f \rangle$ is linear, the Step 2 (thresholding) in the Mumford-Shah MBO scheme is equivalent to

$$f^{k+1} := \text{argmin}_{f \in K} \langle \frac{\delta}{\delta f} G_\tau(f^k, c^k), f \rangle .$$

Theorem 6. *In the Mumford-Shah MBO scheme, the Lyapunov functional $Y_\tau(f^{k+1})$ at the $(k+1)$-th iteration is no greater than $Y_\tau(f^k)$. Equality only holds when $f^k = f^{k+1}$. Therefore, the scheme achieves a stationary point in $\mathbb{B}$ within a finite number of iterations.*

Proof.

$$f^{k+1} := \text{argmin}_{f \in K} \langle \frac{\delta}{\delta f} G_\tau(f^k, c^k), f \rangle \tag{5.9}$$

$\Rightarrow f^{k+1} \in \mathbb{B}$ (due to linearity) and

$$\begin{aligned} 0 &\geq \langle \frac{\delta}{\delta f} G_\tau(f^k, c^k), f^{k+1} - f^k \rangle \\ &\geq G_\tau(f^{k+1}, c^k) - G_\tau(f^k, c^k) \qquad \text{(concavity)} \end{aligned} \tag{5.10}$$

$\Rightarrow G_\tau(f^{k+1}, c^k) \leq G_\tau(f^k, c^k) = Y_\tau(f^k)$. Observe that $c_r^{k+1} = \frac{\langle f_r^{k+1}, u_0 \rangle}{\langle f_r^{k+1}, 1 \rangle}$ is the minimizer of

$$\text{argmin}_{\{c_r\}_{r=1}^{\hat{n}} \in \mathbb{R}^d} G_\tau(f^{k+1}, c)$$

$\Rightarrow G_\tau(f^{k+1}, c^{k+1}) \leq G_\tau(f^{k+1}, c^k) \leq Y_\tau(f^k)$.

$\Rightarrow Y_\tau(f^{k+1}) \leq Y_\tau(f^k)$. Therefore the Lyapunov functional Y_τ is decreasing on the iterations of the Mumford-Shah MBO scheme, unless $f^{k+1} = f^k$. Since $\mathbb{B}$ is a finite set, a stationary point can be achieved in a finite number of iterations.

□

Minimizing the Lyapunov energy Y_τ is an approximation of the minimization problem in (5.3), and the proposed MBO scheme is proven to decrease Y_τ. Therefore, we expect the Mumford-Shah MBO scheme to approximately solve (5.3). In Section 3.3 and Section 3.4, we introduce techniques for computing the MBO iterations efficiently.

5.1.3 Eigen-space approximation

To solve for (5.5) in Step 1 of the Mumford-Shah MBO scheme, one needs to compute the operator Γ_τ, which can be difficult especially for large datasets. For the purpose of efficiency, we numerically solve for (5.5) by using a small number of the leading eigenvectors of $\mathbf{L}$ (which correspond to the smallest eigenvalues), and project f^k onto the eigen-space spanned from the eigenvectors. By approximating the operator $\mathbf{L}$ with the leading eigenvectors, one can compute (5.5) efficiently. We use this approximation because in graph clustering methods, researchers have been using a small portion of the leading eigenvectors of a graph Laplacian to extract structural information of the graph.

Let $\{\phi_m\}_{m=1}^M$ denote the first M (orthogonal) leading eigenvectors of $\mathbf{L}$, and $\{\xi_m\}_{m=1}^M$ the corresponding eigenvalues. Assume $f^k = \sum_{m=1}^M \phi_m a^m$, where a^m is a $1 \times \hat{n}$ vector, with the r-th entry $a_r^m = \langle f_r^k, \phi_m \rangle$. Thus $\hat{f}$ can be approximately computed as:

$$\hat{f} = \sum_{m=1}^{M} e^{-\tau\xi_m}\phi_m a^m - \tau\lambda\left(\|u_0 - c_1^k\|^2, \|u_0 - c_2^k\|^2, \ldots, \|u_0 - c_{\hat{n}}^k\|^2\right). \qquad (5.11)$$

The Mumford-Shah MBO algorithm with the above eigen-space approximation is summarized in Algorithm 1. After the eigenvectors are obtained, each iteration of the MBO scheme is of time complexity $O(N)$. Empirically, the algorithm converges after a small number of iterations. Note that the iterations stop when a *purity* score between the partitions from two consecutive iterations is greater than 99.9%. The purity score, as used in [45] and previous chapters, measures

how "similar" two partitions are. Intuitively, it can be viewed as the fraction of nodes of one partition that have been assigned to the correct class with respect to the other partition.

Algorithm 2 Mumford-Shah MBO Algorithm

Input: f^0, u_0, $\{(\phi_m, \xi_m\}_{m=1}^M$, τ, λ, $\hat{n}$, $k = 0$.

while (purity$(f^k, f^{k+1}) < 99.9\%$) **do**

- $c_r = \frac{\langle u_0, f_r^k \rangle}{\sum_{i=1}^N f_r^k(n_i)}$.
- $a_r^m = \langle f_r^k, \phi_m \rangle$.
- $\hat{f} = \sum_{m=1}^M e^{-\tau\xi_m}\phi_m a^m - \tau\lambda\left(\|u_0 - c_1\|^2, \|u_0 - c_2\|^2, \ldots, \|u_0 - c_{\hat{n}}\|^2\right)$.
- $f^{k+1}(n_i) = \mathbf{e}_r$, where $r = \text{argmax}_c \hat{f}_c(n_i)$.
- $k \leftarrow k + 1$.

end while

5.1.4 Nyström method

The Nyström extension [34] is a matrix completion method which has been used to efficiently compute a small portion of the eigenvectors of the graph Laplacian for segmentation problems [8,35,59,60]. In our proposed scheme, leading eigenvectors of $\mathbf{L}$ are needed, which can require massive computational time and memory. For large graphs such as the ones induced from images, the explicit form of the weight matrix $\mathbf{W}$ and therefore $\mathbf{L}$ is difficult to obtain ($O(N^2)$ time complexity). Hence, we expect to use the Nyström method to approximately compute the eigenvectors for our algorithm.

Given a similarity matrix $\mathbf{W}$ to be computed, the Nyström method only randomly samples a very small number (M) of rows of $\mathbf{W}$, $M << N$. After reordering

the rows, one can assume the first M rows are the sampled ones:

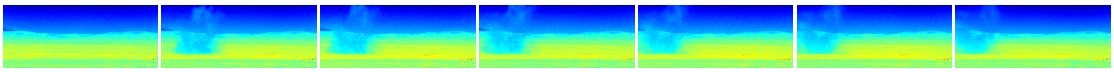
$$\mathbf{W} = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix},$$

i.e. the $M \times M$ matrix A and the $M \times (N-M)$ matrix B are computed and known. The only unknown part of $\mathbf{W}$ is the $(N-M) \times (N-M)$ matrix C. The core idea of the Nyström method is to approximate C by $B^T A^{-1} B$. Based on matrix completion and properties of eigenvectors, ones can approximately obtain M eigenvectors of the symmetric normalized graph Laplacian $\mathbf{L}_{sym}$ without explicitly computing C. Detailed descriptions of the Nyström method can be found in [8, 60].

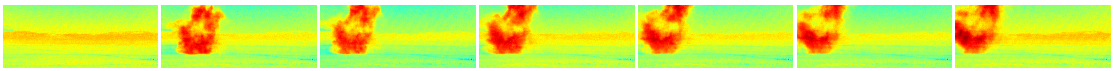
Note that our previous analysis only applies to $\mathbf{L}$ rather than $\mathbf{L}_{sym}$, and the Nyström method can not be trivially formularized for $\mathbf{L}$. Therefore this question remains to be studied. However, the normalized Laplacian $\mathbf{L}_{sym}$ has many similar features compared to $\mathbf{L}$, and it has been used in place of $\mathbf{L}$ in many segmentation problems. In the numerical results shown below, the eigenvectors of $\mathbf{L}_{sym}$ computed via Nyström perform well empirically. One can also implement other efficient methods to compute the eigenvectors for the Mumford-Shah MBO algorithm.

5.2 Numerical Results

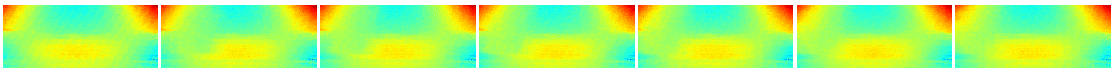
The hyper-spectral images tested in this work are taken from the video recording of the release of chemical plumes at the Dugway Proving Ground, captured by long wave infrared (LWIR) spectrometers. The data is provided by the Applied Physics Laboratory at Johns Hopkins University. A detailed description of this dataset can be found in [18]. We take seven frames from a plume video sequence in which each frame is composed of 128×320 pixels. We use a background frame and the frames numbered 72 through 77 containing the plume. Each pixel has



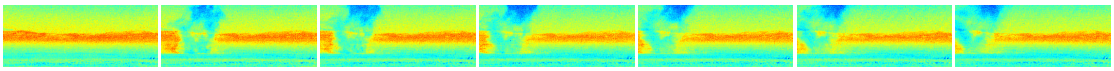
(a) 2nd Eigenvector



(b) 3rd Eigenvector



(c) 4th Eigenvector



(d) 5th Eigenvector

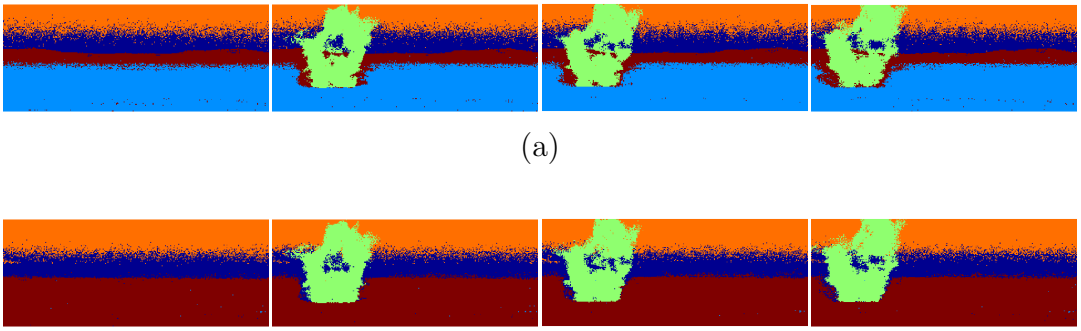
Figure 5.2: The leading eigenvectors of the normalized graph Laplacian computed via the Nyström method.

129 spectral channels corresponding to a particular frequency in the EM spectrum ranging from 7,820 nm to 11,700 nm. Thus, the graph we construct from these seven frames is of size $7 \times 128 \times 320$ with each node n_i corresponding to a pixel with a 129-dimensional spectral signature v_i. The metric for computing the weight matrix is given as:

$$w_{ij} = exp\{-\frac{(1 - \frac{\langle v_i, v_j \rangle}{\|v_i\| \|v_j\|})^2}{2\sigma^2}\},$$

where $\sigma = 0.01$ is chosen empirically for the best performance. Note that in this experiment σ is a robust parameter which gives decent results for a wide range of values ($0.001 < \sigma < 10$).

The goal is to segment the image and identify the "plume cloud" from the background components (sky, mountain,grass), without any ground truth. As described in the previous section, $M = 100$ eigenvectors of the normalized graph Laplacian ($\mathbf{L}_{sym}$) are computed via the Nyström method. The computational



(a)

(b)

Figure 5.3: The segmentation results obtained by the Mumford-Shah MBO scheme, on a background frame plus the frames 72-77. Shown in (a) and (b) are segmentation outcomes obtained with different initializations. The visualization of the segmentations only includes the first four frames.

time using Nyström is less than a minute on a 2.8GHz machine with Intel Core 2Duo. The visualization of the first five eigenvectors (associated with the smallest eigenvalues) are given in Figure 5.2 for the first four frames, (the first eigenvector is not shown because it is close to a constant vector).

We implement the Mumford-Shah MBO scheme using the eigenvectors on this seven frames of plume images, with $\tau = 0.15$, $\lambda = 150$ and $\hat{n} = 5$. The test is run for 20 times with different uniformly random initialization, and the segmentation results are shown in Figure 5.3. Note that depending on the initialization, the algorithm can converge to different local minimum, which is common for most non-convex variational methods. The result in (a) occurred five times among the 20 runs, and (b) for twice. The outcomes of other runs merge either the upper or the lower part of the plume with the background. The segmentation outcome shown in (a) gives higher energy than that in (b). Among the 20 runs, the lowest energy is achieved by a segmentation similar to (a), but with the lower part of the plume merged with the background. It may suggest that the global minimum of the proposed energy does not necessarily give a desired segmentation.

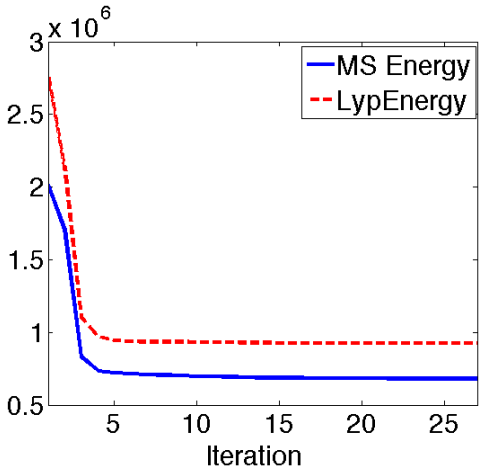
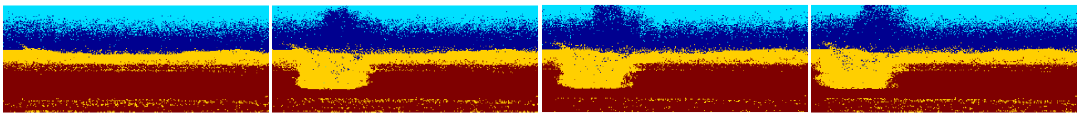


Figure 5.4: Energy $MS(f)$ (blue, solid line) and $Y_\tau(f)$ (red, dash line) at each iteration from the same test as shown in Figure 5.3 (a).

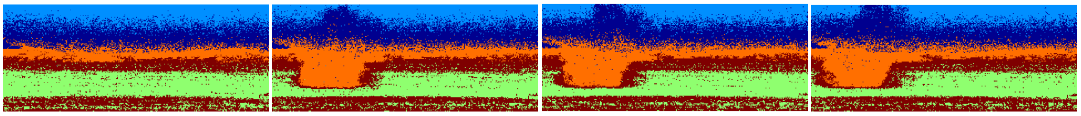
Notice that in Figure 5.3 (b), even though there actually exist five classes, only four major classes can be perceived, while the other one contains only a very small amount of pixels. By allowing $\hat{n} = 5$ instead of $\hat{n} = 4$, it helps to reduce the influence of a few abnormal pixels. The computational time for each iteration is about 2-3 seconds on a 1.7GHz machine with Intel Core i5. The number of iterations is around 20-40.

Figure 5.4 demonstrates a plot of the $MS(f)$ and $Y_\tau(f)$ energies at each iteration from the same test as the one shown in Figure 5.3 (a). The Lyapunov energy $Y_\tau(f)$ (red, dash line) is non-increasing, as proven in Theorem 1. Note that all the energies are computed approximately using eigenvectors.

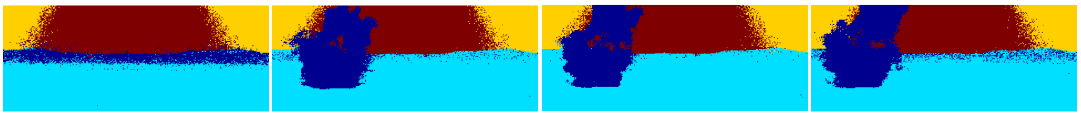
As a comparison, the segmentation results using K-means and spectral clustering are shown in Figure 5.5. The K-means method is performed directly on the raw image data ($7 \times 128 \times 320$ by 129). As shown in (a) and (b), the results obtained by K-means fail to capture the plume; the segmentations on the background are also very fuzzy. For the spectral clustering method, a four-way (or five-way) K-means is implemented on the four (or five) leading eigenvectors of the normalized graph Laplacian (computed via Nyström). As shown in (c)



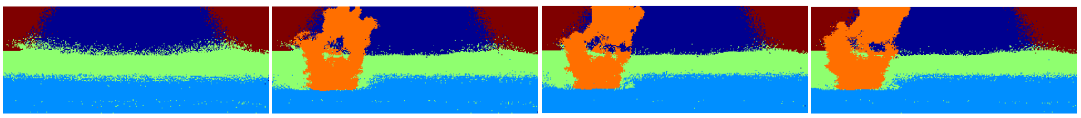
(a) 4-way K-means



(b) 5-way K-means

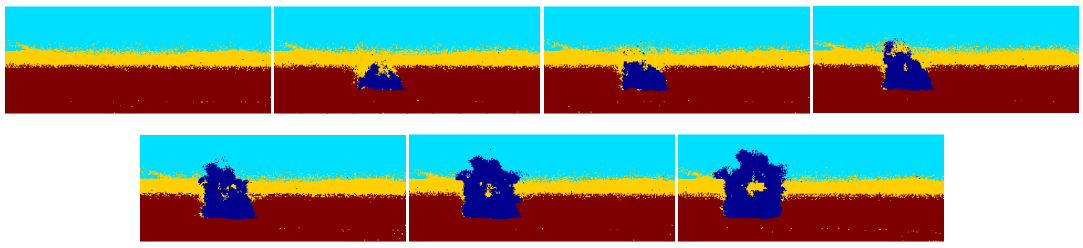


(c) Spectral Clustering with 4-way K-means

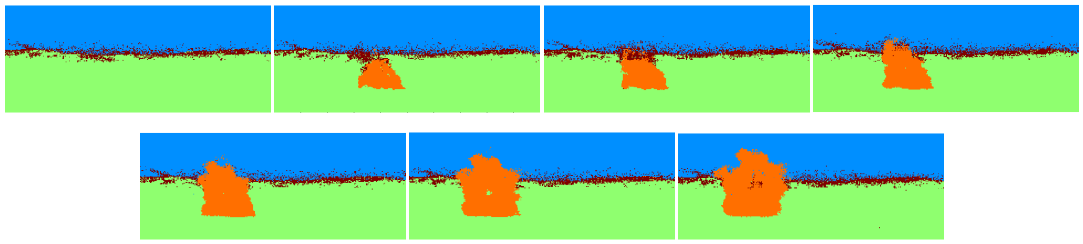


(d) Spectral Clustering with 5-way K-means

Figure 5.5: K-means and spectral clustering segmentation results. The visualization of the segmentations only includes the first four frames.



(a)



(b)

Figure 5.6: The segmentation results obtained by the Mumford-Shah MBO scheme, on a background frame plus the frames 67-72. Shown in (a) and (b) are segmentation outcomes obtained with different initializations.

and (d), the resulting segmentations divide the sky region into two undesirable components. Unlike the segmentation in Figure 5.3 (a) where the mountain component (red, the third in the background) has a well defined outline, the spectral clustering results do not provide clear boundaries. Our approach performs better than other unsupervised clustering results on this dataset [36, 84].

Another example of the plume data is show in Figure 5.6, where the 67th to 72nd frames (instead of the 72nd to 77th) are taken along with the background frame as the test data. The test is run 20 times using different uniformly random initialization, where $\tau = 0.15$, $\lambda = 150$ and $\hat{n} = 5$. The result in Figure 5.6 (a) occurred 11 times among the 20 runs, and (b) for five times. The outcomes from the other four runs segment the background into three components as in (a), but merge the plume with the center component. The segmentation result shown in (a) gives the lowest energy among all the outcomes. The visualization includes all

seven frames since the plume is small in the first several frames.

5.3 Conclusion

In this chapter we present a graph framework for the multi-class piecewise constant Mumford-Shah model using a simplex constrained representation. Based on the graph model, we propose an efficient threshold dynamics algorithm, the Mumford-Shah MBO scheme for solving the minimization problem. Theoretical analysis is developed to show that the MBO iteration decreases a Lyapunov energy that approximates the MS functional. Furthermore, in order to reduce the computational cost for large datasets, we incorporate the Nyström extension method to approximately compute a small portion of the eigenvectors of the normalized graph Laplacian, which does not require computing the whole weight matrix of the graph. After obtaining the eigenvectors, each iteration of the Mumford-Shah MBO scheme is of time complexity $O(n)$. The number of iterations for convergence is small empirically.

The proposed method can be applied to general high-dimensional data segmentation problems. In this work we focus on the segmentation of hyper-spectral video data. Numerical experiments are performed on a collection of hyper-spectral images taken from a video for plume detection; using our proposed method, competitive results are achieved. However, there are still open questions to be answered. For example, the Nyström method can only compute eigenvectors for the normalized Laplacian, while the theoretical analysis for the Lyapunov functional only applies to the un-normalized graph Laplacian. This issue remains to be studied. Note that the graph constructed in this work does not include the spacial information of the pixels, but only the spectral information. One can certainly build a graph incorporating the location of each pixel as well, to generate a non-local means graph as discussed in [8].

CHAPTER 6

An Incremental Reseeding Strategy

This chapter presents the collaborative work in [13], where we propose a resampling-based spectral algorithm for multiway graph partitioning. The algorithm proceeds by alternating three simple routines in an iterative fashion: diffusion, thresholding, and random sampling. On graphs that contain reasonably well-balanced clusters of medium scale, the algorithm provides a strong combination of accuracy, efficiency and robustness to noise in the graph construction process. The algorithm is also exceedingly simple, intuitive and trivial to implement. It also parallelizes trivially, and can therefore scale gracefully to large numbers of clusters as well as to graphs with large numbers of vertices.

We validate these claims via an extensive experimental evaluation of the algorithm. We exhaustively test our algorithm on a total of 26 real world data sets. We also provide a detailed algorithmic comparison using four recent clustering algorithms that claim state-of-the-art results. These experiments demonstrate that our algorithm achieves state-of-the-art performance in terms of cluster purity while running an order of magnitude faster than the other highly accurate clustering methods (e.g. [97]) that we compare against. Moreover, our strategy performs well using only random initialization. This contrasts with other state-of-the-art methods that rely on a lower-level algorithm (such as Normalized Cuts) for initialization. We also provide experiments to demonstrate the robustness of the algorithm with respect to noise and perturbations in the underlying graph. While many highly accurate algorithms exhibit a sharp decrease in accuracy if

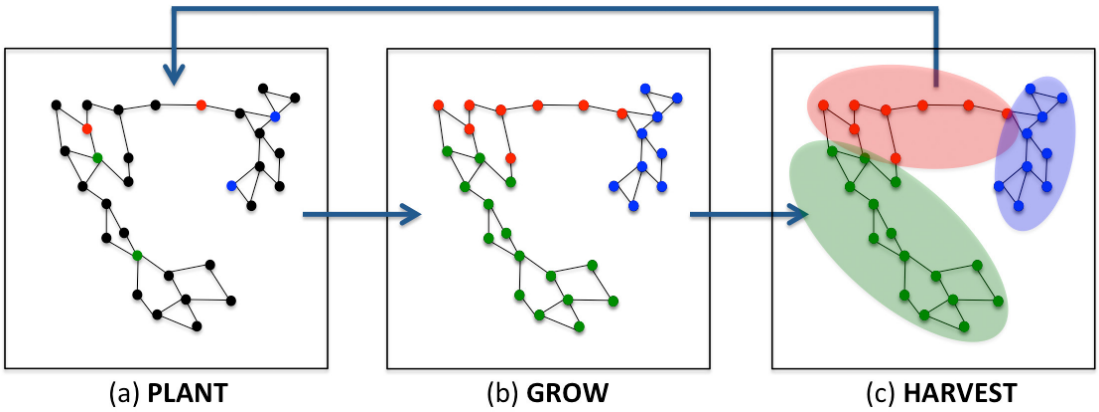


Figure 6.1: Illustration of the Incremental Reseeding (INCRES) Algorithm for $R = 3$ clusters. The various colors red, blue, and green identify the clusters. (a): At this stage of the algorithm, $s = 2$ seeds are randomly planted in the clusters computed from the previous iteration. (b): The seeds grow with the random walk operator. (c): A new partition of the graph is obtained and used to plant $s + ds$ seeds into these clusters at the next iteration.

the input graph is corrupted by noise, our algorithm remains stable: the accuracy of our algorithm decays slowly and gracefully with increasing levels of noise. These results, when taken together, lead to an algorithm with a quite appealing combination of simplicity, performance and ease in out-of-the-box usage.

6.1 Description of the Algorithm

The main idea behind our algorithm arises from a well-known and widely used property of the random walk on a graph. Specifically, a random walker started in a low conductance cluster is unlikely to leave that cluster quickly [56]. This fact provides the basis for transductive label propagation methods [101] as well as for "local" clustering methods [83]. In label propagation, for instance, an oracle provides a set of labeled vertices that are propagated along the graph using a random walk matrix or a diffusion matrix. Each unlabeled vertex is then associated to the

label which, after being propagated, best represents the given unlabeled vertex.

Our algorithm simply iterates upon this basic idea. Assume that the graph has R well-defined clusters of comparable size and low conductance. If we knew these clusters in advance, we could then select a handful of "seed" vertices in the center of each cluster. We would then expect to obtain good results from a transductive label propagation by using these seeds as labels. In an unsupervised context we cannot, of course, *a-priori* place seeds in the center of each cluster. To overcome this, we instead place a handful of seeds at random. We then apply a random walk matrix or diffusion matrix a few times to propagate these seeds. We finally obtain a temporary clustering by assigning each vertex to the seed which, after propagation, best represents the vertex. We then choose new seeds from these temporary clusters and iterate the process. If the clusters improve then the seeds will likely improve, and vice-versa. This incites a feed-back loop and we get a virtuous cycle. We can then excite the speed and improve the quality of this cycle by gradually drawing more and more seeds throughout the process. We refer to this idea as an *incremental reseeding strategy*. Figure 6.1 depicts this cyclic process graphically.

6.1.1 Basic algorithm

To formalize these ideas, recall that (G, E) denotes a weighted, connected graph on N vertices $G = \{n_1, \ldots, n_N\}$ with edge weights $E = \{w_{ij}\}_{i,j=1}^{N}$ that encode a measure of similarity between each pair (n_i, n_j) of vertices. The algorithm starts from a random partition $\mathcal{P} = (\mathcal{C}_1, \ldots, \mathcal{C}_R)$, $\mathcal{C}_1 \cup \cdots \cup \mathcal{C}_R = G$, $\mathcal{C}_r \cap \mathcal{C}_q = \emptyset$ $(r \neq q)$ of the vertices. In other words, each n_i is assigned to one of the R clusters uniformly at random. Let $s = 1$ denote the initial number of seeds. At each of the successive iteration, we update the current partition $\mathcal{P} = (\mathcal{C}_1, \ldots, \mathcal{C}_R)$ according to the steps outlined in the pseudocode Algorithm 3 for INCRES. The pseudocode for the INCRES Subroutines is presented in Algorithm 4.

Algorithm 3 INCRES Algorithm

Input: Similarity matrix $\mathbf{W}$, seed increment ds, number of clusters R.

Initialization: $s = 1$, random partition $\mathcal{P}$.

repeat

$F = \text{PLANT}(\mathcal{P}, s)$

$F \leftarrow \text{GROW}(F, \mathbf{W})$

$\mathcal{P} \leftarrow \text{HARVEST}(F)$

$s \leftarrow s + ds$

until $\mathcal{P}$ converges

Output: $\mathcal{P}$

At the beginning of each iteration, the routine PLANT($\mathcal{P}$,s) will sample s seeds from each of the R clusters $\mathcal{C}_r$ in the current partition $\mathcal{P}$ uniformly at random. These Rs seeds furnish the temporary labels that the GROW routine then propagates along the graph using a random walk matrix. We initialize GROW with an $N \times R$ matrix F, where $F_{i,r} = 1$ if vertex v_i was drawn from cluster $\mathcal{C}_r$ and $F_{i,r} = 0$ otherwise. We then iteratively apply the random walk transition matrix $\mathbf{W}\mathbf{D}^{-1}$ to F until each vertex has a nonzero probability of being visited by a random walker, i.e. until each entry $F_{i,r}$ is nonzero. Finally, the routine HARVEST simply assigns each vertex v_i to its most likely cluster, or in other words the cluster for which $F_{i,r}$ is maximal. This produces a new partition $\mathcal{P}$ of the vertices into R clusters. We then increment s to $s + ds$, and use this partition and number of seeds s to initialize PLANT at the beginning of the next iteration. We refer to this overall procedure as the Incremental Reseeding Algorithm (INCRES).

The overall routine has only a single parameter ds that controls the linear rate at which the number of seeds drawn at each iteration increases. In practice, we select

$$ds = \textbf{speed} \times 10^{-4} \times \frac{N}{R} \tag{6.1}$$

for some proportionality constant **speed** between one and ten. By rescaling ds

Algorithm 4 INCRES Subroutines

function PLANT($\mathcal{P}, s$)

Initialize F as an N-by-R matrix of zeros.

for $r = 1$ to R **do**

for $k = 1$ to s **do**

Draw at random a vertex v_i in cluster $\mathcal{C}_r$.

$F_{i,r} \leftarrow F_{i,r} + 1$

end for

end for

return F.

end function

function GROW($F, \mathbf{W}$)

while $\min_i \min_r F_{i,r} = 0$ **do**

$F \leftarrow \left(\mathbf{W}\mathbf{D}^{-1}\right) F$

end while

return F.

end function

function HARVEST(F)

for $r = 1$ to R **do**

$\mathcal{C}_r = \{i : F_{i,r} \geq F_{i,s} \text{ for all } s\}$

end for

return $\mathcal{P} = (\mathcal{C}_1, \ldots, \mathcal{C}_R)$

end function

in this way, a constant of proportionality **speed**=1 corresponds to a total of $s = 0.1N/R$ seeds planted in each cluster after 1000 iterations. Assuming well-balanced clusters of roughly equal size, approximately one-tenth of each cluster is sampled after 1000 iterations. Drawing a significant fraction of each cluster will cause the subsequent clustering to stabilize, leading to eventual convergence of the algorithm. Using around 10% of labels in each cluster is a typical level at which INCRES will stabilize.

The parameter *ds* therefore represents a "timestep" for INCRES, and the overall algorithm behaves well with respect to this parameter. In general, small increments *ds* will lead to slower convergence at higher accuracy while larger increments *ds* will lead to faster convergence but potentially less accurate solutions. Our experiments show that **speed** = 1 works remarkably well for a large variety of data sets. We also provide results for **speed**=5, which yields faster stabilization with slightly less accuracy, to show that the algorithm is indeed robust and predictable with respect to the choice of this parameter.

The general INCRES framework also proves robust to implementation choices for the three main routines. For instance, in addition to the random walk matrix $\mathbf{WD}^{-1}$, there exists a variety of alternative means to propagate labels along a graph. By-and-large, the overall INCRES strategy does not depend heavily upon the particular implementation of GROW, so long as it realizes the basic idea of label propagation in one form or another. For instance, we have found that replacing the random walk step $F \leftarrow \mathbf{WD}^{-1}F$ with a diffusion step $F \leftarrow \mathbf{D}^{-1}\mathbf{W}F$ or $F \leftarrow \mathbf{D}^{-1/2}\mathbf{WD}^{-1/2}F$ will give similar results in many circumstances. Occasionally, we have found that utilizing a "personalized Page-Rank" step

$$F \leftarrow \alpha\mathbf{WD}^{-1}F + (1-\alpha)F_0 \tag{6.2}$$

can give better performance on small data sets that contain a large (relative to the size of the data set) number of clusters. Here the parameter $0 < \alpha < 1$ denotes a

length-scale that controls the extent of diffusion and F_0 denotes the input to the GROW routine. A propagation step of the form (6.2) is also used in Pagerank-NIBBLE [2] and NMFR [97], up to replacing $\mathbf{WD}^{-1}$ with $\mathbf{D}^{-1/2}\mathbf{WD}^{-1/2}$ in the latter case. As another example, choosing to sample with or without replacement in the PLANT routine leads to essentially no significant difference in the resultant clusterings.

6.1.2 Relation with other work

Our methodology relies upon and incorporates number of ideas from transductive learning. In particular, we leverage the notion of label propagation [101]. In the standard label propagation framework, an oracle provides a set of labeled points or vertices. These labeled points form either nonzero initial conditions or heat sources for a discrete heat equation on the graph. The second step of the INCRES algorithm (the GROW routine) precisely corresponds with a label propagation of the random labels returned from the first step of the algorithm (the PLANT routine).

The NIBBLE algorithm and its relatives [2, 56, 82, 83] use a similar idea to get an unsupervised clustering method from label propagation by planting random seeds. These works cluster the entire graph in a sequential manner: at each step a single random vertex is drawn and propagated. Then a sweep is performed to extract a small cluster around this vertex. These algorithms function well for problems aimed at extracting many small clusters from graphs with fine structure. In contrast, we perform multiway partitioning directly instead of recursively. We also aim at medium scale clusters instead of small scale clusters. We also utilize a significantly different random seeding strategy.

The INCRES algorithm also alternates between label propagation (GROW) and thresholding (HARVEST). The idea of iteratively alternating between a few

steps of label propagation and subsequent thresholding has also appeared in a transductive learning context [35,59], although the presence of labeled information results in a different implementation of the propagation step. The non-negative matrix factorization method [97] also incorporates random walk information in a manner that resembles the GROW routine, but otherwise the underlying principles of the algorithms differ substantially.

Because the GROW function we use iterates the random walk on the graph, our algorithm is a form of spectral clustering. However, our main contribution to the clustering problem, and the primary novelty in our algorithm, is the *incremental reseeding process*. This process is not fundamentally tied to the INCRES algorithm presented here — it seems to be quite universal and can be adapted to other clustering methods. However, combining reseeding with the random walk method offers an excellent combination of accuracy, speed, and robustness.

6.2 Experiments

We now provide the results of our extensive experimental evaluation of the algorithm. This section shows that our algorithm achieves state-of-the-art performance in terms of cluster purity on a variety of real word data sets while running an order of magnitude faster than the other comparably accurate clustering methods. Moreover, whereas some of the competing algorithms are initialized with a partition provided by a low cost algorithm such as NCut, the INCRES algorithm is initialized with a random partition. Finally we show that INCRES is very robust to perturbations in the input graph.

The Algorithms: We compare our method against four clustering algorithms that rely on variety of different principles. We select algorithms that, like our algorithm, partition the graph in a direct, non-recursive manner. The NCut algorithm [98] is a widely used spectral algorithm that relies on a post-processing of the

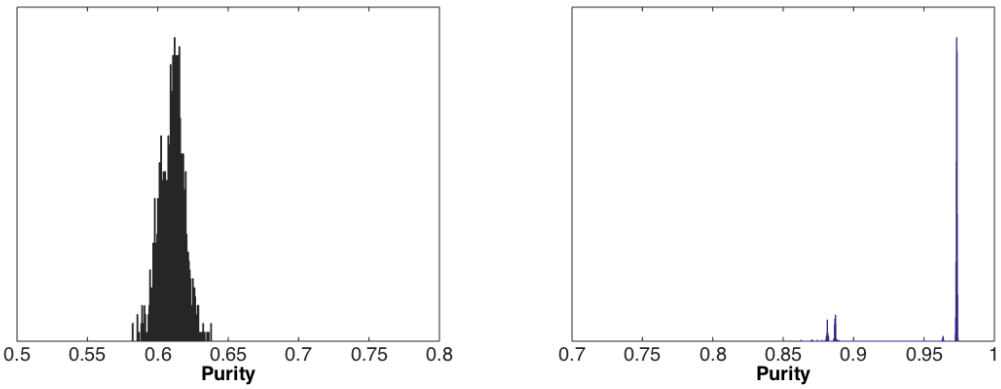
Table 6.1: Algorithmic Comparison via Cluster Purity.

Data	**size**	**R**	**RND**	**NCut**	**LSD**	NMFR	**MTV**	**INCRES** *(speed 1)*	**INCRES** *(speed 5)*
20NEWS	20K	20	6.3%	26.6%	34.3%	60.7%	35.8%	**61.0%**	60.7%
CADE	21K	3	15.5%	41.0%	41.3%	52.0%	44.2%	**52.8%**	52.1%
RCV1	9.6K	4	30.3%	38.2%	38.1%	42.7%	42.8%	**54.5%**	51.2%
WEBKB4	4.2K	4	39.1%	39.8%	45.8%	**58.1%**	45.2%	57.1%	56.8%
CITESEER	3.3K	6	21.8%	23.4%	53.4%	**62.6%**	42.6%	62.0%	62.2%
MNIST	70K	10	11.3%	76.9%	75.5%	**97.1%**	95.5%	96.0%	94.0%
PENDIGIT	11K	10	11.6%	80.2%	86.1%	86.8%	86.5%	**88.8%**	85.9%
USPS	9.3K	10	16.7%	71.5%	70.4%	86.4%	85.3%	**87.8%**	87.4%
OPTDIGIT	5.6K	10	12.0%	90.8%	91.0%	**98.0%**	95.2%	97.4%	94.8%

eigenvectors of the graph Laplacian to optimize the normalized cut energy. The NMFR algorithm [97] uses non-negative matrix factorization and graph-based random walk principles in order to factorize and regularize the original input similarity matrix. The LSD algorithm [4] provides another non-negative matrix factorization algorithm. It aims at finding a left-stochastic decomposition of the similarity matrix. The MTV algorithm from [15] provides a total-variation based algorithm that attempts to find an optimal multiway Cheeger cut of the graph by using L_1 optimization techniques. The last three algorithms (NMFR, LSD and MTV) all use NCut in order to obtain an initial partition. By contrast, we initialize our algorithm with a random partition. We use the code available from [98] for NCut, the code available from [97] to test the two non-negative matrix factorization algorithms (NMFR and LSD) and the code available from [15] for the MTV algorithm.

The Data Sets: We provide experimental results on five text data sets (20NEWS, CADE, RCV1, WEBKB4, CITESEER) and four data sets contain-

ing images of handwritten digits (MNIST, PENDIGITS, USPS, OPTDIGITS). We processed the text data sets by removing a list of stop words as well as by removing all words with fewer than twenty occurrences (for 20NEWS) and fewer than five occurrences (for all others) across the corpus. We then construct a 5-NN graph based on the cosine similarity between tf-idf features. For variety, we include some weighted graphs (RCV1 and CITESEER) as well as some unweighted graphs (20NEWS, CADE and WEBKB4). For MNIST, PENDIGITS and OPTDIGITS we use the similarity matrices constructed by [97], where the authors first extract scattering features [19] for images before calculating an unweighted 10-NN graph. For USPS we constructed a weighted 10-NN graph from the raw data without any preprocessing. The source for these data sets and more details on their construction are provided in the appendix of [13].



(a) 20NEWS (b) MNIST

Figure 6.2: Purity Distributions

6.2.1 Accuracy comparisons

In Table 6.1 we report the accuracy obtained by the selected algorithms LSD, NMFR, MTV and INCRES (for two values of the timestep parameter, **speed** = 1 and **speed** = 5) on the various data sets. Cluster purity, as mentioned in previ-

ous chapters, is used to quantify the quality of the calculated partition, defined according to the relation

$$\text{Purity} = \frac{\text{number of "successes"}}{N} = \frac{1}{N}\sum_{r=1}^{R} \max_{1<i<R} n_{r,i}.$$

Here $n_{r,i}$ denotes the number of data points in the r^{th} cluster that belong to the i^{th} ground-truth class. In other words, given a computed cluster we count a data point as a success if it belongs to the ground truth class that best represents the cluster. We allowed each iterative algorithm a total of 10,000 iterations to reach convergence. Both INCRES and MTV rely on randomization, so for these algorithm we report the average purity achieved over 1000 different runs. The fourth column of the table (RND) provides a base-line purity for reference, i.e. the purity obtained by assigning each data point to a class from 1 to R uniformly at random. The boldface numbers in the table indicate the highest purity score achieved on each data set.

Overall, INCRES and NMFR significantly outperform the other algorithms. This is especially true for text data sets. Both algorithms utilize a random walk strategy to help "smooth" irregular graphs, such as the similarity matrices obtained from text data sets. This strategy also contributes to the robustness of these algorithms and to their solid performance across the full range of data sets. However, the INCRES algorithm typically runs at least one order of magnitude faster than the NMFR algorithm. Due to the similarity of their results, we provide a more exhaustive comparison between these two algorithms in the supplementary material on 17 additional data sets.

Finally, note that the INCRES algorithm performs comparably when **speed** = 1 and **speed** = 5, demonstrating that the algorithm is robust with respect to the choice of the seed increment parameter ds. The INCRES algorithm also performs rather consistently across independent runs of the random process. Figure 6.2 provides an experimental validation of this fact. Specifically, this figure displays a

histogram of the purity values obtained over 1008 independent trials of INCRES on both 20NEWS and MNIST (using **speed** = 1 for this example). On 20NEWS, all independent trials of the INCRES algorithm converge to a solution with a purity value between 58% and 64%. The INCRES algorithm converges to one of two attractors on MNIST, one at roughly 88% purity and one at roughly 97% purity, with the vast majority of all trials converging to the higher-accuracy attractor.

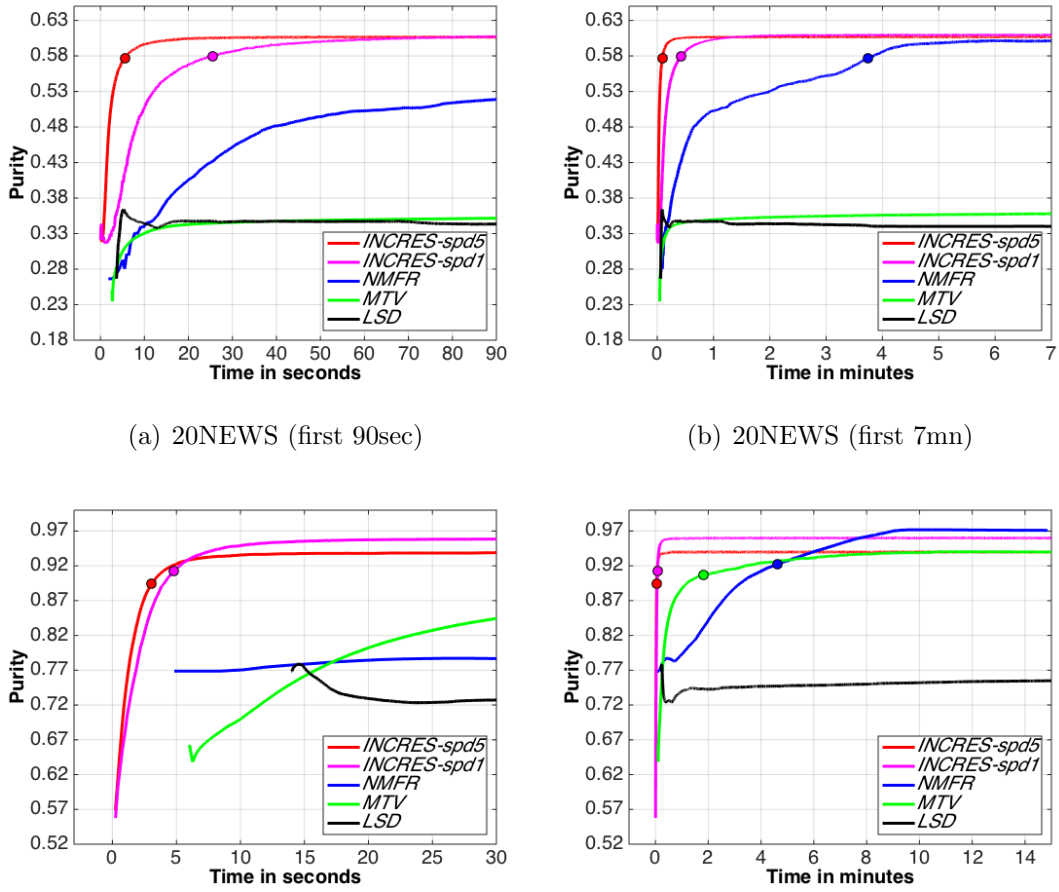
6.2.2 Speed comparisons

Figure 6.3 illustrates the speed at which the iterative algorithms (LSD, MTV, NMFR and INCRES) converge toward their respective solutions. We ran each algorithm for a total of 7 minutes on 20NEWS and for 15 minutes on MNIST. We report the purity obtained by the algorithm at each iteration. For the randomized algorithm (INCRES and MTV) the purity curves were obtained by averaging the results over 240 runs. The overwhelming computational burden for all of these algorithms arises from the sparse-matrix times full-matrix multiplications required at each step. Each algorithm is implemented in a fair and consistent way, and the experiments were all performed on the same architecture.

Table 6.2: Computational Time

Data	**NMFR**	**MTV**	**INCRES** *(speed1)*	**INCRES** *(speed5)*
20NEWS	3.7mn	–	25.4s	5.6s
	(57.7%)	–	(57.7%)	(58%)
MNIST	4.6mn	1.8mn	4.8s	3.1s
	(92.2%)	(90.7%)	(91.2%)	(89.3%)

In order to give an indication of the speed/accuracy trade-off for each algorithm, in Table 6.2 we record the time it took for the purity obtained by each algorithm to reach 95% of its limiting value on both 20NEWS and on MNIST. Overall,



(a) 20NEWS (first 90sec)

(b) 20NEWS (first 7mn)

(c) MNIST (first 30sec)

(d) MNIST (first 15mn)

Figure 6.3: Purity curves for the four algorithms considered on two benchmark data sets (20NEWS and MNIST). We plot purity against time for each algorithm over two different time windows. The circular marks on each curve indicate the point at which the curve reaches 95% of its limiting value. The corresponding times at which this happens are reported in Table 6.2.

the simple INCRES algorithm provides accuracy comparable to the state-of-the-art NMF algorithm [97], yet runs an order of magnitude faster. Timing results on the data sets from table 6.1 are consistent with those obtained for 20NEWS and MNIST, in the sense that INCRES typically runs one order of magnitude faster than NMFR on these data sets as well.

Table 6.3: Robustness Comparisons

Noise	**NCut**	**LSD**	**MTV**	**INCRES** *(speed1)*
20NEWS				
+0% edges	27%	34%	36%	**61%**
+50% edges	21%	27%	20%	**52%**
+100% edges	18%	22%	11%	**44%**
+150% edges	15%	20%	10%	**34%**
+200% edges	14%	18%	9%	**27%**
MNIST				
+0% edges	77%	76%	96%	96%
+50% edges	87%	94%	55%	**97%**
+100% edges	84%	93%	25%	**97%**
+150% edges	74%	87%	18%	**97%**
+200% edges	67%	82%	16%	**96%**

6.2.3 Robustness experiments

Table 6.3 reports accuracy results of various algorithms on graphs that we corrupted by adding different levels of noise. We began with the original 20NEWS graph used in Table 6.1 and added additional edges to the graph uniformly at random. The original graph had $e = 144,632$ edges. For the experiment, we added $0.5e$, e, $1.5e$ and $2e$ additional noise edges. For each of these four levels of noise, we randomly generated 144 separate perturbed graphs. The table reports, for each level of noise, the average purity obtained by each algorithm on the 144 randomly generated matrices. We then proceeded to perturb the original MNIST graph in a similar fashion. The original graph has $e = 1,027,412$ edges, and we randomly generated 120 graphs at each level of noise. This gives a total of 1056 randomly generated graphs for this set of experiments. We provide experimental

results for all algorithms other than NMFR, which simply takes far too much time to run to convergence on all 1056 adjacency matrices. Nevertheless, given the similarity in their performace, we expect that NMFR would perform similarly to INCRES on this set of experiments as well.

The results clearly elucidate the robustness of the INCRES algorithm with respect to noise in the graph construction process. On the 20NEWS data set, for example, all other algorithms experience a sharp decrease in accuracy as soon as noise is added. In contrast, the purity of the INCRES algorithm slowly decreases in a stable fashion. On the MNIST data sets, the results obtained by INCRES remain essentially unchanged across all noise levels. The competing algorithms do not exhibit this behavior. Interestingly, NCut and LSD actually obtain *better* results at the 50% and 100% noise levels. Given that LSD relies on NCut for initialization, it comes as no surprise that gains for NCut produce subsequent gains for LSD as well. This pathological behavior still indicates a lack of robustness, in the sense that both algorithms exhibit a high degree of sensitivity to changes in the underlying graph.

References

[1] MNIST database. *http://yann.lecun.com/exdb/mnist/*.

[2] Reid Andersen, Fan R. K. Chung, and Kevin Lang. Local graph partitioning using PageRank vectors. In *Proceedings of the 47th Annual Symposium on Foundations of Computer Science*, pages 475–486, 2006.

[3] Christopher R. Anderson. A Rayleigh-Chebyshev procedure for finding the smallest eigenvalues and associated eigenvectors of large sparse Hermitian matrices. *Journal of Computational Physics*, 229(19):7477–7487, 2010.

[4] Raman Arora, Maya R. Gupta, Amol Kapila, and Maryam Fazel. Clustering by left-stochastic matrix factorization. In *International Conference on Machine Learning (ICML)*, pages 761–768, 2011.

[5] Guy Barles and Christine Georgelin. A simple proof of convergence for an approximation scheme for computing motions by mean curvature. *SIAM Journal on Numerical Analysis*, 32(2):484–500, 1995.

[6] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009.

[7] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[8] Andrea L. Bertozzi and Arjuna Flenner. Diffuse interface models on graphs for classification of high dimensional data. *Multiscale Modeling & Simulation*, 10(3):1090–1118, 2012.

[9] Vincent D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[10] Vincent D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Louvain method: finding communities in large networks. *https://sites.google.com/site/findcommunities/*, 2008.

[11] Stefan Boettcher and Allon G. Percus. Optimization with extremal dynamics. *Complexity*, 8(2):57–62, 2002.

[12] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[13] Xavier Bresson, Huiyi Hu, Thomas Laurent, Arthur Szlam, and James von Brecht. An incremental reseeding strategy for clustering. *Preprint: arXiv:1406.3837*, 2014.

[14] Xavier Bresson, Thomas Laurent, David Uminsky, and James von Brecht. Convergence and energy landscape for Cheeger cut clustering. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1385–1393, 2012.

[15] Xavier Bresson, Thomas Laurent, David Uminsky, and James von Brecht. Multiclass total variation clustering. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1421–1429, 2013.

[16] Xavier Bresson and Arthur Szlam. Total variation and Cheeger cuts. In *International Conference on Machine Learning*, pages 1039–1046, 2010.

[17] Xavier Bresson, Xue-Cheng Tai, Tony F. Chan, and Arthur Szlam. Multi-class transductive learning based on ℓ_1 relaxations of Cheeger cut and Mumford-Shah-Potts model. *Journal of mathematical imaging and vision*, 49(1):191–201, 2014.

[18] J. B. Broadwater, D. Limsui, and A. K. Carr. A primer for chemical plume detection using LWIR sensors. Technical report, 2011.

[19] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.

[20] Xiaohao Cai, Raymond Chan, and Tieyong Zeng. A two-stage image segmentation method using a convex variant of the Mumford-Shah model and thresholding. *SIAM Journal on Imaging Sciences*, 6(1):368–390, 2013.

[21] Tony F. Chan and Luminita A. Vese. Active contours without edges. *IEEE transactions on Image processing*, 10(2):266–277, 2001.

[22] Fan R. K. Chung. *Spectral graph theory*, volume 92. American Mathematical Society, 1997.

[23] Aaron Clauset, Mark E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[24] Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[25] Mihai Cucuringu, Vincent D. Blondel, and Paul Van Dooren. Extracting spatial information from networks with low-order eigenvectors. *Physical Review E*, 87(3):032803, 2013.

[26] G. Dal Maso. An introduction to gamma convergence. *Progress in Nonlinear Differential Equations and its Applications*, 1989.

[27] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

[28] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72(2):027104, 2005.

[29] Selim Esedog and Yen-Hsi R. Tsai. Threshold dynamics for the piecewise constant Mumford–Shah functional. *Journal of Computational Physics*, 211(1):367–384, 2006.

[30] Selim Esedoglu and Felix Otto. Threshold dynamics for networks with arbitrary surface tensions. *Communications on Pure and Applied Mathematics*, 2014.

[31] Lawrence C. Evans. Convergence of an algorithm for mean curvature motion. *Indiana University Mathematics Journal*, 42(2):533–557, 1993.

[32] David J. Eyre. An unconditionally stable one-step scheme for gradient systems. *Unpublished article*, 1998.

[33] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[34] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[35] Cristina Garcia-Cardona, Ekaterina Merkurjev, Andrea L. Bertozzi, Arjuna Flenner, and Allon G. Percus. Multiclass data segmentation using diffuse interface methods on graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99:1, 2014.

[36] Torin Gerhart, Justin Sunu, Lauren Lieu, Ekaterina Merkurjev, Jen-Mei Chang, Jérôme Gilles, and Andrea L. Bertozzi. Detection and tracking of gas plumes in LWIR hyperspectral video sequence data. In *SPIE Defense, Security, and Sensing*, pages 87430J–87430J. International Society for Optics and Photonics, 2013.

[37] Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[38] Benjamin H. Good, Yves-Alexandre de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.

[39] Roger Guimera and Luis A. Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005.

[40] Roger Guimerà, Marta Sales-Pardo, and Luís A. Nunes Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2):025101, 2004.

[41] Lars Hagen and Andrew B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

[42] Matthew Harris, Xavier Aubert, Reinhold Haeb-Umbach, and Peter Beyerlein. A study of broadcast news audio stream segmentation and segment clustering. In *6th European Conference on Speech Communication and Technology*, 1999.

[43] Matthias Hein and Thomas Bühler. An inverse power method for nonlinear eigenproblems with applications in 1-spectral clustering and sparse PCA. In *Advances in Neural Information Processing Systems (NIPS)*, pages 847–855, 2010.

[44] Matthias Hein and Simon Setzer. Beyond spectral clustering-tight relaxations of balanced graph cuts. In *Advances in neural information processing systems*, pages 2366–2374, 2011.

[45] Huiyi Hu, Thomas Laurent, Mason A Porter, and Andrea L Bertozzi. A method based on total variation for network modularity optimization using the MBO scheme. *SIAM Journal on Applied Mathematics*, 73(6):2224–2246, 2013.

[46] Huiyi Hu, Justin Sunu, and Andrea L Bertozzi. Multi-class graph Mumford-Shah model for plume detection using the MBO scheme. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 209–222. Springer, 2015.

[47] Huiyi Hu, Yves van Gennip, Blake Hunter, Andrea L Bertozzi, and Mason A Porter. Multislice modularity optimization in community detection and image segmentation. In *IEEE 12th International Conference on Data Mining Workshops (ICDM)*, pages 934–936. IEEE, 2012.

[48] Tom Ilmanen. Convergence of the Allen-Cahn equation to Brakke's motion by mean curvature. *Journal of Differential Geometry*, 38(2):417–461, 1993.

[49] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[50] Inderjit S. Jutla, Lucas G. S. Jeub, and Peter J. Mucha. A generalized Louvain method for community detection implemented in MATLAB. *http://netwiki. amath. unc. edu/GenLouvain*, 2011.

[51] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[52] Robert V. Kohn and Peter Sternberg. Local minimisers and singular perturbations. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics*, 111(1-2):69–84, 1989.

[53] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *Physical Review E*, 80(5):056117, 2009.

[54] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008.

[55] Anna C. F. Lewis, Nick S. Jones, Mason A. Porter, and Charlotte M. Deane. The function of communities in protein interaction networks at multiple scales. *BMC Systems Biology*, 4(1):100, 2010.

[56] László Lovász and Miklós Simonovits. Random walks in a convex body and an improved volume algorithm. *Random structures & algorithms*, 4(4):359–412, 1993.

[57] Michael Lustig, David L. Donoho, Juan M. Santos, and John M. Pauly. Compressed sensing MRI. *Signal Processing Magazine, IEEE*, 25(2):72–82, 2008.

[58] R. A. Mercovich, A. Harkin, and D. Messinger. Automatic clustering of multispectral imagery by maximization of the graph modularity. In *Proceedings of SPIE*, volume 8048, pages 80480Z–1, 2011.

[59] Ekaterina Merkurjev, Cristina Garcia-Cardona, Andrea L. Bertozzi, Arjuna Flenner, and Allon G. Percus. Diffuse interface methods for multiclass segmentation of high-dimensional data. *Applied Mathematics Letters*, 33:29–34, 2014.

[60] Ekaterina Merkurjev, Tijana Kostic, and Andrea L. Bertozzi. An MBO scheme on graphs for classification and image processing. *SIAM Journal on Imaging Sciences*, 6(4):1903–1930, 2013.

[61] Ekaterina Merkurjev, Justin Sunu, and Andrea L. Bertozzi. Graph MBO method for multiclass segmentation of hyperspectral stand-off detection

video. In *IEEE International Conference on Image Processing*, pages 689–693, 2014.

[62] Barry Merriman, James K. Bence, and Stanley Osher. Diffusion generated motion by mean curvature. In *Computational Cristal Growers Workshop*, pages 73–83, 1992.

[63] Barry Merriman, James K. Bence, and Stanley J Osher. Motion of multiple junctions: A level set approach. *Journal of Computational Physics*, 112(2):334–363, 1994.

[64] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[65] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1989.

[66] Deanna Needell and Rachel Ward. Stable image reconstruction using total variation minimization. *SIAM Journal on Imaging Sciences*, 6(2):1035–1058, 2013.

[67] Mark E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[68] Mark E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006.

[69] Mark E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[70] Mark E. J. Newman. The physics of networks. *Physics Today*, 61(11):33–38, 2008.

[71] Mark E. J. Newman. *Networks: an introduction*. Oxford University Press, 2010.

[72] Mark E. J. Newman and Michelle Girvan. Mixing patterns and community structure in networks. In *Statistical Mechanics of Complex Networks*, pages 66–87. Springer, 2003.

[73] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[74] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: analysis and an algorithm. *Advances in Neural Information Processing Systems (NIPS)*, 2:849–856, 2002.

[75] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[76] Mason A. Porter, Jukka-Pekka Onnela, and Peter J. Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 1164–1166, 2009.

[77] Syama S. Rangapuram and Matthias Hein. Constrained 1-spectral clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 1143–1151, 2012.

[78] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74:016110, 2006.

[79] Thomas Richardson, Peter J. Mucha, and Mason A. Porter. Spectral tripartitioning of networks. *Physical Review E*, 80(3):036111, 2009.

[80] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[81] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[82] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th annual ACM Symposium on Theory of Computing*, pages 81–90, 2004.

[83] Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2013.

[84] Justin Sunu, Jen-Mei Chang, and Andrea L. Bertozzi. Simultaneous spectral analysis of multiple video sequence data for LWIR gas plumes. In *SPIE Defense+ Security*, pages 90880T–90880T, 2014.

[85] Xue-Cheng Tai, Oddvar Christiansen, Ping Lin, and Inge Skjælaaen. Image segmentation using some piecewise constant level set methods with MBO type of projection. *International Journal of Computer Vision*, 73(1):61–76, 2007.

[86] Guillaume Tochon, Jocelyn Chanussot, Jérôme Gilles, Mauro Dalla Mura, Jen-Mei Chang, and Andrea L. Bertozzi. Gas plume detection and tracking in hyperspectral video sequences using binary partition trees. In *IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 2014.

[87] Amanda L. Traud, Eric D. Kelsic, Peter J. Mucha, and Mason A. Porter. Comparing community structure to characteristics in online collegiate social networks. *SIAM Review*, 53(3):526–543, 2011.

[88] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012.

[89] Yves Van Gennip and Andrea L. Bertozzi. Γ-convergence of graph Ginzburg-Landau functionals. *Advances in Differential Equations*, 17(11/12):1115–1180, 2012.

[90] Yves van Gennip, Nestor Guillen, Braxton Osting, and Andrea L. Bertozzi. Mean curvature, threshold dynamics, and phase field theory on finite graphs. *Milan Journal of Mathematics*, 82(1):3–65, 2014.

[91] Yves van Gennip, Huiyi Hu, Blake Hunter, and Mason A. Porter. Geosocial graph-based community detection. In *IEEE 12th International Conference on Data Mining Workshops (ICDM)*, pages 754–758. IEEE, 2012.

[92] Yves van Gennip, Blake Hunter, Raymond Ahn, Peter Elliott, Kyle Luh, Megan Halvorson, Shannon Reid, Matthew Valasik, James Wo, George E. Tita, Andrea L. Bertozzi, and P. Jeffrey Brantingham. Community detection using spectral clustering on sparse geosocial data. *SIAM Journal on Applied Mathematics*, 73(1):67–83, 2013.

[93] Luminita A. Vese and Tony F. Chan. A multiphase level set framework for image segmentation using the Mumford-Shah model. *International Journal of Computer Vision*, 50(3):271–293, 2002.

[94] Benjamin P. Vollmayr-Lee and Andrew D. Rutenberg. Fast and accurate coarsening simulation with an unconditionally stable time step. *Physical Review E*, 68(6):066703, 2003.

[95] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[96] Dorothea Wagner and Frank Wagner. Between min cut and graph bisection. In *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, MFCS '93, pages 744–750, 1993.

[97] Zhirong Yang, Tele Hao, Onur Dikmen, Xi Chen, and Erkki Oja. Clustering by nonnegative matrix factorization using graph random walk. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1088–1096, 2012.

[98] Stella X. Yu and Jianbo Shi. Multiclass spectral clustering. In *International Conference on Computer Vision*, pages 313–319, 2003.

[99] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 1601–1608, 2004.

[100] Yan Zhang, A. J. Friend, Amanda L. Traud, Mason A. Porter, James H. Fowler, and Peter J. Mucha. Community structure in congressional cosponsorship networks. *Physica A: Statistical Mechanics and its Applications*, 387(7):1705–1712, 2008.

[101] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML)*, pages 912–919, 2003.