Title
Community detection using total variation and surface tension

Author
Boyd, Zachary

Publication Date
2018

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Community detection using total variation and surface tension

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

by

Zachary Mark Boyd

2018

ABSTRACT OF THE DISSERTATION

Community detection using total variation and surface tension

by

Zachary Mark Boyd
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2018
Professor Andrea Bertozzi, Chair

In recent years, a massive expansion in the amount of available network data in fields such as social networks, food networks in ecology, similarity networks in machine learning, transportation networks, brain networks, and many others has motivated the development of "network science" to describe all this data. One of the fundamental branches in network science is "community detection," or the decomposition of large networks into coherent subnetworks, which is useful for visualization, data exploration, hypothesis formation, approximation of network dynamics, link prediction, and a host of other tasks.

Two of the most well-known frameworks for community detection are modularity optimization and stochastic block modeling. They can often uncover meaningful community structure in networks from diverse applications. However, both of these approaches are computationally demanding. In this dissertation, I will show how these two statistically-motivated frameworks for community detection can be reinterpreted more geometrically using the language of graph total variation (TV) and (discretized) surface tension, respectively. This change in perspective allows one to leverage algorithms and analytical tools developed for other problems in the fields of compressed sensing, materials science, and nonlinear partial differential equations. One also can adapt arguments from other domains to obtain theoretical guarantees on the performance of these algorithms. I illustrate these

approaches on a number of synthetic and real-world datasets, yielding results competitive with other state-of-the-art techniques on problems from machine learning, image processing, social networks, and biological networks.

The dissertation of Zachary Mark Boyd is approved.

Mason Alexander Porter

Stanley J. Osher

Christopher R. Anderson

Andrea Bertozzi, Committee Chair

University of California, Los Angeles

2018

TABLE OF CONTENTS

1 Introduction **1**

1.1 Network Science and Community Detection 1

1.2 Nonlinear differential equations and variational methods on networks 2

1.3 Modularity Optimization 4

1.4 Stochastic Block Models (SBMs) 5

2 Background **8**

2.1 Notation and Definitions 8

2.2 Total Variation 9

2.3 The Ginzburg–Landau (GL) Functional 13

2.3.1 Choice of Graph Laplacian 14

2.4 Merriman–Bence–Osher Schemes 15

2.5 Similarity Graphs, Hyperspectral Video, and Nonlocal Means 16

2.6 Graph models, NP-completeness, and theoretical guarantees 20

3 Modularity optimization and total variation **21**

3.1 Introduction 21

3.2 Equivalence Theorem and Its Consequences 23

3.2.1 Formulations of Modularity in Terms of TV and Graph Cuts 23

3.2.2 On Convex Relaxations 27

3.2.3 Ginzburg–Landau Relaxation 29

3.3 Numerical Scheme 30

3.3.1 Merriman–Bence–Osher Iteration 30

3.3.2 Treating the Matrix Exponential . . . 31
3.3.3 Determining the Number of Communities . . . 33
3.3.4 Scaling . . . 34
3.3.5 The Choice of Timestep . . . 34
3.4 Results . . . 36
3.4.1 Summary . . . 36
3.4.2 Analysis of each experiment . . . 40

4 Stochastic block models are a discrete surface tension . . . 46
4.1 Background . . . 46
4.1.1 Stochastic Block Models (SBMs) . . . 46
4.1.2 Surface Tension . . . 50
4.2 An Equivalence Between SBM MLE and Discrete Surface Tension . . . 51
4.3 Mean-Curvature Flow (MCF), Γ-Convergence, and Threshold Dynamics . . . 54
4.3.1 Mean-Curvature Flow . . . 54
4.3.2 Allen–Cahn (AC) Evolution . . . 58
4.3.3 MBO Iteration . . . 58
4.3.4 Learning ω . . . 59
4.4 Empirical Results . . . 61

5 Conclusion . . . 66

Appendices . . . 69

A Γ-Convergence of the Ginzburg–Landau Approximations of Expressions (3.6) and (4.4) . . . 69

B Deferred proofs from Section 3.3.5 . . . . . **71**

C Hyperspectral Image Details **73**

D Eliminating the Diagonal Elements of W **74**

E Additional Notes on the AC and MBO Schemes for Expression (4.4) . . **76**

References **79**

LIST OF FIGURES

2.1 Image of the 1-norm unit ball 10

3.1 Projection of the two moons example onto two dimensions 41

3.2 The urban dataset segmented using different methods 43

3.3 Segmentations of the plume hyperspectral video using different methods . . . 44

4.1 Examples of different connectivity patterns modeled by stochastic block models 48

4.2 Example arrangement of crystals 51

4.3 Segmentation of a hyperspectral video using graph MCF 63

LIST OF TABLES

3.1 Results of modularity optimization on six networks 38

3.2 Results of modularity optimization on hyperspectral image similarity networks 39

3.3 Results of modularity optimization with and without supervision 40

4.1 Results of SBM tests 64

4.2 Computation times 65

4.3 Optimal surface tensions for the MS SBM example 65

ACKNOWLEDGMENTS

My sincerest thanks to Andrea Bertozzi, my advisor, for encouraging me to come to UCLA, for helping me win my fellowship, for constantly offering resources and ideas, and for helping me develop my own scientific style. I am also grateful to my collaborators, including Xue-Cheng Tai for helping develop numerous ideas about modularity and for inviting me to Norway, Egil Bae for sharing notes and ideas as well as many helpful edits of our paper, and Mason Porter for the idea to investigate stochastic block models as well as multiple careful readings of our paper, which greatly improved it. I was also fortunate to work with several scientists at Los Alamos National Laboratory, including Scott D. Ramsey, Roy S. Baty, Joseph Schmidt, Carl Hagelberg, Joanne Wendelberger, and the other scientists at Los Alamos National Laboratory, who devoted countless mentorship and collaboration hours to me and helped me produce great papers not featured in this dissertation. Scott in particular gave a great deal of time and energy to me, for which I am profoundly grateful. A number of others offered helpful advice at various points, notably Ekaterina Merkurjev, Chris Anderson, Rob Hannah, Brent Edmunds, Shyr-Shea Chang, and Inwon Kim, to name only a few among many. I am indebted to Andrea, Chris, Stan, and Mason for serving on my dissertation committee, as well as Fei Sha, who served on my candidacy committee. Thanks are also due to my fellow grad students and Los Alamos student collaborators, who were true friends, team mates, and supporters; and to my parents, who nurtured my desire to excel and supported my family during all my travel. Last and most importantly, special thanks are owed to my wife, Katie, without whom none of this would have been possible, and my children, Nathan, Sam, and Lucy, who motivated me to work efficiently.

Chapter 3 is a version of "Simplified energy landscape for modularity using total variation," by Zachary M. Boyd, Egil Bae, Xue-Cheng Tai, and Andrea L. Bertozzi, submitted to SIAM J. App. Math. EB provided expertise on constrained convex optimization. XCT suggested using convex balancing terms for modularity and contributed ideas for the numerics. ALB helped with the theory, examples, and numerous points of advice. All authors assisted

with manuscript preparation.

Chapter 4 is a version of "Stochastic block models are a discrete surface tension," by Zachary M. Boyd, Mason A. Porter, and Andrea L. Bertozzi, submitted to J. Nonlinear Sci. MAP provided network science expertise, and ALB provided expertise in surface tension dynamics. Both assisted in manuscript preparation.

Parts of Chapters 2 and 5 are also taken from the papers on which Chapters 3 and 4 are based.

This work was supported by the U.S. Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program. Support for travel was provided by NSF grants DMS-1417674 and DMS-1118971 as well as ISP-Matematikk (Project no. 2390033/F20) at the University of Bergen.

This material is based on research sponsored by the Air Force Research Laboratory and DARPA under agreement number FA8750-18-2-0066. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory and DARPA or the U.S. Government.

VITA

2013	B.S. (Mathematics), Brigham Young University, Provo.
2014	M.S. (Mathematics), Brigham Young University, Provo.
2014–2017	Graduate Research Assistant, Los Alamos National Laboratory, Los Alamos, New Mexico.
2015–2018	National Defense Science and Engineering Graduate Fellow, Mathematics Department, UCLA.

PUBLICATIONS

Zachary M. Boyd, Mason A. Porter, and Andrea L. Bertozzi. "Stochastic block models are a discrete surface tension." Submitted to J. Nonlinear Anal., arXiv:1806.02485.

Zachary M. Boyd, Emma M. Schmidt, Scott D. Ramsey, and Roy S. Baty. "Converging shocks and collapsing cavities in non-ideal materials." Submitted to SIAM J. Math. Anal., arXiv:1712.07561.

Zachary M. Boyd, Egil Bae, Xue-Cheng Tai, and Andrea L. Bertozzi. "Simplified energy landscape for modularity using total variation." Accepted at SIAM J. App. Math., arXiv:1707.09285.

Scott D. Ramsey, Emma M. Schmidt, Zachary M. Boyd, Jennifer F. Lillieholm, and Roy S. Baty. "Converging shock flows for a Mie-Gruneisen equation of state." Accepted at Physics of Fluids, arXiv:1712.08236.

Zachary M. Boyd and Joanne Wendelberger. "An integrated approach to parameter learning in infinite dimensional space." Los Alamos technical report LA-UR-17-28326, 2017.

Zachary M. Boyd, Scott D. Ramsey, and Roy S. Baty. "On the existence of self-similar converging shocks for arbitrary equation of state." Qu. J. Mech. and App. Math., 70(4), 401-417, 2017.

Scott D. Ramsey, Zachary M. Boyd, and Sarah Burnett. "Solution of the Noh Problem using the universal symmetry of the gas dynamics equations." Shock Waves 27(3), 477-485, 2017.

Zachary M. Boyd, Scott D. Ramsey, and Roy S. Baty. "Symmetries of the Euler compressible flow equations for general equation of state." Los Alamos technical report LA-UR-15-28034, 2015.

Zachary M. Boyd. Convolutions and convex combinations of harmonic mappings of the disk. Master's thesis, Brigham Young University, 2014.

Zachary M. Boyd, Michael Dorff, Maria Nowak, Matthew Romney, and Magdalena Woloszkiewicz. "Univalency of convolutions of harmonic mappings." Applied Mathematics and Computation, 234C, 326-332, 2014.

Zachary M. Boyd and Michael Dorff. "Harmonic univalent mappings and minimal graphs." Current topics in pure and computational complex analysis. Eds. S. Joshi, M. Dorff, and I. Lahiri. Springer, 21-46, 2014.

Zachary M. Boyd, Michael Dorff, Rachel Messick, Matthew Romney, and Ryan Viertel. "Harmonic univalent mappings with singular inner function dilatation." 60 years of analytic functions in Lublin, 191-200, 2012.

CHAPTER 1

Introduction

1.1 Network Science and Community Detection

The study of networks, in which nodes represent entities and edges encode interactions between entities [120], can provide useful insights into a wide variety of complex systems in myriad fields, such as granular materials [128], disease spreading [130], criminology [74], and more. In the study of such applications, the analysis of large data sets — from diverse sources and applications — continues to grow ever more important.

The simplest type of network is a graph, and empirical networks often appear to exhibit a complicated mixture of regular and seemingly random features [120]. Additionally, it is increasingly important to study networks with more complicated features, such as time-dependence [76], multiplexity [91], annotations [121], and connections that go beyond a pairwise paradigm [127]. One also has to worry about "features" such as missing information and false positives [88].

To try to understand the large-scale structure of a network, it can be very insightful to coarse-grain it in various ways [55, 136, 138, 141, 142]. The most popular type of clustering is the detection of assortative "communities," in which dense sets of nodes are connected sparsely to other dense sets of nodes [55, 138]. Two of the most common approaches to community detection are modularity maximization, which seeks a partition of the graph nodes optimizing a quality function, and stochastic block modeling, in which a family of generative models is fitted to an observed network. The detection of communities has given fascinating insights into a variety of applications, including brain networks [17], social networks [152],

granular networks [13], protein interaction networks [9], political networks [137], and many others.

The network partitioning problem is very ill posed. Real networks are generated by complicated processes with many factors, and thus there are often multiple ways to partition a network that reflect legitimate divisions among the objects being studied [132]. Such effects have been observed, for instance, in the Zachary Karate Club network, which has both a community structure and leader-follower structure [132]. In [69], the energy landscape of modularity was found to have many near-degeneracies, with super-exponentially many locally optima having similar objective values.

The two main results of this work are (1) an equivalence between modularity maximization and modified graph total variation minimization and (2) an equivalence between stochastic block models and surface-tension models from the literature on partial differential equations (PDEs) that model crystal growth. Section 1.2 surveys some related literature to provide motivation. We describe the main results of this dissertation in Sections 1.3 and 1.4.

1.2 Nonlinear differential equations and variational methods on networks

The study of nonlinear PDEs has a long history, with physics and other applications motivating exploration of topics ranging from well-posedness to asymptotics to numerical schemes, all of which have become fields in their own right. With the comparatively recent rise of graph-based machine-learning methods and the host of other applications of network science, there has been an interest in transferring some of the tried-and-true knowledge of PDE theory to these new applications. One of the very early ideas was to consider diffusion on graphs using a graph version of the heat equation, which has been used to model information dissemination, peer pressure, disease spread and other processes, see e.g. [37, 104]. Another prominent family of models centers on SIR equations, which model disease spread [131]. A third approach involving differential equations is gradient descent of graph energies, which is

formulated naturally as a differential equation. See e.g. [10]. Other approaches are reviewed in [149].

The approaches in this dissertation have their roots in a newer body of literature, initiated by Bertozzi and Flenner [14], who showed that semisupervised graph-cut problems arising in machine learning can be solved using a graph version of the Allen–Cahn equation, which is a nonlinear, second-order PDE arising from the gradient descent of a Ginzburg–Landau-type energy. This showed the potential of phase-field models and other nonlinear PDE and variational methods to contribute to graph-based problems, and numerous papers have expanded on this idea [15]. Soon after this, [107] introduced MBO schemes for graph segmentation, and [62] generalized Bertozzi and Bertozzi and Flenner's work to multi-way cuts. On the theory side, van Gennip and Bertozzi [155] proved Γ-convergence of graph Ginzburg–Landau functionals to a graph total variation energy, and van Gennip et al. [156] developed notions of mean curvature flow on graphs, with rigorous estimates on its relationship to previous graph dynamics. Luo and Bertozzi [101] gave additional guarantees on the graph Allen–Cahn scheme. At the same time, more applications of these ideas were developed, including community detection [79], hyperspectral image segmentation [171], balanced cuts [25], body-worn video [106], and many others. PDE and variational models are sometimes faster than the previous state of the art [14] and may have better guaranteed accuracy [108]. Further research needs to be directed toward understanding what kinds of graphs can be subjected to the various PDE methods and the various ways in which graph topology impacts the dynamics that underly most of these methods. There is also a need to develop high-quality codes that can be used by non-experts (but see [105]).

This dissertation is in the same spirit as much of the work just cited. Chapter 3 explores theoretical and practical aspects of the connection between total variation and community detection first found in [79]. Chapter 4 shows that a popular model of heterogeneous community structure is analogous to a surface-tension energy and can be treated using surface-tension inspired methods.

1.3 Modularity Optimization

In Chapter 3, we develop an equivalence between modularity optimization and graph total variation minimization. Modularity is a very popular metric which, given a partition of a network, compares the number of edges within each class to its expected value according to a null model [122]. One uses modularity for community detection by searching among all partitions for the one with the highest modularity score. Modularity can be adapted to account for a variety of features such as directed edges [98], bipartite structure [12], and spacial embedding [50]. The benefits of this approach include its statistical foundation [119], automatic selection of the number of communities, and the existence of effective algorithms and libraries (e.g. [42]).

In [79] an equivalence between modularity optimization (with fixed number of communities) and minimization of a modified graph total variation functional was developed. This allowed the application of techniques from compressed sensing, image processing, and geometric PDE to the community detection problem, yielding good results on benchmark networks. On the other hand, the model they proposed was nonconvex, whereas traditional total variation optimization is convex. This difference restricts the optimization tools and analytic estimates available and forces reliance on ad hoc initialization strategies and multiple runs of the solver to get reliable results. The possibility of obtaining a convex proxy for the model of [79] motivated much of the work in Chapter 3.

Chapter 3 shows that one of the sources of nonconvexity in [79] can be removed, resulting in a convex functional over a discrete space. Our efforts to remove the discrete constraint culminated in a theorem showing that it is impossible to reformulate modularity convexly under certain conditions. We developed a nonconvex relaxation of modularity using a new Ginzburg–Landau functional, which Gamma-converges to the modified total variation functional and thus to (negative) modularity. This proof is of interest outside of community detection since Gamma-convergence results on graphs were previously known only for binary functions [155] rather than the multiphase functions required for modularity. While

not convex, the Ginzburg-Landau relaxation has a scale parameter that allows for a trade-off between the severity of the nonconvexity and the degree of agreement with the original modularity model.[1] In order to numerically solve the Ginzburg-Landau problem, we develop a graph Merriman–Bence–Osher (MBO) scheme for modularity, which uses alternating diffusion and thresholding to approximate operator splitting. We are able to prove theoretical bounds on the associated diffusion dynamics, which we use to automate timestep selection, a major practical improvement over the algorithm in [79], especially in the case where recursive partitioning is required. We tested our approach on several networks, including one with over 29 million edges. The results on hyperspectral image networks were particularly encouraging, beating both the algorithm from [79] and the spectral bipartitioning scheme of [117, 118], as well as performing competitively with the Louvain method. Our method is also compatible with semi-supervised community detection, in which the community assignment of some nodes is known beforehand, which improved considerably the navigability of the otherwise nearly degenerate modularity energy landscape in our tests.

1.4 Stochastic Block Models (SBMs)

Chapter 4 describes the second major result, namely the connection between SBMs and surface tension models. SBMs are a generative model that can produce networks with community structure [55,136].[2] One uses an SBM for community detection by fitting an observed graph to a statistical model to attempt to infer the most probable community assignment for each node. SBMs can incorporate a variety of features, including degree heterogeneity [85], hierarchical structure [133], and metadata [121]. The benefits of an SBM approach include statistical defensibility, theoretical tractability, asymptotic consistency under certain conditions, definable transitions between solvable and unsolvable regimes, and theoretically

[1] In [79], a Ginzburg-Landau functional is also used, although in that context it does not have the interpretation of perturbing the energy for improved convexity.

[2] Networks that are generated from an SBM can also have other types of block structures, depending on the choice of parameters; see Section 4.1.1 for details.

optimal algorithms [113, 136].

Recently, Newman showed that one can interpret modularity maximization [117, 122] as a special case of an SBM [119]. This raises the possibility of formulating SBM maximum-likelihood estimation (MLE) in terms of TV. We develop such a formulation using ideas from models of surface tension and crystal growth.

The main result of Chapter 4 is the establishment of an equivalence between SBMs and surface-tension models from the literature on PDEs that model crystal growth. Crystal growth is an important aspect of certain annealing processes in metallurgy [90, 115]. It is a consolidation process, wherein the many crystals in a metal grow and absorb each other to reduce the surface-tension energy that is associated to the interfaces between them. The various processes involved have been modeled from many perspectives, including molecular dynamics [40], front tracking [60], vertex models [163], and many others. (See [90] for a much more extensive set of references.) It has been observed experimentally that the interface between any two grains evolves according to motion by mean curvature [147]. Although the interfaces follow mean-curvature flow, each different interface can evolve at a different rate, as there are different surface-tension densities between each pair of crystals. In realistic cases, surface tensions are both inhomogeneous and anisotropic, and they require careful adaptation of standard mean-curvature-flow approaches [47, 81], especially for dealing with the topological challenges that arise at crystal junctions, which routinely form and disappear.

Because mean-curvature flow is the gradient descent of the TV energy, this leads naturally to formulations in terms of level sets [126], phase fields [19], and threshold dynamics [111].

Each community in a network is analogous to a crystal, and the set of edges between nodes from a pair of communities is akin to the topological boundary between a pair of crystals. The surface-tension densities correspond to the differing affinities between each pair of communities. To demonstrate the relevance of this viewpoint, we develop and test discrete analogs of surface-tension numerical schemes on several real and synthetic networks, and we find that straightforward analogs of the continuum techniques successfully recover planted community structure in synthetic networks and uncover meaningful structure in

the real networks. We also prove a theoretical result, in terms of Γ-convergence, that one can meaningfully approximate the SBM MLE problem by smoother energies. Finally, we introduce three algorithms — inspired by work on crystal growth — that we test on synthetic and real-world networks.

The paper on which Chapter 4 is based appears to be the first connection between network community detection and surface tension problems of this kind. Recently, Jacobs showed how to apply techniques from models of crystal growth to graph-cut problems from semisupervised learning [81]. (See also [82] for related work.) We also note that several recent papers, which do not directly involve surface tension, have used ideas from perimeter minimization and/or TV minimization for graph cuts and clustering in machine learning [15]. Three of those papers are concerned explicitly with ideas from network science [20, 79, 154].

CHAPTER 2

Background

In this chapter, we review topics which are relevant to both Chapters 3 and 4. Background topics not needed in both Chapters 3 and 4 are included at the beginning of the chapter to which they are relevant. Sections 2.2 to 2.4 develop continuum and graph versions of total variation, Ginzburg–Landau functionals, and MBO schemes. Section 2.5 introduces nonlocal means and similarity graphs, a framework used to convert machine learning problems into graph theory problems to which community detection methods can be applied.

2.1 Notation and Definitions

We use the following notation throughout this dissertation: Let G be an undirected, sparse graph with N nodes, weighted adjacency matrix A, degree vector k satisfying $k_i = \sum_j A_{ij}$, and $2m = \sum_i k_i$. We treat k as a column vector. In Chapter 3, G is non-negatively weighted, and in Chapter 4, it is unweighted. We also assume that G is sparse, meaning that the number of edges is $O(N)$ rather than $O(N^2)$. None of the theory we develop relies on the sparsity assumption, but our algorithms do require it in order to be efficient — for example, we assume that the map $U \to AU$ for a vector U can be computed in $O(N)$ time.

Definition 1. *Let B_α be a subset of the nodes of G for each $\alpha \in \{1, \dots, \hat{n}\}$. Then $B_1, \dots, B_{\hat{n}}$ is a* partition *of the nodes of G if $\bigcup_{\alpha=1}^{\hat{n}} B_\alpha$ includes all nodes in G and $B_{\alpha_1} \cap B_{\alpha_2}$ is empty for each $\alpha_1 \neq \alpha_2$.*

Throughout this dissertation, we will use two other representations of a partition of graph

nodes: community assignment vectors and partition matrices. Given a partition $B_1, \ldots, B_{\hat{n}}$, the associated community assignment vector $g \in \mathbb{R}^N$ satisfies $g_i = \alpha$ iff node i is in B_α. The equivalent partition matrix is defined as follows.

Definition 2. *Let $\Pi(G)$ be the set of all partitions of the nodes of G. For each partition $B_1, \ldots, B_{\hat{n}}$ in $\Pi(G)$, there is an $N \times \hat{n}$* partition matrix *U defined by,*

$$U_{i\alpha} = \begin{cases} 1 & i \in B_\alpha \\ 0 & i \in B_\alpha^c \end{cases}$$

For a matrix U, we say $U \in \Pi(G)$ when U is the partition matrix of some partition.

2.2 Total Variation

We briefly introduce total variation (TV) and why it is interesting to establish a connection between community detection frameworks and TV. Consider a smooth function $f : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ for some d. The TV of f is

$$|f|_{\mathrm{TV}} = \int_\Omega |\nabla f| \mathrm{d}x \,. \tag{2.1}$$

For $d = 1$, (2.1) describes the total amount of increase and decrease of the function f. If f is smooth except for jump discontinuities, one can interpret the derivative of f in a generalized sense, yielding

$$|f|_{\mathrm{TV}} = \int_{\mathbb{R}^d - \Gamma} |\nabla f| \mathrm{d}x + \int_\Gamma |[f]| \, \mathrm{d}x \,,$$

where Γ is the union of all curves of discontinuity and $[f]$ is the height of the jump across the discontinuity. The first integral uses a d-dimensional measure, and the second uses a $(d-1)$-dimensional Hausdorff measure. In the particular case in which $d = 2$ and f is the characteristic function of some set S, we see that $|f|_{\mathrm{TV}}$ is the perimeter of S. Similarly, when $d = 3$, we obtain surface area.

Total variation is an important regularizer in machine learning. It is worth contrasting it with the Dirichlet energy $\displaystyle\int_\Omega |\nabla f|^2 \mathrm{d}x$, which has minimizers that satisfy $\Delta f = 0$, a condition

that guarantees smoothness. However, minimizers of TV need not be smooth, as they can admit jump discontinuities. In image denoising, for instance, regularization using Dirichlet energy tends to blur edges to remove discontinuities, whereas a TV regularizer leaves the edges intact [28, 144].

Another use of TV energy is in relaxations, in which one can transform a nonconvex problem involving piecewise-constant constraints into a convex problem with the same minimizers [28, 108]. A common heuristic explanation for this phenomenon (see Fig. 2.1) uses the shape of the 1-norm unit ball. The simplest case is in two dimensions, where the 1-norm ball is diamond-shaped, and minimizing the 1-norm over certain domains (e.g., a line) gives a sparse solution, in the sense that most components of the solution vector are 0. In this case, minimizing the 1-norm, constrained to a line, is the same as minimizing the number of nonzero elements of the vector, subject to the same constraint.

In the context of TV minimization, we take the 1-norm of a function's gradient, rather than of the function itself. Thus, instead of promoting sparsity of the function values, we promote sparse gradients, thereby incentivizing piecewise-constant minimizers for TV. Our discussion above is heuristic, but the ideas therein can be treated rigorously [28].

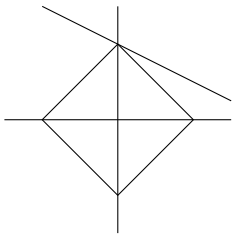


Figure 2.1: Image of the 1-norm unit ball and a line in the plane. The point on the line with the smallest 1-norm is almost always on one of the axes.

Algorithmically, one can minimize TV using, for example, phase-field models [19] or threshold dynamics [111], both of which rely on the fact that the gradient descent of TV is a generalization of mean-curvature flow. The alternating-directions method of multipliers (ADMM) [68] and graph-cut methods, such as the one in [22], also solve similar problems

effectively.

Thus far, we have restricted our discussion of TV to a continuum setting. There are graph analogs of the mathematical objects — gradients, measures, integrals, tangent spaces, divergences, and so on — that one uses to define TV in a continuum setting. For instance, for any function f on the nodes of a graph and any edge between nodes i and j, the discrete derivative at i in the direction j is

$$\nabla f(i,j) = f(j) - f(j)\,.$$

Using the inner products

$$\langle f, g\rangle = \sum_{i=1}^{N} f_i g_i\,,$$
$$\langle \phi, \psi\rangle = \sum_{i,j} A_{ij}\phi_{ij}\psi ij$$

on the spaces of functions on the nodes and edges, respectively, gives the divergence as the adjoint of the gradient:

$$(\operatorname{div}\phi)_i = \sum_j A_{ij}\phi_{ji}\,.$$

In a continuum, an alternative definition of TV is

$$|f|_{\mathrm{TV}} = \sup\langle \operatorname{div}\phi, f\rangle\,, \tag{2.2}$$

where the supremum is over an appropriate set of test functions. For a graph, (2.2) is equivalent to

$$|f|_{\mathrm{TV}} = \frac{1}{2}\sum_{i,j} A_{ij}|f(i) - f(j)|\,. \tag{2.3}$$

See [65, 156] for a detailed justification of these definitions. We will also use a slight generalization of (2.3) to the case where $f : \{1,\ldots,N\} \to \mathbb{R}^{\hat{n}}$ is vector-valued, in which case

$$|f|_{TV} = \sum_{\alpha} |f_\alpha|_{TV}$$

where f_α is the α-th component of f. It is sometimes convenient in this case to identify f with an $N \times \hat{n}$ matrix U where $U_{i\alpha} = f_\alpha(i)$. Then we have

$$|U|_{TV} = \sum_{\alpha}\frac{1}{2}\sum_{i,j} A_{ij}|U_{i\alpha} - U_{j\alpha}|.$$

In particular, Section 2.2 allows one to take evaluate the total variation of the matrix of a partition.

Some methods for graph clustering (e.g., see [160]) rely on the combinatorial (or unnormalized) graph Laplacian

$$L = \mathrm{diag}(k) - A, \tag{2.4}$$

which is a discrete analog of the continuum Laplacian Δ. The continuum Laplacian arises in solutions to constrained optimization problems that involve the Dirichlet energy, so it is reasonable to expect minimizers of energies that involve the graph Laplacian to have analogous properties to minimizers of the Dirichlet energy. Indeed, it is well-known that the minimizers that arise from spectral methods are usually smooth, instead of having sharp interfaces, so one needs to threshold them in some way. Such thresholding is a major source of difficulties for attempts to obtain theoretical guarantees about the nature of minimizers after thresholding. In contrast, methods that use graph TV can directly accommodate piecewise-constant solutions [108], which do not require thresholding to give classification information. Several previous papers have exploited this property of TV on graphs [14, 79, 153, 171].

Graph total variation is connected to graph cuts, which correspond roughly to perimeter in Euclidean space.

Definition 3. *Let B be a subset of the nodes of G. Then the graph cut*[1] *associated to B is given by*

$$\mathrm{Cut}(B, B^c) = \sum_{i \in B, j \in B^c} A_{ij}.$$

Let $f : \{1, \ldots, N\} \to \mathbb{R}$ be the characteristic function of a set of nodes B. Then we can calculate

$$|f|_{TV} = \frac{1}{2} \sum_{i,j} A_{ij} |f(i) - f(j)| = \sum_{i \in B, j \in B^c} A_{ij} = \mathrm{Cut}(B, B^c). \tag{2.5}$$

TV minimization on a graph tends to produce piecewise-constant functions whose corresponding graph cut is small [108].

[1] In Chapter 4, we use a slightly different notation for graph cuts that accommodates the additional

2.3 The Ginzburg–Landau (GL) Functional

One approach which minimizes total variation by approximating mean-curvature flow is approximation by a Ginzburg–Landau functional. This approach is popular due both to its simple implementation and the existence of unconditionally-stable numerical methods [14]. The idea is to use a *phase-field* u. In the two-phase case, an example GL functional is

$$\int_\Omega \left[\epsilon \left| \nabla u \right|^2 + \frac{1}{2\epsilon} u^2 (1-u)^2 \right] \mathrm{d}x \,, \tag{2.6}$$

where $u : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ is a smooth function and ϵ is a small parameter. The L^2 gradient descent of the GL functional is

$$u_t = \epsilon \Delta u - \frac{1}{2\epsilon} \frac{\mathrm{d}}{\mathrm{d}u} \left[u^2 (1-u)^2 \right] \,,$$

which is the Allen–Cahn (AC) equation. The minimizers of the GL energy are mostly constant, with $O(\epsilon)$-width transition layers between the constant regions. Further, one can show that the GL energy Γ-converges to the TV energy as $\epsilon \to 0$, assuming that $\int_\Omega u \, \mathrm{d}x = \text{const}$ [112]. Consequently, if u_ϵ is a minimizer of the constrained GL energy with parameter ϵ and the minimizers converge in bounded-variation (BV) space as $\epsilon \to 0$, then the accumulation point is a minimizer of the TV energy.

In order to derive the graph Ginzburg–Landau functional, observe that if we ignore boundary terms, then integration by parts gives

$$\int_\Omega \|\nabla u\|_2^2 = \int_\Omega \nabla u \cdot \nabla u \mathrm{d}x = \int_\Omega - \operatorname{div} \nabla u \cdot u \mathrm{d}x = \int_\Omega -\Delta u \cdot u, \mathrm{d}x \tag{2.7}$$

which suggests that we use a graph Laplacian in our formulation. The Laplacian that is appropriate for our context is (2.4). Following this hint, as well as previous works, notably [14, 155], we obtain the following graph GL functional.[2]

$$\mathcal{F}_\epsilon = \|\nabla U\|_2^2 + \frac{1}{\epsilon} \sum_{i=1}^N \Psi(U(i,:)) + G(U) \tag{2.8}$$

complications of the problem considered in that chapter. See Section 4.2 for details. The definition of volume is correspondingly adjusted as well.

[2]Note that due to the discrete setting, there is no epsilon factor preceding the Laplacian term, see [155].

$$:= U^T L U + \frac{1}{\epsilon} \sum_{i=1}^{N} \Psi(U(i,:)) + G(U), \tag{2.9}$$

where U is an $N \times \hat{n}$ matrix, G is an application-specific potential, and Ψ is a multi-well potential on $\mathbb{R}^{\hat{n}}$ that is small for arguments with exactly one nonzero entry. For example, [62] found that the following potential works well:

$$\Psi(U) = \left(\prod_{\alpha=1}^{\hat{n}} \frac{1}{4} \left\| U(i,:) - e_\alpha \right\|_{\ell^1}^2 \right),$$

where e_α is an $\hat{n}$-element vector that is equal to 0 except for a 1 in the α-th entry. In Theorem 2 and Section 4.3, we show that (2.9) Γ-converges to graph total variation (plus the potential G), and in Chapter 4 we develop a more complicated GL functional that is suitable for a stochastic block model problem and also show Γ-convergence in this case, as well as showing how to optimize it numerically.

2.3.1 Choice of Graph Laplacian

The graph TV formulation we are working with has led us to use the unnormalized Laplacian (2.4). In practice, three different Laplacians are commonly used, with the other two being the random walk Laplacian, $I - D^{-1}W$, and the symmetric normalized Laplacian, $I - D^{-1/2} W D^{-1/2}$, where $D = \operatorname{diag} k$. In many applications, normalization is used to reduce the influence of connections to high-degree nodes, which can dominate the dynamics of the unnormalized Laplacian, leading to issues such as eigenvector localization (e.g. [129]). This also leads to issues with consistency of the operator [160] and can lead to visually unappealing clusterings [14]. On the other hand, this is the Laplacian which naturally arises when using statistically principled approaches such as modularity optimization and stochastic block models, which are developed in Chapters 3 and 4. The network science community has dealt with the problems of normalization by imposing degree-corrected null models rather than the simplest Erdős–Renyi models, see Chapter 3. Nonetheless, it is important to be aware of these issues when writing algorithms, since naive approaches based on the spectrum of the unnormalized Laplacian can have unexpected results if the degree correction terms are not

properly incorporated. This will, for instance, influence our decisions when implementing the pseudospectral scheme in Chapter 3.

2.4 Merriman–Bence–Osher Schemes

In [111], Merriman, Bence, and Osher showed that continuum mean-curvature flow is well-approximated by the simple iteration in Algorithm 1. In a rectangular domain, the iteration is extremely efficient, as one can use a fast Fourier transform when solving the heat equation. It also gracefully handles topological changes that routinely occur in mean curvature flow. Another approach to understanding MBO schemes goes through operator splitting in the

Algorithm 1 A two-phase, continuum MBO scheme.

Input the initial domain.

Initialize u as the characteristic function of the initial domain.

for $n = 1, 2, \ldots$ **do**

$u^{n+1/2}$ is the solution at time dt of $u_t = \Delta u$ with initial condition u^n.

$u^{n+1} = \lfloor u^{n+1/2} + 0.5 \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function.

end for

Output the set of points for which $u = 1$.

Allen–Cahn equation. Since the theory of mean curvature flow on graphs and its connection to MBO schemes is still developing [156], we will use the Allen–Cahn approach to show how graph MBO works.

We minimize the functional from (2.9) where ϵ is small. The connection between graph-based TV and MBO was first made in [107] and [62]. Consider the Ginzburg–Landau gradient descent equation (at fixed $\hat{n}$)

$$\frac{d}{dt}U = -LU - \frac{1}{\epsilon}\nabla\left(\sum_i \Psi(U(i,:))\right) - \nabla G(U),$$

where ∇ denotes the gradient in the Frobenius inner product. One way to approximate this flow is by operator splitting [66, p.22] with time-step dt and $t_n = ndt, n = 0, 1, 2, \ldots$. Given

U^n one obtains $U^{n+\frac{1}{2}}$ as the solution to

$$\begin{aligned} \frac{d}{dt}U_1 &= -LU_1 - \nabla G(U_1), \quad t \in [t_n, t_{n+1}], \\ U_1(t_n) &= U^n, U^{n+1/2} = U_1(t_{n+1}). \end{aligned} \tag{2.10}$$

Then one gets U^{n+1} by solving

$$\begin{aligned} \frac{d}{dt}U_2 &= -\frac{1}{\epsilon}\nabla\left(\sum_i \Psi(U_2(i,:))\right), \quad t \in [t_n, t_{n+1}], \\ U_2(t_n) &= U^{n+1/2}, U^{n+1} = U_2(t_{n+1}). \end{aligned} \tag{2.11}$$

The iteration continues until a fixed point is reached. Such operator splitting schemes are typically first-order accurate in time. Expression (2.11) is essentially a thresholding operation, pushing all values of U into the nearest well, i.e.

$$U_{i\alpha}^{n+1} = \begin{cases} 1 & \alpha = \underset{\beta}{\operatorname{argmax}}\, U_{i\beta}^{n+\frac{1}{2}} \\ 0 & \text{otherwise} \end{cases}$$

Thus, the graph MBO scheme, like the continuum version, consists of alternating diffusion and thresholding. See Algorithm 2 for an example of pseudocode.

The most expensive part of this procedure is solving the diffusion equation. We accomplish this efficiently using a pseudospectral scheme, which will be described in Section 3.3.2 and Appendix E.

2.5 Similarity Graphs, Hyperspectral Video, and Nonlocal Means

In some of the data sets used in this dissertation, the information does not come in a network format, but can be transformed into a graph. A typical situation is when one wishes to classify objects into groups whose members are similar to each other, either in an unsupervised, supervised, or semisupervised setting. In this case, one wishes to identify the objects with graph nodes, where (possibly weighted) graph edges encode similarity between pairs of objects. One then attempts to cluster the graph nodes in such a way that similar nodes reside in the same cluster, a notion which can be made precise in various ways, including

modularity optimization, balanced cuts [87], conductance [36], etc. The details of graph construction can have a profound impact on the success of graph-based methods, since if the relevant relationships between objects are not successfully encoded into the graph structure, no graph clustering method can be expected to extract the desired structure. (In Section 3.4, we will see with the MNIST example that a naive similarity graph construction leads to issues where the graph clustering algorithm is effective but extracts the wrong information. This is the only example where we will see such issues, however.)

A typical graph construction procedure might proceed as follows: (1) feature vectors are extracted from each object, (2) a similarity score is computed between each pair of feature vectors, and (3) fully-connected graph is constructed, where the weight of the edge between two distinct nodes is the similarity score. (We do not allow self-edges.) For step one, feature vectors can be hand-constructed, computed using machine-learning approaches, such as principle components analysis, or computed using more complicated methods, such as neural networks. The similarity score is often given by some decreasing function of the distance between feature vectors, for example

$$e^{\frac{-d(x_i,x_j)^2}{\sigma}}, \tag{2.12}$$

where x_i is the feature vector corresponding to node i and $\sigma > 0$. In this case, the selection of an appropriate distance is important, with options including Euclidean distance

$$d(x_i, x_j)^2 = \sum_\alpha (x_{i\alpha} - x_{j\alpha})^2,$$

cosine similarity

$$d(i,j) = \frac{\sum_\alpha x_{i\alpha} x_{j\alpha}}{\|x_i\| \, \|x_j\|},$$

and ℓ^1 distance

$$d(i,j) = \sum_\alpha \sum_\alpha |x_{i\alpha} - x_{j\alpha}|.$$

Note that cosine similarity is not technically a metric since it does not obey the triangle inequality, but it is useful in situations where the sign and scale of the vectors should be ignored. The ℓ^1 metric is more robust to noisy dimensions than the Euclidean distance, but

the Euclidean distance is rotation-invariant. Each of these distances has a weighed version, for instance

$$d(x_i, x_j)^2 = \sum_\alpha w_\alpha (x_{i\alpha} - x_{j\alpha})^2$$

for non-negative constants w_α. The choice of w is a non-trivial problem that can have a large impact on performance, and its solution in practice tends to rely more on domain knowledge and ad hoc engineering than generalizable theoretical considerations. Another issue to consider is the choice of σ in (2.12). It can be hand-tuned, or a self-tuning approach can be used where σ varies from edge to edge and is given by $\sqrt{\tau_i \tau_j}$, where i and j index the nodes at each end of the edge, and τ_i is the distance from node i to its κ-th nearest neighbor for some value of κ [169].

When the number of objects, N, is large, it can be cumbersome to compute and store N^2 similarity scores. One solution is to use an interpolation technique such as the Nyström extension to approximate information about the graph, such as a few leading eigenvectors of a graph Laplacian, without fully computing or storing the graph [14, 57, 58]. In this dissertation, we take a different approach, since we will want more detailed information than just a few eigenvectors. Instead, we sparsify the similarity graph by thresholding to 0 all values of the weighted adjacency matrix except for the largest κ entries, where κ is often 10. The adjacency matrix can then be symmetrized by setting $A = \max(A, A^T)$. This fixes the issue of requiring quadratic storage, since such a graph can be stored as an adjacency list in $O(\kappa N)$ space. It is also possible to approximate the κ-nearest neighbor lists in $O(N \log N)$ time using heuristics. In this dissertation, we use the VLFeat library [157] to achieve this. Finally, for the sake of simplicity, we sometimes remove the weights from the sparsified graph by setting all non-zero entries of the adjacency matrix to 1.

The methods described in Chapters 3 and 4 are suitable for arbitrary graphs, but the methods in Chapter 3 work particularly well on similarity graphs. One reason may be that similarity networks are particularly "spatial." The triangle inequality restricts the possible distances among any three feature vectors. This is reflected in the fully connected similarity matrix because if $A_{i\ell}$ and $A_{\ell j}$ are both large, then A_{ij} will also be fairly large. This is,

of course, not true of arbitrary, non-similarity graphs. Since our methods are inspired by methods known to be effective on discretizations of Euclidean space and other manifolds, our method's particularly good performance on similarity graphs is reasonable.

Some of our examples focus on pixel classification involving hyperspectral images or videos. A hyperspectral video is different from an RGB video, in that each pixel in the former encodes the intensity of light at a large number (e.g., 129, in this case of the plumes example from Chapters 3 and 4) of different wavelengths rather than only 3. The extra information encoded in hyperspectral images and videos can be used to identify physical properties of the objects being observed, which has applications in finding new oil fields, tracking the movement of dangerous chemicals, and monitoring the development of crops [70]. We consider the classification problem of identifying pixels that include similar materials (such as dirt, road, grass, and so on). This problem is difficult because in practice, a point in space can hold a mixture of materials, and boundaries tend to be diffuse, irregular, and ill-defined. Furthermore, the signal is sometimes faint and spread out among many wavelengths. We construct a graph representation of this video using "nonlocal means," feature vectors, as described in [27]. The advantage of this construction is that it captures similarity in local texture that may not be apparent at the level of a single pixel. Specifically, we use the following construction. For each pixel p and in each of 7 frames, we construct a vector v_p by concatenating the data in a 3×3 window that is centered at p. We then use a weighted cosine distance as a similarity measure on these $(3 \times 3 \times 129)$-component vectors, where we give the most weight to the components from the center of the window. We weight the center pixel components by 1, the components from adjacent pixels by 0.5, and the components from corner pixels by 0.25. That is, we let $v_{\hat{i}\hat{j}}$ be the 129-element vector at pixel $(\hat{i}, \hat{j})$, and we define $w_{\hat{i}\hat{j}}$ as the concatenation of $v_{\hat{i}\hat{j}}$, $.5v_{\hat{i}+1,\hat{j}}$, $.5v_{\hat{i}-1,\hat{j}}$, $.5v_{\hat{i},\hat{j}+1}$, $.5v_{\hat{i},\hat{j}-1}$, $.25v_{\hat{i}+1,\hat{j}+1}$, $.25v_{\hat{i}+1,\hat{j}-1}$, $.25v_{\hat{i}-1,\hat{j}+1}$, and $.25v_{\hat{i}-1,\hat{j}-1}$. We then calculate the cosine similarity between each pair of $w_{\hat{i}\hat{j}}$ vectors. Finally, using the VLFeat software package [157], we build a 10-nearest-neighbor graph using the similarity measure and a κ-dimensional tree (with $\kappa = 10$).

2.6 Graph models, NP-completeness, and theoretical guarantees

When working on graphs, many important problems are NP-complete, for example, the modularity optimization problem in Chapter 3 [24]. This implies that (assuming $P \neq NP$) any efficient (i.e. polynomial-time) algorithm yields suboptimal partitions on many graphs. Nonetheless, these algorithms often perform well in practical applications. One explanation is that the modularity energy landscape for typical real-world networks has many near-degeneracies [69], with super-exponentially many near-optimal clusterings that may be useful solutions for many applications. Another explanation for the success of many algorithms is that the graphs seen in applications are not uniformly sampled from the set of arbitrary graphs of a given size and that such "real life" graphs present an easier family of graphs to optimize on than the family of arbitrary graphs in a given size range. Indeed, there is a vast literature on different types of graph models known to produce realistic graphs for particular applications [150]. Examples include similarity graphs, stochastic block models (see Chapter 4), and Barabási–Albert scale-free graphs [11]. The case of stochastic block models is particularly relevant, as it is conjectured that there is a "computability phase transition," for which, as one varies the parameters of the generative model, the SBM optimization problem suddenly becomes NP-hard [113]. A related result is the No Free Lunch Theorem from [132], which implies that no single algorithm can be optimal for all community-detection tasks. An important and underdeveloped area of research is creating frameworks that will enable modelers and algorithm designers understand when various algorithms are appropriate for the data they want to deal with. Important results in this direction include [26, 80, 83].

CHAPTER 3

Modularity optimization and total variation

3.1 Introduction

The *modularity* of a partition $B_1, \ldots, B_{\hat{n}}$ of the nodes of G is given by

$$Q = \frac{1}{2m} \sum_{\alpha} \sum_{i,j \in B_\alpha} \left(A_{ij} - \gamma \frac{k_i k_j}{2m} \right), \tag{3.1}$$

where γ is a parameter whose role is explained later in this section. The *modularity optimization problem* seeks an partition of the nodes that maximizes (3.1) for a given graph and fixed $\gamma > 0$. Intuitively, we are to understand A_{ij} as the observed edge weight and $\frac{k_i k_j}{2m}$ as the expected weight if the edges had been placed at random according to a certain null model [122]. Thus, when maximizing (3.1), there is an incentive to group those nodes that are connected — or, more precisely, those nodes that have an unusually high-weight connection compared to the null model.

Modularity optimization is a very popular heuristic for uncovering community structure. However, the results of modularity optimization must be interpreted carefully. For example, maximizing (3.1) will yield non-trivial partitions even in a Erdős–Rényi graph [71], which indicates that overfitting is a danger. In addition, many dissimilar partitions may yield near-optimal modularity values on a given network [69, 132], indicating that the model may be under-specified or at least has more than one near-optimal solution. This is to be expected, due to the ill-posed nature of community detection in general. One way to leverage this diversity of high-modularity partitions in practice, as well as prevent the discovery of communities in random graphs, relies on consensus clustering [170]. Another approach is simply to sample many high-modularity partitions, expecting that multiple intuitively-meaningful

partitions may be found. Such effects have been observed, for instance, in the Zachary Karate Club network, which has both a community structure and "leaders and followers" (role-based) structure [132].

It has been shown [119, 143] that modularity optimization is equivalent to a special case of likelihood maximization on SBM models, which means that it is just as principled as an SBM approach as long as one accepts the extra assumptions that reduce SBM to modularity. These assumptions are that the communities are assortative and statistically similar to each other, meaning that the probability of intra- (or inter-) community edges forming is the same for all communities.

Modularity also has preferred scale for communities [54,95]. For this reason, one typically includes a resolution parameter $\gamma > 0$ [7, 139]. When γ is nearly 0, the incentive is to place many nodes in the same community, so that the edge weight is included in the sum. When γ is large, few nodes are placed in each community, to avoid including the large penalty term $\gamma \frac{k_i k_j}{2m}$.

A number of heuristics have been proposed to optimize modularity [53, 55], with prominent approaches including spectral [117, 118], simulated annealing [71], and Louvain algorithms [18]. Modularity optimization can also be interpreted in terms of force-directed layout and optimized using visualization techniques [124]. Optimizing (3.1) for a given graph with $\gamma = 1$ is NP-hard [24], which means (assuming $P \neq NP$) that any given polynomial-time heuristic will fail to optimize (3.1) for many graphs.

The rest of the chapter is organized as follows: Section 3.2 develops the main theoretical results optimizing (3.1). Section 3.3 develops the theory and practical implementation of our algorithm, Balanced TV. Section 3.4 gives numerical examples. Appendices A to C contain additional background and deferred proofs.

3.2 Equivalence Theorem and Its Consequences

In this section, we derive four equivalent formulations the modularity optimization problem, (3.1), and explore some consequences of these formulations. We will need this definition:

Definition 4. *For any subset B of the nodes of G, its* volume *is given by* $\operatorname{vol} B = \sum_{i \in B} k_i$.

3.2.1 Formulations of Modularity in Terms of TV and Graph Cuts

Proposition 1 presents the different formulations of modularity that form the basis for our subsequent analysis. (See Section 2.1 for notation.)

Proposition 1 (Equivalent forms of modularity)**.** *The following optimization problems all have the same solution set:*

$$\textit{Modularity:} \quad \underset{\hat{n}\in\mathbb{N},\{B_\alpha\}_{\alpha=1}^{\hat{n}}\in\Pi(G)}{\operatorname{argmax}} \quad \sum_\alpha \sum_{i,j\in B_\alpha} \left(A_{ij} - \gamma \frac{k_i k_j}{2m}\right) \tag{3.2}$$

$$\textit{Balanced cut (I):} \quad \underset{\hat{n}\in\mathbb{N},\{B_\alpha\}_{\alpha=1}^{\hat{n}}\in\Pi(G)}{\operatorname{argmin}} \quad \sum_\alpha \left(\operatorname{Cut}\left(B_\alpha, B_\alpha^c\right) + \frac{\gamma}{2m}\left(\operatorname{vol} B_\alpha\right)^2\right) \tag{3.3}$$

$$\textit{Balanced cut (II):} \quad \underset{\hat{n}\in\mathbb{N},\{B_\alpha\}_{\alpha=1}^{\hat{n}}\in\Pi(G)}{\operatorname{argmin}} \quad \sum_\alpha \left(\operatorname{Cut}\left(B_\alpha, B_\alpha^c\right) + \frac{\gamma}{2m}\left(\operatorname{vol} B_\alpha - \frac{2m}{\hat{n}}\right)^2\right) + \gamma\frac{2m}{\hat{n}} \tag{3.4}$$

$$\textit{Balanced TV (I):} \quad \underset{\hat{n}\in\mathbb{N},U\in\Pi(G)}{\operatorname{argmin}} \quad |U|_{TV} + \frac{\gamma}{2m}\left\|k^T U\right\|_2^2 \tag{3.5}$$

$$\textit{Balanced TV (II):} \quad \underset{\hat{n}\in\mathbb{N},U\in\Pi(G)}{\operatorname{argmin}} \quad |U|_{TV} + \frac{\gamma}{2m}\left\|k^T U - \frac{2m}{\hat{n}}\right\|_2^2 + \gamma\frac{2m}{\hat{n}} \tag{3.6}$$

Each of the preceding forms has a different interpretation. Expression (3.2) is based on comparison with a statistical model and views communities as regions that are more connected than they would be if edges were random. The cut formulations represent modularity as favoring sparsely interconnected regions with balanced volumes, and the TV formulation seeks a piecewise-constant partition function U whose discontinuities have small perimeter,

together with a balance-inducing quadratic penalty. The cut and TV forms come in pairs. The first form (labelled "I") is simpler to write but harder to interpret, while the second (labelled "II") has more terms, but the nature of the balance term is easier to understand, as it is minimized (for fixed $\hat{n}$) when each community has volume $2m/\hat{n}$. Furthermore, the third term of the forms labelled II reveals that the incentive to increase the number, $\hat{n}$, of communities can be quantified in terms of an $O(\hat{n}^{-1})$ penalty term, which is not obvious from other formulations of modularity of which we are aware.

One can compare these equivalent formulations with [79], in which minimizing the functional

$$|U|_{TV} - \gamma \left\| U - \text{mean}(U) \right\|^2_{\ell^2(G)} = |U|_{TV} - \gamma \sum_{i\alpha} k_i \left(U_{i\alpha} - \frac{1}{2m} \sum_{j=1}^{N} k_j U_{j\alpha} \right)^2 \tag{3.7}$$

is shown to be equivalent to modularity optimization, subject to the same constraint as the other TV formulas presented here. Thus, in [79], there are two sources of nonconvexity, namely the balance term and the constraint, while in our formulation, the discrete constraint is the only source of nonconvexity.[1] It is clear from (3.6) which features of a solution are incentivized by modularity optimization, namely, the two priorities of having a small graph cut and balanced class sizes are the only considerations. The relative weight of these considerations, as well as the number of communities, is governed by γ, via the second and third terms of (3.6). The rest of the chapter shows how these theoretical simplifications make the nonconvexity of the problem easier to navigate.

We note that forms similar to (3.3) to (3.6) have appeared in the literature before (see e.g. [139]), although the only previous work to consider any modularity formula in terms of total variation is [79].[2] To the best of our knowledge, the decomposition of modularity

[1] To see rigorously that (3.7) is nonconvex, consider the special case of two nodes connected by a single edge, $\gamma = 1$ and $U = [c\ 0; 0\ 0]$. Then considering (3.7) as a function of c immediately shows the nonconvexity. The nonconvexity is actually very general; computing the second derivative of the second term in (3.7) with respect to any component of U gives a negative value for any connected graph with more than one node. Since the TV term grows asymptotically linearly, it is eventually dominated by the quadratic growth of the second, concave term.

[2] Another very recent paper [154] used total variation for maximizing modularity, although it was not phrased primarily in those terms.

into the three intuitively meaningful terms in the forms labelled II is also novel. We will see shortly that the TV perspective on (3.3) to (3.6), combined with the convexity of the functionals in (3.5) and (3.6) leads to a number of new developments.

Expressions (3.3) to (3.6) provide a convenient way to incorporate metadata into the partitioning process.[3] This can be done by simply incorporating a fidelity term and minimizing the functional

$$|U|_{TV} + \frac{\gamma}{2m}\left\|k^T U\right\|_2^2 + \mu\left\|\chi \odot (U - f)\right\| \tag{3.8}$$

where $\mu > 0$ is a parameter, f is a term containing the metadata labels, $\odot$ is the entry-wise matrix product, χ is a matrix satisfying

$$\chi_{i\alpha} = \begin{cases} 1 & \text{label } i \text{ is known} \\ 0 & \text{otherwise} \end{cases},$$

and $||\cdot||$ is some application-appropriate norm. Including metadata should always be done with care, of course, but the general utility of semisupervised learning is well-attested in image processing and machine-learning applications. (See e.g. [14]. See also Table 3.3 for two numerical examples.)

Proof of Proposition 1. Notice that the cut and TV formulations are really just a change of notation. That leaves two nontrivial equivalences, namely the equivalence of (3.2) with (3.3) and the equivalence of (3.3) and (3.4). We first show the equivalence of (3.2) with (3.3). Fix $\hat{n}$, and consider an otherwise arbitrary partition $\{B_1, \ldots, B_{\hat{n}}\}$ of the nodes of G. Then we have

$$Q = \frac{1}{2m}\sum_{\alpha}\sum_{i,j\in B_\alpha}\left(A_{ij} - \gamma\frac{k_i k_j}{2m}\right) \tag{3.9}$$

$$= \frac{1}{2m}\sum_{\alpha}\left(\sum_{i\in B_\alpha, j\in\{1,\ldots,N\}} A_{ij} - \sum_{i\in B_\alpha, j\in B_\alpha^c} A_{ij}\right) - \frac{\gamma}{2m}\sum_{\alpha}\sum_{i,j\in B_\alpha}\frac{k_i k_j}{2m} \tag{3.10}$$

$$= \frac{1}{2m}\sum_{i,j} A_{ij} - \frac{1}{2m}\sum_{\alpha}\sum_{i\in B_\alpha, j\in B_\alpha^c} A_{ij} - \frac{\gamma}{2m}\sum_{\alpha}\sum_{i,j\in B_\alpha}\frac{k_i k_j}{2m} \tag{3.11}$$

[3] See [78, 121] for other approaches.

$$= 1 - \frac{1}{2m}\sum_{\alpha}\sum_{i\in B_\alpha, j\in B_\alpha^c} A_{ij} - \frac{\gamma}{2m}\sum_{\alpha}\sum_{i,j\in B_\alpha}\frac{k_i k_j}{2m} \tag{3.12}$$

$$= 1 - \frac{1}{2m}\sum_{\alpha}\mathrm{Cut}(B_\alpha, B_\alpha^c) - \frac{\gamma}{2m}\sum_{\alpha}\sum_{i,j\in B_\alpha}\frac{k_i k_j}{2m}. \tag{3.13}$$

Summing along the j index first yields

$$= 1 - \frac{1}{2m}\sum_{\alpha}\left(\mathrm{Cut}(B_\alpha, B_\alpha^c) + \frac{\gamma}{2m}\sum_{\alpha}\sum_{i\in B_\alpha} k_i \mathrm{vol}\, B_\alpha\right) \tag{3.14}$$

$$= 1 - \frac{1}{2m}\sum_{\alpha}\left(\mathrm{Cut}(B_\alpha, B_\alpha^c) + \frac{\gamma}{2m}(\mathrm{vol}\, B_\alpha)^2\right). \tag{3.15}$$

Thus, the maxima of modularity coincide with the minima the functional from (3.3), as required.

To see that (3.3) and (3.4) are equivalent, we calculate:

$$\sum_{\alpha}\left(\mathrm{Cut}\left(B_\alpha, B_\alpha^c\right) + \frac{\gamma}{2m}\left(\mathrm{vol}\, B_\alpha - \frac{2m}{\hat{n}}\right)^2\right) \tag{3.16}$$

$$= \sum_{\alpha}\left(\mathrm{Cut}\left(B_\alpha, B_\alpha^c\right) + \frac{\gamma}{2m}\left((\mathrm{vol}\, B_\alpha)^2 - \frac{4m}{\hat{n}}\mathrm{vol}\, B_\alpha + \frac{4m^2}{\hat{n}^2}\right)\right). \tag{3.17}$$

Distributing the summation produces

$$= \sum_{\alpha}\left(\mathrm{Cut}\left(B_\alpha, B_\alpha^c\right) + \frac{\gamma}{2m}\left(\mathrm{vol}\, B_\alpha\right)^2\right) - \frac{\gamma}{2m}\frac{4m}{\hat{n}}\sum_{\alpha}\mathrm{vol}\, B_\alpha + \frac{\gamma}{2m}\sum_{\alpha}\frac{4m^2}{\hat{n}^2}. \tag{3.18}$$

Using the fact that $\sum_{\alpha}\mathrm{vol}\, B_\alpha = \sum_{\alpha}\sum_{i\in B_\alpha} k_i = \sum_{i=1}^{N} k_i = 2m$ and the fact that the third term is constant then gives

$$= \sum_{\alpha}\left(\mathrm{Cut}\left(B_\alpha, B_\alpha^c\right) + \frac{\gamma}{2m}\left(\mathrm{vol}\, B_\alpha\right)^2\right) - \frac{\gamma}{2m}\frac{8m^2}{\hat{n}} + \frac{\gamma}{2m}\frac{4m^2}{\hat{n}} \tag{3.19}$$

$$= \sum_{\alpha}\left(\mathrm{Cut}\left(B_\alpha, B_\alpha^c\right) + \frac{\gamma}{2m}\left(\mathrm{vol}\, B_\alpha\right)^2\right) - \gamma\frac{2m}{\hat{n}}, \tag{3.20}$$

as expected. □

3.2.2 On Convex Relaxations

Proposition 1 makes it very tempting to look for a convex relaxation of (3.5). Recall that given two sets, $\Omega_1 \subset \Omega_2$, where Ω_1 is discrete, and a functional $\mathcal{F} : \Omega_1 \to \mathbb{R}$, a relaxation of $\mathcal{F}$ is any function $\bar{\mathcal{F}} : \Omega_2 \to \mathbb{R}$ such that $\mathcal{F} = \bar{\mathcal{F}}$ on Ω_1. A relaxation is called *exact* in the context of minimization if $\min_{x \in \Omega_1} \mathcal{F} = \min_{x \in \Omega_2} \bar{\mathcal{F}}$.[4] Finally, a relaxation is called *convex* if $\bar{\mathcal{F}}$ is convex.

Modularity (3.2) and balanced TV (3.5) are both defined over a discrete domain, and we would like an extension, or relaxation, of these functions to a larger, continuum domain so that they are easier to work with numerically. Ideally, we could arrive at a convex relaxation and have access to the powerful tools of convex optimization. The formulation in (3.5) suggests one way to proceed. Using (3.5), we already have a convex functional except for the domain, so one would hope that the obvious relaxation obtained by using formula (3.5) on all of $\mathbb{R}^{N \times \hat{n}}$ would be useful. Unfortunately, Theorem 1, below, shows that this obvious relaxation is minimized by the constant matrix and is thus not likely to be useful. In fact, it shows that a large class of other convex relaxations will be uninformative. This will force us to look for nonconvex approaches in Section 3.2.3. Before we state the theorem, we include three more definitions:

Definition 5. *The symmetric group on $\hat{n}$ symbols, $S_{\hat{n}}$, is the set of all permutations on $\{1, \ldots, \hat{n}\}$. Each element $s \in S_{\hat{n}}$ acts on a matrix $U \in \mathbb{R}^{N \times \hat{n}}$ with columns $U(:,1), \ldots, U(:,\hat{n})$ by sending U to another matrix, $s(U)$ with columns $U(:,s(1)), \ldots, U(:,s(\hat{n}))$. If $U \in \Pi(G)$, then $s(U)$ is the same partition with the labels permuted.*

Definition 6. *A map $\mathcal{F}$ from some set of matrices to the real numbers is* symmetric *if it is invariant under column permutations, that is $\mathcal{F}(U) = \mathcal{F}(s(U))$ for all s and U.*

The balanced TV functional (3.5), for example is symmetric.

[4] Analogous notions apply to maximization problems, but we are using (3.5) rather than (3.2) for the moment, for consistency with later sections of the chapter.

Definition 7. *Given a set S lying in a vector space, the convex hull is the smallest convex set containing S.*

In a finite-dimensional vector space, the convex hull exists and is the intersection of all convex sets containing S. For example, if S is given by three noncolinear points in the plane, the convex hull is a triangle.

We now state and prove our theorem on convex relaxations of modularity.

Theorem 1. *Let $\mathcal{F}$ be given by (3.5) with domain $\Pi(G, \hat{n}) = \Pi(G) \cap \mathbb{R}^{N \times \hat{n}}$ (the set of partitions of the nodes of G into $\hat{n}$ classes), and let $\tilde{\mathcal{F}}$ be any symmetric, convex extension of $\mathcal{F}$ to the convex hull of $\Pi(G, \hat{n})$. Then $\tilde{\mathcal{F}}$ has a trivial, global minimizer $\tilde{U}$ that has all columns equal to each other.*

If the symmetry requirement is dropped, then $\tilde{U}$ will have an objective value at least as low as any $U \in \Pi(G, \hat{n})$.

Informally speaking, this theorem states that the solutions of symmetric, convex relaxations of the modularity problem over $\mathbb{R}^{N \times \hat{n}}$ yield no classification information.

Proof. We first consider the symmetric case. Let U lie in the convex hull of $\Pi(G, \hat{n})$. We will use the symmetry and convexity of $\tilde{\mathcal{F}}$ to show that mean all the column permutations of U satisfies the properties required of $\tilde{U}$. Let $\tilde{U} = \dfrac{1}{\hat{n}!} \displaystyle\sum_{s \in S_{\hat{n}}} s(U)$. Then by Jensen's inequality we have

$$\tilde{\mathcal{F}}(\tilde{U}) = \tilde{\mathcal{F}} \left(\frac{1}{\hat{n}!} \sum_{s \in S_{\hat{n}}} s(U) \right) \leq \frac{1}{\hat{n}!} \sum_{s \in S_{\hat{n}}} \tilde{\mathcal{F}}(s(U)) = \tilde{\mathcal{F}}(U). \tag{3.21}$$

Since U was arbitrary, $\tilde{U}$ is a global minimizer.

Finally, all the columns of $\tilde{U}$ are equal[5] and thus uninformative. To see this, take any $\alpha, \beta \in \{1, \ldots, \hat{n}\}$. Let s'' be the permutation that swaps these two values and leaves all the others fixed. Then any $s \in S_{\hat{n}}$ can be written uniquely as $s'' \circ s'$, with $s' = s'' \circ s$. (Proof:

[5]Incidentally, all of the rows are also equal, since row stochasticity is preserved under column permutation.

$s'' \circ s''$ is the identity, so left-multiply by s''.) Thus the β-th column of $\tilde{U}$ is given by

$$\tilde{U}(:,\beta) = \frac{1}{\hat{n}!} \sum_{s \in S_{\hat{n}}} s(U)(:,\beta) \tag{3.22}$$

$$= \frac{1}{\hat{n}!} \sum_{s' \in S_{\hat{n}}} s'' \circ s'(U)(:,\beta) \tag{3.23}$$

$$= \frac{1}{\hat{n}!} \sum_{s' \in S_{\hat{n}}} s'(U)(:,\alpha) \qquad \text{(Note the change in index!)} \tag{3.24}$$

$$= \tilde{U}(:,\alpha) \tag{3.25}$$

so all columns of $\tilde{U}$ are equal.

The non-symmetric case is similar, except that U must lie in $\Pi(G,\hat{n})$ in order for the last equality of (3.21) to hold. Therefore we can only show that the value of $\tilde{\mathcal{F}}$ at $\tilde{U}$ is at least as large as at any point in $\Pi(G,\hat{n})$. □

This means that modularity cannot be convexly relaxed using this embedding of $\Pi(G,\hat{n})$ in $\mathbb{R}^{N\times\hat{n}}$.[6][7] Thus, to make use of smooth optimization techniques we seek a non-convex relaxation. In the following subsection, we present one such family of relaxations.

3.2.3 Ginzburg–Landau Relaxation

In this subsection, we develop a way to relax the modularity problem to a continuum domain, which can make the nonconvexity more manageable. We begin with a convergence result, which, together with Theorem 3 in Chapter 4 is the first multiphase Γ convergence result of which I am aware.

Theorem 2 (Γ-convergence for the Balanced TV problem)**.** *Assume* $\dfrac{\Psi(U(i,:))}{\|U(i,:)\|} \to \infty$ *as* $\|U(i,:)\| \to \infty$*, where* $U(i,:)$ *is the* i*-th row of* U*. Then the functionals in* (2.9) *with* $G =$

[6] We do note, however, that by means of a different embedding [34] was able to obtain a convex relaxation with solutions which, while not discrete, are also not trivial. Thus, the embedding requirement is a non-trivial part of our theorem. Other related works include [32] and [1].

[7] Our proof does not rely on many specific properties of modularity, and indeed, a similar theorem holds for any symmetric quality function over a discrete domain.

$||k^T U||_2^2$ Γ-*converge to the functional*

$$\begin{cases} |U|_{TV} + \dfrac{\gamma}{2m}||k^T U||_2^2 & \text{if } U \text{ corresponds to a partition} \\ +\infty & \text{otherwise.} \end{cases} \tag{3.26}$$

In particular,

- *for any sequence* $\epsilon_n \to 0$*, and any corresponding sequence* U_ϵ *of minimizers of* $\mathcal{F}_{\epsilon_n}$*, there is a subsequence that converges to a maximizer of modularity, and*
- *any convergent subsequence of the* U_ϵ *converges to a maximizer of modularity.*

The proof is given in Appendix A.

Moving forward, we focus on minimizing the relaxed functionals from Theorem 2.[8]

3.3 Numerical Scheme

3.3.1 Merriman–Bence–Osher Iteration

We minimize the functional from (2.9) using an adaptation of the graph MBO scheme. We call our approach Balanced TV. The pseudocode for our particular case is Algorithm 2. The most expensive part of this procedure is evaluating the matrix exponential, for which we use a pseudospectral scheme (see Section 3.3.2).

We treat the forcing term implicitly, which differs from several recent studies, such as [14, 79, 107]. This can be done efficiently because the operator $M = L + \dfrac{\gamma}{m}kk^T$ is positive semi-definite and can be applied to a vector in linear time. Implicit treatment has the advantage of avoiding an inner loop, which is time-consuming, has a timestep restriction, and adds another user-set parameter (namely the inner loop timestep).

[8] While using the Ginzburg–Landau functional does introduce a Laplacian into our formulation, our approach should not be considered a spectral method, such as those in [117, 118]. In Section 3.3, we will use a *pseudospectral* approach in our numerical scheme, but our energy (3.26) is TV-based, which, as discussed in Section 2.2, has very different optima from related quadratic optimization energies, which are more closely associated with some spectral approaches. In Section 3.4, we will see numerically that the answers are different from one particular spectral method.

Algorithm 2 Balanced TV MBO scheme

Input A, $\hat{n}$, γ, dt

Initialize U uniformly randomly.

Set $n = 0$.

while A stationary point has not been reached **do**

$U^{n+\frac{1}{2}} = e^{-dtM}U^n$ where $M = L + \dfrac{\gamma}{m}kk^T$

$U^{n+1} = \text{threshold}(U^{n+\frac{1}{2}})$

$n = n + 1$

end while

Output U^{n+1}, a partition matrix

The case where A is dense[9] can be approached using the Nyström method, as in [14,57,58]. Note, however, that one must find a way to estimate k and $2m$ efficiently in that case. An alternative is to sparsify the network in preprocessing, which is the approach taken in Section 3.4. This was fast compared to the time to partition the resulting sparse network.

3.3.2 Treating the Matrix Exponential

As stated above, the most time-intensive step in the MBO iteration is the matrix exponential, and this step is repeated many times. Therefore, it makes sense to use a pseudospectral scheme, as described, for instance, in [14]. This means that we precompute the eigenvalues and eigenvectors of M, and use them to solve the matrix exponential. By doing the eigenvalue calculation up front, each iteration is accelerated enough that the computational bottleneck is moved away from the MBO iteration. The scheme is depicted in Algorithm 3.

[9] That is, A requires $O(N^2)$ storage, and $x \to Ax$ requires $O(N^2)$ operations for a vector x.

Algorithm 3 Pseudospectral Balanced TV MBO scheme

Input A, $\hat{n}$, γ

Initialize U uniformly randomly.

Calculate the eigenvalues of M, and form the diagonal matrix D with its diagonals being the eigenvalues.

Also calculate the eigenvectors and form the matrix V whose columns are the eigenvectors.

while a stationary point has not been reached **do**

$\hat{U}^n = V^T U^n$

$\hat{U}^{n+1} = e^{-dtD}\hat{U}^n$

$U^{n+\frac{1}{2}} = V\hat{U}^{n+1}$

$U^{n+1} = \text{threshold}(U^{n+\frac{1}{2}})$

end while

Output U^{n+1} a partition matrix.

In practice, it may not be possible to calculate the full spectrum of M, if M is large. In this case, we calculate the N_{eig} smallest eigenvalues and eigenvectors of M. Then instead of changing coordinates using a full matrix, use the $N \times N_{\text{eig}}$ matrix V exactly the same way as before. This is equivalent to projecting onto a subspace generated by these eigenvectors, and it makes the algorithms very efficient. See Section 3.4.

To understand the effect of computing only a few eigenvectors, recall that M is positive semi-definite. Therefore, it has an orthonormal eigenbasis. The evolution we are solving, namely $\frac{d}{dt}U = -MU$, can be diagonalized[10] as $\hat{U}_t = -D\hat{U}$ where $\hat{U} = V^T U$, and V is the full matrix of eigenvalues, and D is a non-negative, diagonal matrix. Therefore, the evolution occurs in distinct "modes" (corresponding to the rows of $\hat{U}$) with rates of decay controlled by the eigenvalues of M. The modes corresponding to small eigenvalues persist longer than those corresponding to large eigenvalues (which experience rapid exponential decay), so that it is not a bad approximation to simply project these components away when it is practically necessary.

[10] M is symmetric and thus diagonalizable.

We use Anderson's iterative Rayleigh–Chebyshev code [6] — which the author kindly provided to us — to get the eigenvalues and eigenvectors. We generally set $N_{\text{eig}} = 5\hat{n}$, although principled ways to determine N_{eig} is an important area of potential further study, as the performance of the algorithm depends on this parameter in non-obvious ways.

3.3.3 Determining the Number of Communities

The preceding algorithm assumes a fixed $\hat{n}$. In practice, we found three methods of determining the value of $\hat{n}$:

1. Use domain knowledge — for instance, the plumes example, we know that the main pixel groups ought to be sky, dirt, gas, etc.

2. Try several values of $\hat{n}$ and take whichever one produces the highest modularity — this is most efficient in cases where there are few communities, as in MNIST. Note that the most time-consuming part of the MBO scheme, namely computation of eigenvectors, need only be done once. Thus several different values of $\hat{n}$ can be tried without incurring much extra cost. For example, in the MNIST example, we could try $\hat{n} = 1, \ldots, 15$ to explore which digits can be split or merged together.

3. Recursively partition the network — this is most useful when many communities are present, as in the LFR networks used in Section 3.4.[11] Further partitioning is only accepted at each step if it increases modularity. This approach worked well in our examples, although in the case of LFR, where $O(N)$ communities are present, a lot of recursion is needed. This is compensated by the fact that the subgraphs grow smaller and smaller near the end.

[11] The value of $\hat{n}$ at which recursion becomes efficient is not trivial to define rigorously, or even heuristically, as it involves understanding how large N_{eigs} should be for a given $\hat{n}$, the depth of recursion expected, and the computation time required to get N_{eigs} eigenvalues, which in practice depends on the efficiency of parallelization, the computer architecture, etc. Informally, we found that recursion begins to make sense on our data sets when $\hat{n}$ exceeds about 10.

3.3.4 Scaling

We expect the scaling of our approach to be roughly linear in N, as suggested by the following informal argument. The main components of the algorithm are

1. finding eigenvalues and eigenvectors (empirically better than quadratic in N),
2. changing coordinates using only the leading eigenvectors ($O(N)$ per iteration, with empirically $O(1)$ iterations needed to converge),
3. evaluating the exponential of a vector componentwise (also $O(N)$ per iteration), and
4. thresholding ($O(N)$ per iteration).

The preceding estimates all apply in the case where no recursion is needed. If the recursion is done by partitioning the graph into $\hat{n}$ pieces at each level, then the cost is heuristically on the order of

$$\tilde{O}(N) + \hat{n}\tilde{O}\left(\frac{N}{\hat{n}}\right) + \hat{n}^2\tilde{O}\left(\frac{N}{\hat{n}^2}\right) + \cdots + O(N)O(1) = \tilde{O}(N)$$

where $\tilde{O}$ means that logarithmic terms are neglected, and each term in the sum is the product of the number of partitioning problems to be solved with the size of the partitioning problems. This scalability is roughly borne out in Tables 3.1 and 3.2, although we note that there are additional complications, based on the varying number of communities to be produced, differences in the efficiency of parallelization at different scales, and differences in the structure of A.

3.3.5 The Choice of Timestep

Our approach requires the selection of parameters, including γ, dt, N_{eig}, and $\hat{n}$. In order to simplify the exploration of this parameter space in practical applications, it is useful to have some theory about the choice of these parameters. Here, we describe how to set dt in the MBO scheme. This is especially useful in the recursive implementation, as the appropriate

timestep empirically decreases as the graph gets smaller, and it would be laborious for a human to tune a parameter each recursion step.

Our derivations are inspired by those in [156], and proofs are deferred to Appendix B. First, we consider a lower bound on the timestep:

Proposition 2 (Lower bounds on the timestep)**.** *Let $U_0 \in \Pi(G, \hat{n})$. If U satisfies $\frac{d}{dt}U = -MU$ with initial data U_0, then we have the following bounds:*

1. $$\|U(\tau) - U_0\|_\infty \leq e^{2(\gamma+1)k_{\max}\tau}$$

   where τ is the MBO timestep and $k_{\max}$ is the largest element of k.

2. *In the case where $\hat{n} = 2$, this bound implies that if the MBO timestep τ satisfies*

   $$\tau < \frac{\log 2}{2(\gamma+1)k_{\max}} \approx \frac{0.15}{(\gamma+1)k_{\max}},$$

   then the MBO iteration is stationary/reaches a fixed point.

3. *If R is the spectral radius of M, we also have*

   $$\|U(\tau) - U_0\|_\infty \leq \sqrt{\hat{n}}\,\|U_0\|_2 \left(e^{\tau R} - 1\right).$$

4. *If $\hat{n} = 2$, the MBO iteration is guaranteed to be stationary whenever*

   $$\tau < R^{-1} \log\left(1 + N^{-\frac{1}{2}}\right).$$

Although we had to restrict to $\hat{n} = 2$ in Proposition 2, we used the timestep restriction regardless of $\hat{n}$ — indeed we expect that $\hat{n} = 2$ is the worst case, although we are unable to prove it at present.

The upper bound on the timestep is more delicate. Normally, the upper bound would be determined using error bounds and stability estimates, the theory of which is incomplete in the graph setting. Instead, we use the following heuristic to motivate our bounds: In most

cases, M is strictly positive definite, so the evolution $\frac{d}{dt}U = -MU$ forces U to decay toward 0. The idea behind MBO is that the diffusion effects give information about curvature on short time scales, and the long time scales give information about more global quantities. Therefore, in the graph context, it makes sense to try to understand the time scale that is "long" and set the timestep to be shorter than that. Using the approach to 0 as a convenient notion of long-time behavior, we obtain the following useful bounds:

Proposition 3 (Decay estimates for M)**.** *Let $\frac{d}{dt}U = -MU$ with initial data $U_0 \in \Pi(G, \hat{n})$. Then the following bounds hold:*

1. *Assume λ_1 is the smallest eigenvalue of M. Then*

$$\|U\|_2 \leq e^{-\tau\lambda_1} \|U_0\|_2 .$$

2. *Let M be nonsingular. Then for any $\epsilon > 0$, we have $\|U(\tau)\|_\infty < \epsilon$ if*

$$\tau > \lambda_1^{-1} \log\left(\frac{\|U_0\|_2}{\epsilon}\right).$$

In practice, setting the timestep as the geometric mean between this upper bound and the lower bound from Proposition 2 has produced good results without resorting to hand-tuning of parameters.

3.4 Results

3.4.1 Summary

Tables 3.1 and 3.2 summarize the results of our Balanced TV algorithm on several examples, mostly drawn from machine learning and image processing.[12] We compared our method to the Modularity MBO algorithm from Hu et al. [79], as well as three other well-known

[12] We also performed some tests of our method on biological and social networks but found that the results were not as encouraging, apparently due to some structural differences from our machine learning networks — it would be interesting to understand this issue more.

algorithms: the Louvain method [18],[13] the hierarchical method of Clauset, Newman, and Moore [38],[14] and a classic spectral recursive bipartitioning method of Newman [118]. Our own method and that of Hu et al. were written in MATLAB except for the eigenvector computations, which use Anderson's Rayleigh–Chebyshev code [6], written in C++ with OpenMP support. The three other methods are slight modifications of igraph's library implementations [42], which are written in the C programming language. In practice, the difference in programming language may make a difference in speed, although the eigenvalue computation is typically the most time-intensive part of the computation. We chose a single conservative timestep for Modularity MBO rather than hand-tuning for each experiment. Our method and that of Hu et al. use a random starting seed, so we ran those codes 20 times and report the best modularity and classification rate and the median time.

Overall, we found that our method is competitive on these data sets. Our method generally found higher-modularity partitions and had faster run times than either the method of Hu et al. or of Newman.[15] The Louvain method and our method often gave similar modularity scores, although the partitions they uncovered were not necessarily similar. For example, on the MNIST example, our method achieved the higher modularity score, but the Louvain partition matched the true labels more closely. On the Plume40 example, the opposite effect occurs, with our method achieving the lower modularity score but finding a partition that is closer to the true labeling of the pixels. Such issues are a manifestation of the well-known near degeneracy of the modularity energy [69], where many dissimilar partitions can receive similarly high modularity scores. It is also an indication that modularity can benefit from supervision, regularization, biased initialization, or some other device in order to

[13]Initial tests relied on the slower GenLouvain code from [84]. A reviewer later suggested that we use a faster implementation, which led us to redo some the tests using the igraph Louvain code. Thus, in this section, we will carefully distinguish between results obtained with GenLouvain code and results from igraph's Louvain code.

[14]This is an old method but is the standard "fast" solver in the IGRAPH library, which is very widely used. Other, more recent approaches, may give better results.

[15]We chose this particular spectral method because it was available in igraph. A complete comparison with other spectral methods would be interesting but is beyond the scope of this chapter.

reliably find the partition that is most appropriate for the problem. In Table 3.3, we illustrate the effectiveness of including a small amount of supervision with our method, using (3.8).

		Moons	MNIST	LFR50k	Urban	Plume7	Plume40
	Nodes	2,000	70,000	50,000	94,249	286,720	1,638,400
	Edges	1.8×10^4	465,832	7.9×10^5	707,042	5,281,070	29,247,003
	Communities	2	10	2,000	5	5	5
	Res. Param.	0.2	0.5	15	0.1	1	1
Mod.	Our method	0.84	0.92	0.77	0.95	0.76	0.64
	Hu et al. [79]	0.85	0.91	0.58	0.95	0.74	0.64
	Hier. [38]	0.77	0.88	0.88	0.94	0.65	0.92
	Louvain [18]	0.72	0.83	0.89	0.90	0.78	0.97
	Spectral [118]	0.60	0.56	−5.88	0.90	0.30	0.04
	Ref	0.83	0.92	0.89	0.90	0.00	0.00
Classification	Our method	0.97	0.90	0.92	—-	—-	—-
	Hu et al. [79]	0.95	0.80	0.72	—-	—-	—-
	Hier. [38]	0.98	0.93	0.80	—-	—-	—-
	Louvain [18]	0.98	0.96	0.87	—-	—-	—-
	Spectral [118]	0.95	0.30	0.09	—-	—-	—-
Time (sec.)	Our method	0.55	59	63	19	135	1284
	Hu et al. [79]	0.80	167	206	42	152	39196
	Hier. [38]	0.55	16	6	44	3066	9437
	Louvain [18]	0.38	9	6	14	89	520
	Spectral [118]	0.87	301	1855	24	265	1804

Table 3.1: Results on six networks. Our method generally does better than the spectral method and Hu et al., but note that modularity and classification score sometimes disagree. Dashes mark cases where metadata was unavailable.

		Jas. Rid.	Samson	Cuprite	FLC	Pavia U	Salinas	Salinas 1
	Nodes	19,800	14,820	30,162	208,780	207,400	7,092	111,063
	Edges	116,375	83,995	170,775	1,582,478	1,683,720	51,397	551,980
	Communities	4	3	12	3	9	6	16
Mod.	Our method	0.99	0.98	0.99	0.94	0.93	0.97	0.96
	Hu et al. [79]	0.99	0.98	0.90	0.94	0.94	0.97	0.96
	Hier. [38]	0.98	0.98	0.99	0.93	0.93	0.97	0.96
	Louvain [18]	0.99	0.98	0.99	0.90	0.88	0.95	0.95
	Spectral [118]	0.91	0.90	0.91	0.90	0.90	0.96	0.90
	Ref	0.90	0.90	0.90	0.90	0.90	0.90	0.90
Time (sec.)	Our method	17	13	42	121	160	4.6	96
	Hu et al. [79]	40	27	63	203	270	3.3	117
	Hier. [38]	1.5	1.1	2.4	378	411	0.74	66
	Louvain [18]	1.5	1.2	2.7	39	40	0.75	15
	Spectral [118]	28	10	148	38	65	6.5	24

Table 3.2: Results on additional hyperspectral data sets. The resolution parameter was 0.1, and the reference partition has all nodes in the same community. Our method achieves the highest modularity score in each network except for Salinas, where Hu et al.'s method gets slightly higher results. Our method partitioned recursively and initialized with $k$means clustering on leading eigenvectors that had been computed for use in the pseudospectral scheme.

		Moons	MNIST
Modularity	Unsupervised	0.84	0.91
	10% supervised	0.84	0.92
	Reference	0.83	0.92
Classification	Unsupervised	0.97	0.90
	10% supervised	0.97	0.97
Modularity Consistency	Unsupervised	0.75	0.65
	10% supervised	1.00	1.00
Classification Consistency	Unsupervised	0.75	0.05
	10% supervised	1.00	0.65

Table 3.3: Results of our method using networks constructed from the two-moons and MNIST examples with and without 10% supervision. Consistency indicates the fraction of cases for which the results were within 2% of the best value achieved. In the two-moons example, supervision improves consistent matching to metadata. In the MNIST example, both consistency and classification are substantially improved. Note that in both cases, the peak modularity is not changed, indicating that the supervision helps the solver find local maxima that are more relevant to the classification task.

3.4.2 Analysis of each experiment

We now describe the individual experiments.

Two Moons Two Moons consists of 2,000 points in 100-dimensional space, sampled from two half-circles of radius 1 centered at $(0, 0, 0, 0, \ldots)$ and $(1, .5, 0, 0, \ldots)$, respectively, with Gaussian noise of variance 0.02 added, see Fig. 3.1. The classification problem is to assign each point to the circle in which it originated. We constructed a 13-nearest neighbors graph[16] with the edge weights given by a Gaussian law, with locally-determined decay parameters

[16]The choice of 13 here is somewhat arbitrary. We did not tune this parameter.

[169]. The number of classes assumed known in our tests.

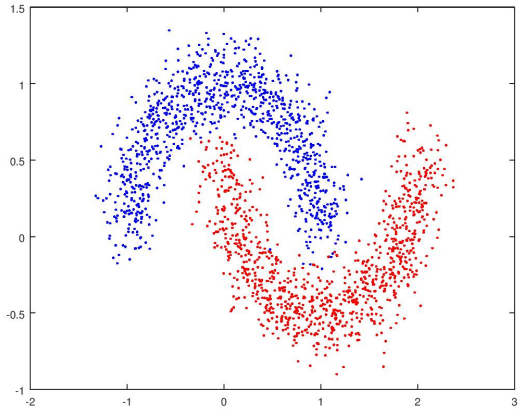


Figure 3.1: Projection of the two moons example onto two dimensions

MNIST MNIST consists of 70,000 28×28-pixel images, each of which contains a single handwritten digit [97]. The task is to identify the digit in each image. I constructed the graph by projecting onto 50 principle components for each image and then using a 10-nearest neighbors graph with self-tuning Gaussian decay for the weights [169]. The number of classes was assumed known in our tests. As in [79], 11 classes were assumed, as there are two different ways to write the digit 1, with or without the top flag and flat base. This modularity landscape was particularly troublesome, with about 25% of the partitions we found having higher modularity than the ground-truth partition, despite the fact that partitions with a classification accuracy greater than 95% were found only about 4% of the time by our solver. This illustrates the importance of getting good features when building similarity graphs.

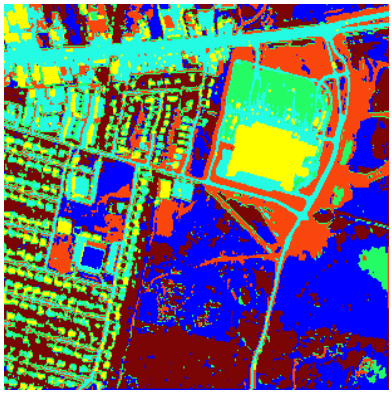
LFR 50k This is a well-known ensemble of artificial networks [96]. We used the following parameters to generate it: 50,000 nodes, mean degree of 20, maximum degree of 50, power law degree-distribution with exponent 2, power law community size distribution with exponent 1, effective mixing parameter of 0.2, maximum community size of 50, minimum community size of 10.[17] The large number of small communities makes this a challenging problem — similar

[17]This is different from the LFR-type network in [79], which combined multiple LFR networks.

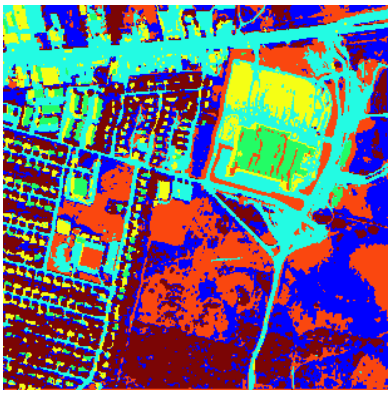
experiments I performed on a 1,000-node networks with 40 communities gave near-perfect classification. We use purity to gauge classification accuracy. Given two partitions $B_1, \ldots, B_{\hat{n}}$ and $B'_1, \ldots, B'_{\hat{n}'}$, the purity is defined as $\frac{1}{N} \sum_{\alpha=1}^{\hat{n}} \max_{\beta=1,\ldots,\hat{n}'} \#\{i : i \in B_\alpha \text{ and } i \in B'_\beta\}$, where $\#$ denotes the cardinality. Recursive partitioning was used on this network as described in Section 3.3.3.

Urban Image The urban hyperspectral image is a 307×307 image of an urban setting, where each pixel encodes the intensity of light at 129 different wavelengths. The classification problem is to identify pixels that contain similar materials, such as dirt, road, grass, etc. I computed the graph representation using nonlocal means as described in Section 2.5. I selected the images in Fig. 3.2 for visual appeal from a collection of 200 segmentations. We compared with a recent NLTV algorithm [171], which is specifically designed for hyperspectral imaging applications and found our segmentation competitive in terms of visual appeal. We also compared with Modularity MBO and GenLouvain [84] segmentations. For instance, Balanced TV does well at placing the grass into a single class and correctly resolved the difference between pavement and dirt. Balanced TV gives the sharpest resolution of the roads and the surrounding dirt in the upper right. Our method does have a little trouble compared to GenLouvain when resolving the buildings just below the large road in the upper-left corner of the picture, although this is partly due to the fact that the roofs there are made of different materials from most of the houses further down in the image, and NLTV has a similar problem.

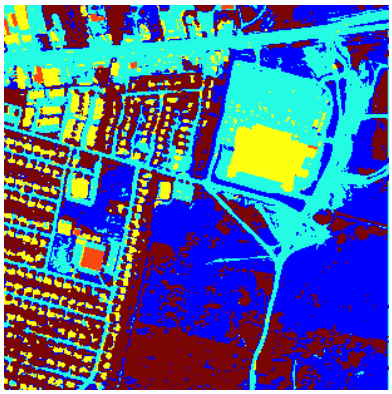
RGB image



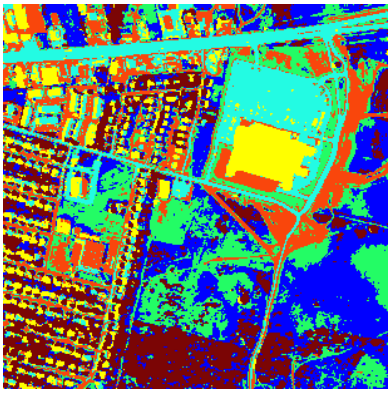
Our method



Modularity MBO



GenLouvain segmentation



NLTV segmentation [171]

Figure 3.2: The urban dataset segmented using different methods. Our method effectively separates the dirt from roads, resolving the roads in the upper right corner, and placing all of the grass into a single class. It has some difficulty with the buildings in the upper left corner, just below the main road, which are a different material from the other buildings.

Plume Hyperspectral Video The gas plume hyperspectral video records a gas plume being released at the Dugway Proving Ground [63, 102, 110]. The graph was constructed by the same procedure as the urban dataset, simply concatenating each frame side-by-side into one large image and using nonlocal means to form the graph. Each frame has 320×128 pixels with data from 129 wavelengths. I used two versions of this data set, one with 7 frames, and another with 40 frames. We have included the segmentation of one frame in Fig. 3.3, together with segmentations produced by competing algorithms. This problem is difficult because of the diffuse nature of the gas, which leads to a faint signal among many wavelengths and with boundaries that are difficult to determine. Our method is the only one among those I tested that places the entire plume in a single class. The images shown were chosen by me as the best out of thirty for visual appeal.

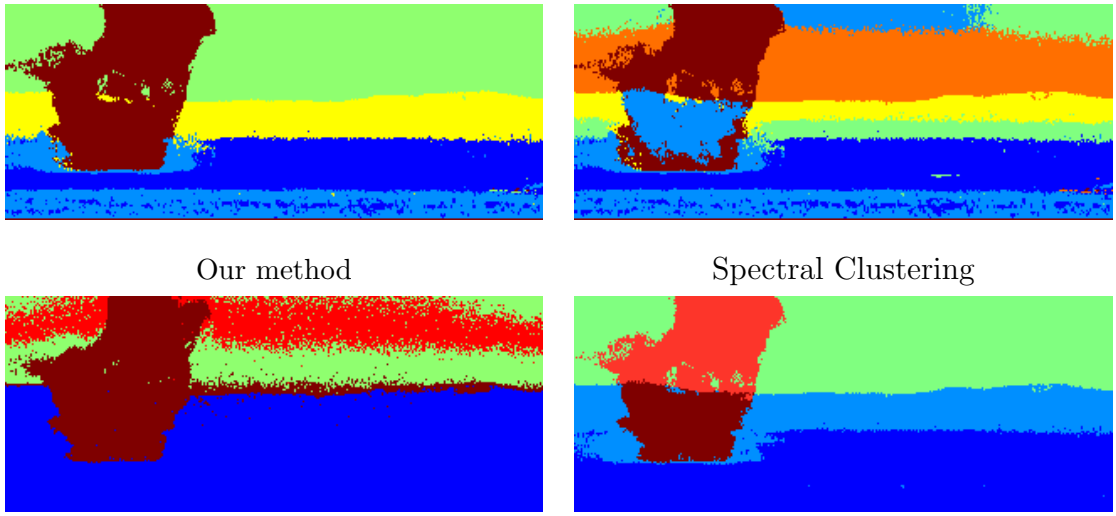


Figure 3.3: Segmentations of the plume hyperspectral video using different methods. Observe that our method is the only method that gets the whole plume into a single class without any erroneous additions.

Other Hyperspectral Examples We included seven additional hyperspectral image examples, which are well-known in the image processing community. In each case, we formed

the 10-nearest-neighbor[18] graph using nonlocal means and VLFeat. See Appendix C for more details on the images used and Table 3.2 for the numerical results. Overall, our algorithm performs competitively on these examples in terms of modularity. The speed is slower than Louvain, but the run time is still reasonable, and the modularity scores are consistently high.

[18]The number of nearest neighbors not tuned for optimality.

CHAPTER 4

Stochastic block models are a discrete surface tension

This chapter is organized as follows. In Section 4.1, we present background information about stochastic block models, total variation, and surface tension. In Section 4.2, we state and prove our main result, which establishes an equivalence between discrete surface tension and maximum likelihood estimation via an SBM. In Section 4.3, we discuss three numerical approaches for performing SBM MLE: mean-curvature flow, Γ-convergence, and threshold dynamics. We discuss our results on both synthetic and real-world networks in Section 4.4.

4.1 Background

4.1.1 Stochastic Block Models (SBMs)

The most basic SBM has N nodes and an assignment $g : \{1, \dots, N\} \to \{1, \dots, \hat{n}\}$ that associates each node with one of $\hat{n}$ sets. It also has an $\hat{n} \times \hat{n}$ symmetric, nonnegative matrix ω. One generates an undirected, unweighted graph as follows: for each pair of nodes, i and j, we place an edge between them with probability $\omega_{\alpha\beta}$, where α and β, respectively, denote the community assignments of nodes i and j. Similar models have been studied and rediscovered many times [41, 52, 55, 59, 75, 136, 148]. In the present chapter, we use the SBM from [119].

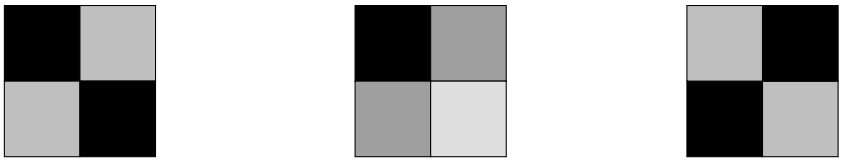
There is considerable flexibility in the choice of ω, which leads in turn to flexibility in the SBMs themselves [55, 136]. Three examples of ω, using $\hat{n} = 2$, will help illustrate the diversity of block structures.

1. If $\omega_{11} = \omega_{22} > \omega_{12}$, one obtains traditional assortative community structure, in which nodes have a larger probability to be adjacent to nodes in the same community instead of ones in different communities.

2. If $\omega_{11} = \omega_{22} < \omega_{12}$, nodes tend to associate more with ones that are in other communities. As $\omega_{12} \to 0$, the graph becomes increasingly bipartite.

3. If $\omega_{11} > \omega_{12} > \omega_{22}$, there is a core–periphery (CP) structure: nodes from set 1 are connected densely to many nodes, but nodes from set 2 are connected sparsely to other nodes [43, 141].

These three examples are also illustrated in Fig. 4.1. To simplify our presentation, we refer to latent block structures as "community structure," regardless of the form of ω.



Community structure Core–periphery structure Bipartite structure

Figure 4.1: Examples of different connectivity patterns that one can generate using stochastic block models. Each panel corresponds to a different kind of structure. In each panel, the upper-left and lower-right squares represent the density of connections between nodes in the same set, and the upper-right and lower-left squares represent the density of connections between nodes in different sets. Darker squares represent more densely connected sets of nodes. In (assortative) community structure, nodes are densely connected to other nodes in the same community but sparsely connected to nodes in other communities. In the core–periphery structure, core nodes (as illustrated by the dark square in the upper left) are densely both to other core nodes and somewhat densely connected to peripheral nodes, but the latter predominantly have connections only to core nodes. In bipartite block structures, a set of nodes is more densely connected to nodes in other sets than to nodes in its own set. One can also model other structures, such as hierarchical and role-based structures, using SBMs. See Section 4.1.1 for additional discussion. [This figure is inspired by a figure from [83].]

The above SBM is not realistic enough for many applications, largely because each node has the same expected degree [85]. To address this issue, one can suppose that one knows the degree sequence $\{k_i\}$ and then define connection probabilities to take this information into account. The easiest approach is to model the adjacency-matrix elements A_{ij} as Poisson-distributed with parameter $\omega_{g_i g_j} \frac{k_i k_j}{2m}$, where k and m are defined as in the previous chapter. An important point to note is that this allows both multi-edges and self-edges. Although such edges can have important effects [56], we neglect them for simplicity.

Given an observed network, one can attempt to infer some sort of underlying community structure by statistical fitting methods. There are several ways to do this, including

maximum-likelihood estimation (MLE), maximum a posteriori (MAP) estimation, and maximum marginal likelihood (MML). In MLE, one chooses the parameters g and ω under which an observed network is most probable (without using a prior), MAP yields the most probable parameter configuration under a Bayesian prior, and MML yields the best community assignment for each node individually by integrating out all of the other variables [113, 136]. We use MLE, which is the simplest approach. In mathematical terms, the problem is stated as

$$\operatorname*{argmax}_{g,\omega} P(A|g,\omega)\,, \tag{4.1}$$

where P is the probability density function. Because we determine the edges independently, P is given by

$$P(A|g,\omega) = \prod_{i\le j} P(A_{ij}|g,\omega) = \prod_{i\le j} P\left(A_{ij}\,\middle|\, w_{g_i g_j}\frac{k_i k_j}{2m}\right)\,.$$

We use a Poisson distribution, so

$$P(A_{ij}|\lambda) = \begin{cases} \dfrac{\lambda^{A_{ij}}}{A_{ij}!}e^{-\lambda}\,, & i \neq j\,, \\ \dfrac{\lambda^{A_{ij}/2}}{(A_{ij}/2)!}e^{-\lambda}\,, & i = j\,, \end{cases}$$

where the need for cases arises from the convention that $A_{ii} = 2$ if a self-edge is present. To solve (4.1), one can equivalently maximize the logarithm of $P(A|g,\omega)$. Conveniently, this changes the multiplicative structure into additive structure and allows one to drop irrelevant constants. The resulting objective function is

$$\operatorname*{argmax}_{g,\omega} \sum_{i,j}\left[A_{ij}\log(\omega_{g_i g_j}) - \omega_{g_i g_j}\frac{k_i k_j}{2m}\right]\,. \tag{4.2}$$

The exact optimization of (4.2) is NP-hard, so one needs to use a heuristic. Possibilities include greedy ascent [85], Kernighan–Lin (KL) node swapping [85, 87], and coordinate descent [119]. As far as we are aware, the theory of these approaches has not received much attention, although the associated chapters generally include positive results. In light of the extreme nonconvexity of the modularity objective function [69] (which is known to be related to the planted-partition form of the SBM [119]), we expect that multiple random

initializations are needed for any local algorithm. (Ideas from consensus clustering may also be helpful [55].)

Ways to elaborate the above SBM include incorporating overlapping and hierarchical communities [133, 135], generalizing to structures such as time-dependent and multilayer networks [134], or incorporating metadata [121]. There are also Bayesian models and pseudo-likelihood-based methods [5, 136]. We do not consider such embellishments in this chapter, although we conjecture that our approach will generalize to some of these settings.

In the case of SBMs, one can use TV to express (4.2), but we find a more natural formulation in terms of surface-tension energy (a related notion).

4.1.2 Surface Tension

Very roughly, one can consider a metal object as being composed of a large number of crystals that range in size from microscopic to macroscopic [8]. Each crystal is a highly-ordered lattice; and there is a thin, disordered interface between crystals. The sizes and orientations of these crystals affect material properties, and one goal of annealing processes is to allow crystals to reorganize to produce a useful metal. The potential energy of a given crystal configuration is roughly

$$\sum_{\alpha,\beta} \sigma_{\alpha\beta} \text{Area}(\Gamma_{\alpha\beta}) \,, \tag{4.3}$$

where $\Gamma_{\alpha\beta}$ is the interface between crystals α and β, and $\sigma_{\alpha\beta}$ is the surface tension energy density between these crystals. Each $\sigma_{\alpha\beta}$ is different, based on physical considerations that involve the exact offset between the orientations of the lattices in each pair of crystals. When prepared and heated appropriately, the individual crystals decrease (4.3) by growing to consume their neighboring crystals. See [47, 81, 90] for further background information.

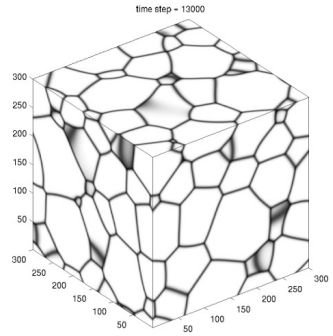


Figure 4.2: An example arrangement of crystals. The interfaces between pairs of crystals grow into each other according to motion by mean curvature. [This image from Cenna/Wikimedia Commons/Public Domain [29].]

We exploit the appearance of surface area in (4.3) to cast it as a TV problem. Mathematically, we model the metal as a region of space that is partitioned into $\hat{n}$ regions, corresponding to the crystals in the metal. Let u^α and u^β, respectively, denote the characteristic functions of the regions α and β. Therefore,

$$\text{Area}_{\alpha\beta} = |u^\alpha|_{\text{TV}} + |u^\beta|_{\text{TV}} - |u^\alpha + u^\beta|_{\text{TV}}\,.$$

Each interface between two regions evolves according to mean-curvature flow, which is the gradient descent of TV. Thus, the surface-tension flow is locally mean-curvature flow, except at the junction of three or more crystals [47,81]. Because of this connection, one can use some of the ideas (such as phase-field and threshold-dynamics methods [47]) from TV minimization to perform surface-tension minimization. When using threshold dynamics, it is possible to do theoretical analysis in the form of Lyapunov functionals, Γ-convergence, and descent conditions [81].

4.2 An Equivalence Between SBM MLE and Discrete Surface Tension

We now present a mathematical result that connects SBM MLE and discrete surface tension.

Proposition 4. *Maximizing the likelihood of the parameters g and ω in the degree-corrected SBM is equivalent to minimizing*

$$\sum_{\alpha,\beta} \left[W_{\alpha\beta} \mathrm{Cut}_{g,A}(\alpha,\beta) + e^{-W_{\alpha\beta}} \frac{\mathrm{vol}_{g,A}(\alpha)\,\mathrm{vol}_{g,A}(\beta)}{2m} \right], \tag{4.4}$$

where $\mathrm{Cut}_{g,A}(\alpha,\beta) = \sum_{\substack{g_i=\alpha \\ g_j=\beta}} A_{ij}$, $\mathrm{vol}_{g,A}(\alpha) = \sum_{g_i=\alpha} k_i$, *and* $W_{\alpha\beta} = -\log \omega_{\alpha\beta}$.

The analogy with continuum surface tension is as follows. Graph cuts are analogous to surface area: given a domain in $\mathbb{R}^3$, one can superimpose a fine grid on space and count the number of edges that cross the boundary to estimate its surface area. In the limit of an infinitely fine grid, this estimate converges to the surface area under appropriate conditions [21]. Similarly, graph volumes are analogous to continuum volumes. The quantities $W_{\alpha\beta}$ play the role of surface tensions $\sigma_{\alpha\beta}$, so the first set of terms is analogous to (4.3). One can view the second set of terms as a soft volume constraint. A constraint is "soft" if violating it adds a finite penalty on an objective function, so minimizers will usually approximately satisfy the constraint. Volume-constrained versions of (4.3) have received a great deal of attention [82, 90].[1]

Proof. In [119], it was shown that maximizing the log-likelihood of the parameters g and ω for a particular version of the degree-corrected SBM amounts to maximizing (4.2). Let $\Pi(G,\hat{n})$ be the set of partitions of the nodes of a graph G (associated with an adjacency matrix A) into at most $\hat{n}$ sets. Substituting $W_{\alpha\beta} = -\log \omega_{\alpha\beta}$ into (4.2) gives

$$\operatorname*{argmin}_{\substack{W_{\alpha\beta}\in\mathbb{R} \\ g\in\Pi(G)}} \sum_{i,j} \left[A_{ij} W_{g_i g_j} + \frac{k_i k_j}{2m} e^{-W_{g_i g_j}} \right].$$

Rearranging the summations gives

$$\operatorname*{argmin}_{\substack{W_{\alpha\beta}\in\mathbb{R} \\ g\in\Pi(G,\hat{n}) \\ \hat{n}\in\mathbb{N}}} \left[\sum_{\alpha,\beta} \sum_{\substack{g_i=\alpha \\ g_j=\beta}} A_{ij} W_{\alpha\beta} + \sum_{\alpha,\beta} \sum_{\substack{g_i=\alpha \\ g_j=\beta}} \frac{k_i k_j}{2m} e^{-W_{\alpha\beta}} \right],$$

[1] As far as we are aware, our formulation of SBM MLE in terms of graph cuts and volumes is novel, although similar formulas have appeared previously in the literature; see, e.g., [136].

where the inner sums are over all nodes i and j such that $g_i = \alpha$ and $g_j = \beta$. Rearranging again gives

$$\operatorname*{argmin}_{\substack{W_{\alpha\beta}\in\mathbb{R} \\ g\in\Pi(G,\hat{n}) \\ \hat{n}\in\mathbb{N}}} \left[\sum_{\alpha,\beta} W_{\alpha\beta} \sum_{\substack{g_i=\alpha \\ g_j=\beta}} A_{ij} + \sum_{\alpha,\beta} e^{-W_{\alpha\beta}} \sum_{\substack{g_i=\alpha \\ g_j=\beta}} \frac{k_i k_j}{2m} \right] .$$

Using the definition of $\mathrm{Cut}_{g,A}$ in the first set of terms and summing over the j index independently in the second set of terms gives

$$\operatorname*{argmin}_{\substack{W_{\alpha\beta}\in\mathbb{R} \\ g\in\Pi(G,\hat{n}) \\ \hat{n}\in\mathbb{N}}} \left[\sum_{\alpha,\beta} W_{\alpha\beta} \mathrm{Cut}_{g,A}(\alpha,\beta) + \sum_{\alpha,\beta} e^{-W_{\alpha\beta}} \sum_{g_i=\alpha} \frac{k_i}{2m} \mathrm{vol}_{g,A}(\beta) \right] .$$

Finally, we sum over the i index in the second set of terms to obtain

$$\operatorname*{argmin}_{\substack{W_{\alpha\beta}\in\mathbb{R} \\ g\in\Pi(G,\hat{n}) \\ \hat{n}\in\mathbb{N}}} \sum_{\alpha,\beta} \left[W_{\alpha\beta} \mathrm{Cut}_{g,A}(\alpha,\beta) + e^{-W_{\alpha\beta}} \frac{\mathrm{vol}_{g,A}(\alpha)\,\mathrm{vol}_{g,A}(\beta)}{2m} \right] . \tag{4.5}$$

□

One difference between (4.4) and (4.3) is that in the latter (i.e., for a graph), one performs optimization over the $W_{\alpha\beta}$, whereas in the former (i.e., in a continuum), one ordinarily treats the surface-tension densities as fixed by the choice of material that one is modeling. Another difference is that the surface-tension coefficients in the graph setting can be any element of $(-\infty, \infty]$, subject only to the symmetry condition $W_{\alpha\beta} = W_{\beta\alpha}$. In contrast, for a continuum, further restrictions are necessary to ensure well-posedness. Esedoglu and Otto [47] proved the following sufficient conditions for well-posedness:

(1) $\sigma_{\alpha\beta} \geq 0$ for all α, β,

(2) $\sigma_{\alpha,\alpha} = 0$ for all α, and

(3) $\sigma_{\alpha\nu} + \sigma_{\nu\beta} \geq \sigma_{\alpha\beta}$ for all α, β, ν.

In a graph setting, one can use a straightforward change of variables to make W satisfy

requirement (2).[2] In general, however, at least one of requirements (1)–(3) are not necessarily satisfied for a graph.[3]

4.3 Mean-Curvature Flow (MCF), Γ-Convergence, and Threshold Dynamics

We now outline three algorithmic approaches that illustrate how one can use tools from surface-tension theory to solve SBM MLE problems. Our three algorithms are graph versions of mean-curvature flow (MCF), AC evolution, and MBO dynamics. In Section 4.4, we conduct several numerical experiments to demonstrate that these algorithms can effectively evaluate (4.2). We expect the performance of these algorithms to be good relative to other algorithms for SBM MLE, though a full evaluation of this claim is beyond the scope of this chapter. We have posted our code at `http://www.math.ucla.edu/~zach.boyd/code/SBM.zip`. In the next three subsections, we describe how we infer g when ω is fixed. We then describe how to jointly infer ω and g.

4.3.1 Mean-Curvature Flow

Surface-tension dynamics are governed by mean-curvature flow except at junctions. Intuitively, this means that each point on a surface moves in the direction normal to the surface at a speed given by the mean curvature at that point. In the two-phase case, such dynamics have been well-studied, and notions of viscosity solutions and regularity theory have been developed [103]. In the multiphase case, the situation is much more complicated, notably because of the topological changes that can occur and the issue of defining the behavior at

[2]See Appendix D for the change of variables, which causes the sum in (4.4) to instead be over all $\alpha \neq \beta$, so that there are no "internal" surface tensions.

[3]Requirement (1) is false whenever some component of W is negative; this occurs exactly when ω has a component that is larger than 1. Requirement (3) may not hold, because the the components of W can assume any real value. Thus, it is possible to pick some W that violates requirement (3) and generate a network using it.

the junction of three or more phases. In two-phase surface-tension dynamics, it was shown in [22] that one can approximate the flow by solving a discrete-time minimizing-movements problem. Let $\hat{C}_n$ be one of the two regions at time $n\,dt$, where dt is the time step. We then have

$$\hat{C}_{n+1} = \text{argmin}\left[\text{SurfaceArea}(\hat{C}) + \frac{1}{dt}\int_{\hat{C}_n \Delta \hat{C}} \hat{\rho}(p, \hat{C}_n)\mathrm{d}p\right], \tag{4.6}$$

where

$$\hat{\rho}(p, \hat{C}_n) = \inf_{x \in \partial \hat{C}_n} \|x - p\|_2\,,$$

the operation Δ denotes the symmetric difference, and ∂ is the topological boundary operator. The idea behind this approach is, at each time step, to shorten the curve as much as possible without straying too far from the curve location at the previous time step.

In the setting of graphs, a similar approach was developed in [156], where the mean-curvature flow was given by

$$C_{n+1} = \text{argmin}\left[\text{Cut}(C, C^c) + \frac{1}{dt}\sum_{i \in C_n \Delta C} \rho(i, \partial(C_n))\right], \tag{4.7}$$

where C_n is a subset of graph nodes, the operation Δ is again the symmetric difference, and $\rho(i, \partial(C_n))$ is the shortest-path distance from node i to the boundary of C_n. In this context, the boundary of a set of nodes is the set of nodes in C_n with at least one neighbor in C_n^c along with the nodes in C_n^c that have at least one neighbor in C_n. We use the term *boundary node* for any node that lies on the boundary. In the limit of small dt, (4.7) may still evolve, as opposed to the MBO scheme (which we use later), which becomes "stuck" when the time step is too small. Such evolution can still occur, because the penalty (associated with moving any node in $\partial(C_n)$) induced by the second set of terms in (4.7) is 0, regardless of the value of dt. Conveniently, this implies for sufficiently small dt that the only acceptable moves at each time step are allowed to change only the boundary nodes themselves. This makes it possible to drastically reduce the search space when solving (4.7).

Because careful studies in the spirit of [156] are not yet available for multi-way graph partitioning, we resort to a heuristic approach based on what is known in the binary case. Specifically, we are motivated by situation in which time steps are sufficiently small that

only boundary nodes can change their community assignment. Ideally, we wish to compute an optimal reassignment of all boundary nodes jointly in order to minimize (4.4). To save computation time and facilitate implementation, we instead decouple the computations in the following manner: During a single time step, for each boundary node, we compute an optimal assignment of that node, assuming that all other nodes keep their assignment from the beginning of the time step. After this (but before the end of the time step), we assign each boundary node to its community, as computed previously in the time step. Because most nodes are boundary nodes[4] in our SBM-generated graphs, we find it more efficient and easier to consider reassigning all nodes in each time step rather than maintaining and referencing a separate data structure to track the boundary. In Algorithm 4, we give pseudocode for this graph MCF procedure.

[4]Recall that a node is a boundary node if it shares an edge with a node that lies outside of its own community, so most reasonable partitions of many real graphs have many boundary nodes. Additionally, because we use a random initialization of g, most nodes will initially be boundary nodes for most graphs.

Algorithm 4 Modified graph mean-curvature flow (MCF) for SBM MLE (4.2).

Input A, W, $\hat{n}$.

Initialize g uniformly at random, and let $eW = e^{-W}$ be the entry-wise exponential of W.

while not converged **do**

Let $U_{i\alpha} = \delta_{g_i \alpha}$ for each i, α.

Let $X = AU$. // Counts the number of neighbors that each node has in each class

Let $\mathrm{vol}_{g,A} = (k^T U)$.

for $\alpha = 1$ to $\hat{n}$ **do**

Let J_α be the set of nodes currently assigned to group α.

for $\beta = 1$ to $\hat{n}$ **do**

Let J be the indices $1, \ldots, \hat{n}$ aside from β and α.

Let $\mathrm{Delta}(J_\alpha, \beta)$ be given by the following formula:

$$\begin{aligned}
\mathrm{Delta}(J_\alpha, \beta) = {} & 2X(J_\alpha, J)W(J, \beta) \\
& - 2X(J_\alpha, J)W(J, \alpha) \\
& + 2X(J_\alpha, \beta)W(\beta, \beta) \\
& - 2X(J_\alpha, \beta)W(\beta, \alpha) + 2X(J_\alpha, \alpha)W(\beta, \alpha) \\
& - 2X(J_\alpha, \alpha)W(\alpha, \alpha) \\
& + \frac{1}{2m}\left(2k(J_\alpha)\mathrm{vol}_{g,A}(eW(:, \beta) - eW(:, \alpha))\right. \\
& \left. + k(J_\alpha)^2(eW(\beta, \beta) + eW(\alpha, \alpha) - 2eW(\beta, \alpha))\right).
\end{aligned}$$

end for

end for

for $i = 1$ to N **do**

$g_i = \mathrm{argmin}(\mathrm{Delta(i, :)})$. // [Choose uniformly at random in case of a tie.]

end for

end while

Output g.

4.3.2 Allen–Cahn (AC) Evolution

For the particular case of surface-tension dynamics, we proceed as follows. Given a partition of a network, if U is the corresponding $N \times \hat{n}$ matrix, one can show that $W \odot (U^T L U) = -W \odot (U^T A U)$, where $\odot$ is the entry-wise product. Therefore, an appropriate GL functional for our problem is

$$\sum_{\alpha,\beta} \left[-W_{\alpha\beta} U_\alpha^T L U_\beta + \frac{\mathrm{vol}_{g,A}(\alpha)\,\mathrm{vol}_{g,A}(\beta)}{2m} \right] + \frac{1}{2\epsilon} \sum_i \Psi(U(i,:)) \,. \tag{4.8}$$

Because $k^T u$ gives the vector of volumes, one can rewrite (4.8) as

$$\sum_{\alpha,\beta} \left[-W_{\alpha\beta} U_\alpha^T L U_\beta + \frac{k^T U_\alpha e_{\alpha\beta}^{-W} U_\beta^T k}{2m} \right] + \frac{1}{2\epsilon} \sum_i \Psi(U(i,:)) \,, \tag{4.9}$$

where e^{-W} is the entry-wise exponential.

As in a continuum setting, one can prove Γ-convergence.

Theorem 3. *The functionals in* (4.9) Γ*-converge to* (4.4) *as* $\epsilon \to 0$.

See Appendix A for a proof. As far as we are aware, this is the first Γ-convergence result for a multiphase graph energy.

The resulting AC equation is

$$U_t = LUW - \frac{1}{2m} k k^T U e^{-W} - \frac{1}{\epsilon} \sum_i \nabla \Psi(U(i,:)) \,. \tag{4.10}$$

See Appendix E for further details on the numerical solution of (4.10).

4.3.3 MBO Iteration

Esedoglu and Otto [47] developed a generalized version of the MBO scheme (see Algorithm 5) for computing the evolution of multiphase systems modeled by (4.3).

One can apply the MBO idea to community detection in networks by replacing the continuum Laplacian with the (negative) combinatorial graph Laplacian, replacing σ with W, changing u to U, and adding appropriate forcing terms for the gradient descent of the volume-balance terms. See Appendix E for additional implementation details.

Algorithm 5 A multiphase, continuum MBO scheme.

Input the initial state of the domain.

Initialize $u_1, \ldots, u_{\hat{n}}$ as the characteristic functions of the initial domains.

for $n = 1, 2, \ldots$ **do**

  for $\alpha = 1, \ldots, \hat{n}$ **do**

    $u_\alpha^{n+1/2}$ is the solution at time dt to $u_{\alpha,t} = \Delta u_\alpha$ with initial condition u_α^n.

  end for

  for each point x **do**

    $\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \sum_\beta \sigma_{\alpha\beta} u_\beta(x)$. // [Choose uniformly at random in case of a tie]

    $u_{\hat{\alpha}}(x) = 1$ and $u_\beta(x) = 0$ if $\beta \neq \hat{\alpha}$.

  end for

end for

Output u.

4.3.4 Learning ω

The MCF, AC, and MBO algorithms are able to yield a good partition of a network, given W, but they do not include a way to find W. A simple way to address this issue is to use an expectation-maximization (EM) algorithm, in which one alternates between an update of g with fixed W and an update of W with fixed g. One can find the optimal W, given g, in closed form by differentiating (4.4) with respect to any component of W and setting the result to 0 [85].

One must be careful, however, because the optimal $W_{\alpha\beta}$ when $\mathrm{Cut}_{g,A}(\alpha, \beta) = 0$ is infinite. This is a problem, because once one of the entries in W is infinite, it prevents g in subsequent iterations from taking any nonzero value of $\mathrm{Cut}_{g,A}(\alpha, \beta)$, which gives bad results in our test examples. (See Section 4.4 for a discussion of these examples.) We address this issue by modifying the EM algorithm to reset all infinite values of W to $1.1 \times W_{\max}$, where $W_{\max}$ is the largest non-infinite element of W.

We also need to address another practical issue for an EM approach to work. Specifically,

the algorithm that we have described thus far in this section often finds bad local minima, in which two communities are merged erroneously or a single community is split inappropriately. To overcome this issue, we implement a wrapper function (see Algorithm 6) that checks each community that is returned by MCF, AC, or MBO for further possible splitting or merging with other communities. Whenever we call MCF, AC, or MBO on a subgraph, we use the values of k and m for the whole graph rather than for a subgraph.[5]

There is also a danger of overfitting by setting $\hat{n} = N$, which gives a likelihood of 1 in (4.2). The proper selection of $\hat{n}$ is a complicated problem, both algorithmically and theoretically [123, 140]. For our tests, we were very successful by using a simple heuristic approach. (However, our framework in this chapter is also compatible with more sophisticated methods for selecting $\hat{n}$.) For each data set, one supplies an expected value of $\hat{n}$ for that data set, and one adds a quadratic penalty to the objective value whenever $\hat{n}$ differs from its expected value. This helps curtail overfitting, while still allowing our algorithms to perform merges and splits to escape bad local minima.

[5] A similar choice was used for the recursive partitioning procedures in [79, 117].

Algorithm 6 The splitting–merging wrapper that we use to escape bad local minima.

Input A, $\hat{n}_{\text{expected}}$.

Place all nodes in the same community and add this community to a queue.

while the queue is not empty **do**

Save $g_{\text{old}} = g$ and $W_{\text{old}} = W$.

Save the current objective value as $\hat{Q}_{\text{old}}$.

Partition the next community in the queue (as a graph in its own right) into $\min(\hat{n}_{\text{expected}}, \sqrt{N})$ communities using MCF, AC, or MBO with $w_{\alpha\beta} = \begin{cases} 1\,, & \alpha = \beta\,, \\ 0.1\,, & \alpha \neq \beta\,. \end{cases}$

while it is possible to improve the objective by merging **do**

Perform the merge that most improves the objective.

end while

if the objective is better than $\hat{Q}_{\text{old}}$ **then**

Add any newly created communities to the queue.

else

Set $g = g_{\text{old}}$ and $W = W_{\text{old}}$.

Remove the current community from the queue.

end if

end while

Output g, W.

4.4 Empirical Results

In this section, we discuss our results from several numerical experiments to (1) confirm that our algorithms can successfully recover g and ω from networks that we generate using SBMs and (2) explore their applicability to real-world networks. In our experiments, we use three different families of SBMs, three Facebook networks (whose community structure is partly understood [151,152]), and an example related to hyperspectral video segmentation. Because

of the random initialization in our approach, we perform three trials on each of the networks for each algorithm, and we report the best result in each case.[6] For comparison, we also report the results of a Kernighan–Lin (KL) algorithm, which was reported in [85] to be effective. We summarize our results in Table 4.1, and we highlight that we consistently recover the underlying structure in the synthetic examples. For the real networks, we compare our results with a reference partition based on metadata that is thought to be correlated with the community structure. We find that the MCF scheme performs the best among our three schemes on these networks, and it finds partitions with a larger likelihood than the reference partition. We implement our methods in MATLAB, so one should interpret our computation time as indicative that the run time is reasonable for networks with millions of edges and perhaps on larger networks, given a careful implementation in a compiled language. For an example of code for a similar problem that was solved by an MBO scheme at large scale, see [105].

We briefly describe the three families of SBM-generated networks that we use in our numerical experiments.

- Planted partition (PP) is a 16,000-node graph that consists of 10 equally-sized communities. It is produced by the method that was described in [85]. It builds a degree-corrected SBM with a truncated power-law degree distribution with exponent 2. The parameter λ from Equation (27) in [85] is 0.001, indicating a fairly clear separation between communities.

- Lancichinetti–Fortunato–Radicchi (LFR) is a standard benchmark SBM network [96]. We construct 1000-node LFR graphs with a power-law degree distribution (with exponent 2), mean degree 20, maximum degree 50, power-law-distributed community sizes

[6] We chose to use three trials to illustrate that our algorithms do not require a large number of attempts to reach a good optimum. In most of our trials, even a single run of the solver is likely to give good results. In Table 4.1, we report our best scores. Our worst scores for MCF are 0.00, 0.00, 0.00, -0.14, and 0.01 for the the PP, MS, LFR, Caltech, and Princeton networks, respectively. We did not record the worst score for Penn. St. or the plume network. Our corresponding worst scores for AC and MBO, respectively, are 0.00, 0.00, 0.01, 0.22, 0.86 and 0.15, 0.00, 0.02, 0.53, 1.12. Comparing these results with Table 4.1, we see that our best and worst scores are often similar to each other.

(with exponent 1), community sizes between 10 and 50 nodes, and mixing parameter 0.1.

- Multiscale SBM (MS). To construct such a graph, we take a union of disjoint components: a 10-clique, a 20-clique, and a sequence of Erdős–Rényi (ER) graphs (drawn from the $G(n, p)$ model with expected mean degree 20) of sizes 40, 80, 160, ..., 5120; there are a total of $10,230$ nodes. We connect the components to each other by adding a single edge, from nodes chosen uniformly at random, between each consecutive clique or ER graph. This construction tests whether an algorithm can find communities of widely varying sizes in the same graph [7, 54].

The hyperspectral video is the same plumes video used in Chapter 3, with 7 frames and the same graph construction procedure. We see from Fig. 4.3 that partitions with small values of (4.4) correspond to meaningful segmentations of the image.

Figure 4.3: Segmentation of a hyperspectral video using graph MCF. The gas plume is clearly represented in the yellow and orange pixels. The two bottom blue classes are the ground, and the other two are the sky. This image is frame 3 of 7.

In Table 4.3, we include an example of a W matrix that we obtain from an MS network to illustrate that we recover different surface tensions between different pairs of communities.[7]

[7]For this example, we have applied the change of variables from Appendix D to eliminate the diagonal elements.

	PP	LFR	MS	Caltech	Princeton	Penn. St.	Plume
Nodes	16,000	1,000	10,230	762	6,575	41,536	284,481
Edges	2.9×10^5	9.8×10^3	1.0×10^5	$16,651$	$293,307$	$1,362,220$	$2,723,840$
Communities	10	40	10	8	4	8	5
MCF	0	0	0	-0.16	-0.02	-0.56	-1.41
AC	0	0	0	0.21	0.58	-0.04	-1.23
MBO	0	0	0	0.53	1.12	0.40	-1.21
KL	0.28	0.03	0.04	-0.16	0.11	-0.55	-1.38
Reference	0	0	0	0	0	0	0

Table 4.1: Results of several tests on several synthetic and empirical networks. We use three surface-tension-based methods (mean-curvature flow, Allen–Cahn, and Merriman–Bence–Osher) and the Kernighan–Lin algorithm from [85] to partition three synthetic networks (Planted Partition, LFR, and Multiscale SBM) and the largest connected components of three empirical networks (Caltech36, Princeton12, and Penn94) from the FACEBOOK100 data set [152]. The score is the difference between the recovered surface-tension energy (4.4) and the corresponding energy of a reference partition, divided by the absolute value of the energy of the reference partition. Smaller values indicate better performance, and 0 corresponds to a partition that is of comparable quality as the reference partition. For the synthetic networks, we use the planted (and hence ground-truth) community structure as the reference partition. For the Facebook networks, we use metadata that is positively correlated with community structure (namely, House affiliation for Caltech and graduation year for the others). For the plume video, our reference partition is to assign all nodes to the same community. The edge counts on the synthetic networks give the order of magnitude, because the exact number differs across realizations.

	PP	LFR	MS	Caltech	Princeton	Penn. State	Plume
MCF	5.36	17.71	3.47	1.39	1.46	38.91	77.91
AC	5.37	26.27	7.28	8.84	480.4	3853	268.7
MBO	4.27	11.05	1.73	0.67	7.43	382.31	270.0
KL	16,566	176	5,117	20	662	95,603	980,520

Table 4.2: Computation times (in seconds).

0	5.22	∞	∞	∞	∞	∞	∞	∞	∞
5.22	0	6.1817	∞	∞	∞	∞	∞	∞	∞
∞	6.1817	0	6.8471	∞	∞	∞	∞	∞	∞
∞	∞	6.8471	0	7.6316	∞	∞	∞	∞	∞
∞	∞	∞	7.6316	0	8.362	∞	∞	∞	∞
∞	∞	∞	∞	8.362	0	9.0869	∞	∞	∞
∞	∞	∞	∞	∞	9.0869	0	9.7926	∞	∞
∞	∞	∞	∞	∞	∞	9.7926	0	10.4911	∞
∞	∞	∞	∞	∞	∞	∞	10.4911	0	11.1869
∞	∞	∞	∞	∞	∞	∞	∞	11.1869	0

Table 4.3: Optimal surface tensions for the MS SBM example. (Note that the entries are heterogeneous, so there are different surface tensions between different pairs of communities.) The infinite entries correspond to sets with no observed edge between them.

CHAPTER 5

Conclusion

This dissertation revisits two popular community-detection frameworks from the perspective of nonlinear, geometric PDEs and associated variational methods. This allowed us to to use novel theoretical, algorithmic, and numerical methods to understand and compute community structure. This can be seen as part of a broader literature seeking to understand the ways in which classical continuum dynamics are relevant to the structural properties of networks (See [15, 149] for reviews). The resulting theorems (such as nonconvexity of modularity, Theorem 1, and Gamma-convergence, Theorems 2 and 3) are unlikely to have been discovered by traditional approaches, and the numerical methods generally have excellent scalability (linear complexity and feasible on networks of millions of nodes or more). Further development of theoretical and algorithmic connections with PDEs has great potential to increase our understanding of network structure and processes.

In Chapter 3, we related modularity optimization to graph cuts and graph total variation. This led to a nonconvexity theorem Theorem 1 that presents a fundamental barrier to future attempts to handle modularity convexly and formalizes the common perception that modularity maximization is a nonconvex problem. We next developed a novel, nonconvex smoothing relaxation that is tunable to allow a reasonable tradeoff between accuracy and ease of navigating the nonconvex landscape. Optimizing this relaxed functional yielded an efficient algorithm with rigorous bounds on the underlying dynamics in Section 3.3.5. The existence of good theoretical bounds reduces the need for parameter-tuning. Using this approach, we effectively computed community structure on hyperspectral image and video examples with up to 29 million edges, as well as a number of other networks without case-by-case parame-

ter tuning. We also showed our approach can incorporate a small amount of supervision to more successfully navigate the nearly degenerate modularity landscape. These theoretical and practical advances develop connections between PDEs, geometry, and graph clustering currently being explored in other parts of the literature (see e.g. [15, 82, 109, 155, 156]), with the broad theme being that popular machine learning, graph analysis, and network science tasks can be solved accurately and efficiently by nonlinear PDE methods. Another direction in which these connections might fruitfully be extended include applications to muliplex and time-dependent graphs, directed graphs, and attributed graphs, which increasingly important as more sophisticated network data becomes available [120].

In Chapter 4, we showed that a particular stochastic block model (SBM) maximum-likelihood estimation problem is equivalent to a discrete version of a well-known surface-tension problem. The equivalence associates graph cuts to surface areas and SBM parameters to physical surface tensions. This gives new geometric and physical interpretations to SBM MLE problems, which are traditionally viewed from a statistical perspective. We used the new connection to adapt three well-known surface-tension-minimization algorithms to community detection in graphs. Our subsequent computations suggest that the resulting algorithms are able to successfully find underlying community structure in SBM-generated graphs. When applied to graphs that are constructed from empirical data, our mean-curvature-flow method performs very well, but the other two methods face some issues (which will be interesting to explore in future studies). We also proved a Γ-convergence result that gives theoretical justification for our algorithms and is the first multiphase Γ-convergence result of which we are aware.

Although the focus of Chapter 4 was a specific form of an SBM and an associated MLE problem, our techniques should also be insightful for other SBM problems. One straightforward adaptation is to consider SBMs without degree correction, although that is more interesting for theoretical work than for applications. Additionally, one can likely incorporate priors on the values of g and ω as regularizers in the surface-tension energy. Another viable extension is to incorporate a small amount of supervision into the community-inference pro-

cess using techniques (such as quadratic fidelity terms) from image processing, similar to the treatment in Chapter 3. It is also important to generalize our approach to more complicated types of networks, such as multilayer [91] and temporal networks [77], and to incorporate metadata [121] into our inference methodology. For example, given our successful results on the hyperspectral video, it may be particularly interesting to use temporal-network clustering to analyze time-dependent communities in the video.

Our results also raise a number of interesting questions. For example, our result on the nonconvexity of modularity is restricted to a common family of embeddings. Are there analogous results that show that the problem is nonconvex regardless of the embedding family? Both the modularity and SBM solvers had different degrees of success depending on the structure of the network. What properties of the networks are responsible for this variability? The modularity solver was robust to different initial conditions in the semi-supervised case. Is the semi-supervised modularity landscape convex in some sense, and if so, are there useful quantitative bounds related to this convexification? Finally, the ϵ parameter in the Allen–Cahn equation is related to the width of the transition layer in Euclidean space—an important theoretical question is whether an analogous notion of transition layer exists in the graph setting, where diffusion distances are inherently more complicated. This has important applications in removing a tunable parameter from our numerical methods, as well as those of several other papers [15].

Approaches such as inference using SBMs and modularity maximization are also related to other approaches for community detection, and the results in this chapter may help further illuminate those connections. These include recent work that relates SBMs to local methods for community detection that are based on personalized PageRank [92] and very recent work that established new connections between modularity maximization and several other approaches [158]. We expect that further mapping of the relations between the diverse available perspectives for community detection (and other problems in network clustering) will yield many new insights for network theory, algorithms, and applications.

Appendix A

Γ-Convergence of the Ginzburg–Landau Approximations of Expressions (3.6) and (4.4)

The following are some fundamental facts about Gamma-convergence to aid in understanding the results of this dissertation. See [44] for more details.

Definition 8. *Let Y be a metric space, and let F_n be a sequence of functionals on Y that take values in $\mathbb{R} \cup \{\infty\} \cup \{-\infty\}$. We say that F_n* Γ-converges *to another functional F if for all $z \in Y$, the following bounds hold:*

1. *(Lower bound) For every sequence $z_n \to z$, we have $F(z) \leq \liminf_{n\to\infty} F_n(z_n)$.*
2. *(Upper bound) For every $z \in Y$, there is a sequence $z_n \to z$ such that $F(z) \geq \limsup_{n\to\infty} F_n(z_n)$.*

For our purposes, Γ-convergence is primarily a tool for ensuring that the minimizers of F_n approach the minimizers of F, as guaranteed by the following:

Theorem 4. *Let F_n Γ-converge to F, and let z_n be a minimizer of F_n. Then every cluster point of the z_n is a minimizer of F. If F is continuous, then $F_n + \mathcal{G}$ Γ-converges to $F + \mathcal{G}$.*

We end with the proof of Theorems 2 and 3.

Proof. We largely follow [155], though we generalize to account for the multiphase nature of our problem.

Observe that all of the terms in (3.6) and (4.4) that do not involve the potential Ψ are continuous and independent of ϵ, so they cannot interfere with the Γ-convergence [44].

Therefore, it suffices to prove that $\frac{1}{\epsilon}\Psi$ Γ-converges to

$$\iota(U) = \begin{cases} 0\,, & \text{if } U \text{ corresponds to a partition}\,, \\ +\infty\,, & \text{otherwise}\,. \end{cases}$$

To prove the lower bound, let $U_n \to U$ and $\epsilon_n \to 0$. If U corresponds to a partition, then $\iota(U) = 0$, which is automatically less than or equal to $\frac{1}{\epsilon_n}\Psi(U_n)$ for each n. If U does not correspond to a partition, then $\iota(U) = +\infty$. Pick N_1 such that whenever $n > N_1$, the distance from U_n to the nearest feasible point is at least $c > 0$. Let $\hat{\Psi}$ be the infimum of Ψ on all of $\mathbb{R}^{N\times\hat{n}}$ except for the balls of radius c that surround each feasible point (so, in particular, $\hat{\Psi} > 0$). It follows that $\liminf_{n\to\infty} \frac{1}{\epsilon_n}\Psi(U_n) \geq \lim_{n\to\infty} \frac{1}{\epsilon_n}\hat{\Psi} = +\infty$. Thus, the lower bound always holds.

To prove the upper bound, let U be any $N \times \hat{n}$ matrix. If U corresponds to a partition, then letting $U_n = U$ for all n gives the required sequence. If U does not correspond to a partition, then $U_n = U$ for all n still satisfies the upper bound.

Thus, both the upper and lower bound requirements hold, and we have proven Γ-convergence for both theorems. $\square$

Appendix B

Deferred proofs from Section 3.3.5

In this section, we give proofs of propositions stated earlier in the dissertation.

Proof of Proposition 2. We first get pointwise estimates on $U - U_0$:

$$\|U - U_0\|_\infty \leq \left\|e^{-\tau M} - I\right\|_\infty \|U_0\|_\infty = \left\|e^{-\tau M} - I\right\|_\infty \leq \sum_{n=1}^{\infty} \frac{1}{n!}\tau^n \|M\|_\infty^n = e^{\tau\|M\|_\infty} - 1 \tag{B.1}$$

We estimate $\|M\|_\infty$ as follows:

$$\begin{aligned}\|M\|_\infty &= \max_i \sum_j |L_{ij} + \frac{\gamma}{m}k_i k_j| = \max_i \sum_j |k_i\delta_{ij} - A_{ij} + \frac{\gamma}{m}k_i k_j| \\ &\leq \max_i k_i + k_i + \frac{\gamma}{m}k_i 2m = 2(1+\gamma)k_{\max}\end{aligned}$$

These computations do not depend on $\hat{n}$, but in order to get a timestep, we assume that $\hat{n} = 2$. We have $U(:,1)_t = -MU(:,1)$ and $U(:,2)_t = -MU(:,2)$. Subtracting these, and letting $v = U(:,1) - U(:,2)$ yields $v_t = -Mv$. Allowing v to evolve until the time of thresholding, we see that node i will switch classes if and only if $v(i)$ has changed sign, that is if $|v - v_0|_i > 1$. The quantity in (B.1) is less than 1 exactly when $\tau < \dfrac{\log 2}{2(\gamma+1)k_{\max}} \approx \dfrac{0.15}{(\gamma+1)k_{\max}}$. This is exactly the bound we sought.

Next, we work on the L^2 bound

$$\|U - U_0\|_\infty \leq \sqrt{\hat{n}}\,\|U - U_0\|_2 \leq \sqrt{\hat{n}}\left\|e^{-\tau M} - I\right\|_2 \|U_0\|_2 \leq \sqrt{\hat{n}}\,\|U_0\|_2 \sum_{n=1}^{\infty}\frac{1}{n!}\tau^n\|M\|_2^n \tag{B.2}$$

$$= \sqrt{\hat{n}}\,\|U_0\|_2\left(e^{\tau\|M\|_2} - 1\right) = \sqrt{\hat{n}}\,\|U_0\|_2\left(e^{\tau R} - 1\right) \tag{B.3}$$

As before, when we let $\hat{n} = 2$, one can subtract the columns to get v, so that no node will switch communities as long as $\|v - v_0\|_\infty < 1$, which is guaranteed if $\tau < R^{-1} \log\left(1 + N^{-\frac{1}{2}}\right)$. $\square$

Proof of Proposition 3. To get the bound, we let Λ be a diagonal matrix with the eigenvalues of M on the diagonal. Since M is positive semi-definite, we can write $M = Q\Lambda Q^T$ for some orthogonal matrix Q. Then we have

$$\|U(\tau)\|_2 = \left\|e^{-\tau M} U_0\right\|_2 \leq \left\|e^{-\tau M}\right\|_2 \|U_0\|_2 = \left\|e^{-\tau \Lambda}\right\|_2 \|U_0\|_2 = e^{-\tau \lambda_1} \|U_0\|_2$$

Setting the latter quantity less than ϵ and then solving for τ yields the required bound. $\square$

Appendix C

Hyperspectral Image Details

In this appendix we collect some basic facts about the images used in Table 3.2.

- Jasper Ridge: An image of a river area. It has 198 channels and 100x100 pixels. Retrieved from `http://www.escience.cn/people/feiyunZHU/Dataset_GT.html`.
- Samson: An image of a coastline. It has 156 channels and 952x952 pixels. Retrieved from `http://www.escience.cn/people/feiyunZHU/Dataset_GT.html`.
- Cuprite: An image of ground near Las Vegas. It has 224 channels and 250x190 pixels. Retrieved from `http://www.escience.cn/people/feiyunZHU/Dataset_GT.html`.
- FLC: A moderate-dimensional image. It has 12 channels and 949x220 pixels. Available at `ftp://www.daba.lv/pub/TIS/atteelu_analiize/MultiSpec/tutorial/ModDimensionDataSet.zip`.
- Pavia U: An image of Pavia University in Northern Italy. It has 103 channels and 610x610 pixels. Retrieved from `http://lesun.weebly.com/hyperspectral-dataset.html`.
- Salinas: An image containing vineyard fields, soils, and vegetation. It has 224 channels and 512x217 pixels. Retrieved from `http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes`.
- Salinas 1: A subimage of the previous image containing 86x83 pixels.

Appendix D

Eliminating the Diagonal Elements of W

It is difficult to determine the parameters $W_{\alpha\alpha}$ in the context of (4.4) and our surface-tension analogy, because they correspond to "internal" surface tensions of a single crystal. In this appendix, we use a change of variables to eliminate these diagonal terms and replace them with additional volume terms, which are much easier to interpret.

We begin with the identity

$$\sum_{\alpha,\beta} W_{\alpha\beta}\mathrm{Cut}_{g,A}(\alpha,\beta) = \sum_{\alpha}\sum_{\beta\neq\alpha} W_{\alpha\beta}\mathrm{Cut}_{g,A}(\alpha,\beta) + \sum_{\alpha} W_{\alpha\alpha}\mathrm{Cut}_{g,A}(\alpha,\alpha)\,, \tag{D.1}$$

and we compute

$$\begin{aligned}
\sum_{\alpha} W_{\alpha\alpha}\mathrm{Cut}_{g,A}(\alpha,\alpha) &= \sum_{\alpha} W_{\alpha,\alpha} \sum_{g_i=\alpha,g_j=\alpha} A_{ij} \\
&= \sum_{\alpha} W_{\alpha,\alpha} \left(\sum_{g_i=\alpha,j=1,\ldots,N} A_{ij} - \sum_{g_i=\alpha,g_j\neq\alpha} A_{ij} \right) \\
&= \sum_{\alpha} W_{\alpha,\alpha} \left(\sum_{g_i=\alpha} k_i - \sum_{\beta\neq\alpha,g_i=\alpha,g_j=\beta} A_{ij} \right) \\
&= \sum_{\alpha} W_{\alpha,\alpha} \left(\mathrm{vol}_{g,A}(\alpha) - \sum_{\beta\neq\alpha} \mathrm{Cut}_{g,A}(\alpha,\beta) \right) .
\end{aligned} \tag{D.2}$$

Combining (D.2) with (D.1) yields

$$\sum_{\alpha,\beta} W_{\alpha\beta}\mathrm{Cut}_{g,A}(\alpha,\beta) = \sum_{\alpha\neq\beta} (W_{\alpha\beta} - W_{\alpha\alpha})\,\mathrm{Cut}_{g,A}(\alpha,\beta) + \sum_{\alpha} W_{\alpha\alpha}\mathrm{vol}_{g,A}(\alpha)\,. \tag{D.3}$$

This formulation has eliminated the diagonal at the cost of making W asymmetric. We can fix this issue by replacing (D.3) with

$$\sum_{\alpha,\beta} W_{\alpha\beta}\mathrm{Cut}_{g,A}(\alpha,\beta) = \sum_{\alpha\neq\beta} \left(W_{\alpha\beta} - \frac{1}{2}W_{\alpha\alpha} - \frac{1}{2}W_{\beta\beta} \right) \mathrm{Cut}_{g,A}(\alpha,\beta) + \sum_{\alpha} W_{\alpha\alpha}\mathrm{vol}_{g,A}(\alpha)$$

$$= \sum_{\alpha \neq \beta} \hat{\sigma}_{\alpha\beta} \text{Cut}_{g,A}(\alpha, \beta) + \sum_{\alpha} W_{\alpha\alpha} \text{vol}_{g,A}(\alpha)\,, \tag{D.4}$$

where $\hat{\sigma}_{\alpha\beta} = W_{\alpha\beta} - \frac{1}{2}W_{\alpha\alpha} - \frac{1}{2}W_{\beta\beta}$. The matrix $\hat{\sigma}$ is symmetric and has 0 values on the diagonal.

Finally, we expand a bit on the role of the volume terms in (4.4). The term

$$\sum_{\alpha} W_{\alpha\alpha} \text{vol}_{g,A}(\alpha) \tag{D.5}$$

is the inner product of the vector of volumes with the diagonal of W. We minimize (D.5), subject to the constraints $\sum_{\alpha} \text{vol}_{g,A}(\alpha) = 2m$ and $\text{vol}_{g,A}(\alpha) \geq 0$, by placing all of the nodes in the community that corresponds to the smallest[1] entry in the diagonal of W. Thus, these terms incentivize placing more mass in the communities with the smallest volume penalty.

[1]When referring to "smallest" eigenvalues in the appendices, we mean the smallest positive or most-negative values rather than those that are smallest in magnitude.

Appendix E

Additional Notes on the AC and MBO Schemes for Expression (4.4)

In this appendix, we discuss some practical details regarding our implementation of the AC and MBO solvers in Chapter 4.

The choice of ϵ in AC is important, because it selects a characteristic scale of the transition. If it is too small, the barrier to transition is large, and no evolution occurs. If it is too large, the transition layer becomes wide enough that a large part of the graph is caught in it, such that U does not approximately correspond to a partition of the graph. Furthermore, Theorem 3 asserts only that the minimizers of (4.4) and (4.9) are related when ϵ is sufficiently small. In our numerical experiments, we set $\epsilon = 0.004$, a choice that we selected by hand-tuning using our synthetic networks. There is no reason to believe that the same value should work for all networks. For example, for the well-known Zachary Karate Club network [168], we obtain much better results for $\epsilon = 0.04$. A very interesting problem is to determine a correct notion of distance and accompanying quantitative estimates to allow an automated selection of ϵ to obtain a transition layer with an appropriate width to give useful results. We discretize the AC equation via convex splitting [51]:

$$(1 + c\,dt)U^{n+1} + LU^{n+1}W = -dt\,\left(cU^n + \nabla T(U^n) + \frac{1}{2m}kk^TUe^{-W}\right),$$

where $c > 2/\epsilon$ [101]. Using the constant c leads to an unconditionally stable scheme, which negates the stiffness caused by the $1/\epsilon$ scale.

It is necessary to solve linear system of the form

$$(1 + c\,dt)U^{n+1} + LU^{n+1}W = \bar{F}^n \tag{E.1}$$

many times. In a continuum setting, one can use a fast Fourier transform, but we do not know of a graph analog with comparable computational efficiency. Instead, we find the $2\hat{n}$ eigenvectors that correspond to the smallest eigenvalues[1] of L and the entire spectrum of W. Consequently, $L \approx V_L D_L V_L^T$ and $W = V_W D_W V_W^T$, and the system (E.1) is approximately equivalent to

$$(1 + c\,dt)V_L^T U^{n+1} V_W + D_L V_L^T U^{n+1} V_W D_W = V_L^T \bar{F}^n V_W\,.$$

Letting $\hat{U}^n = V_L^T U^n V_W$ and $\hat{\bar{F}}^n = V_L^T \bar{F}^n V_W$, we write

$$(1 + c\,dt)\hat{U}^{n+1} + D_L \hat{U}^{n+1} D_W = \hat{\bar{F}}^n\,, \tag{E.2}$$

which is easy to solve for $\hat{U}^{n+1}$. We convert $\hat{U}^{n+1}$ to a solution using $U^{n+1} = V_L \hat{U}^{N+1} V_W^T$. (See [14] for a discussion of this method of recovering U^{n+1} from $\hat{U}^{n+1}$.)

One final detail that we wish to note is that we want the evolution of U to be restricted to have a row sum of 1, so that we can interpret it in terms of probabilities. To do this, we use a vectorized version of the projection algorithm from [35] at each time step.

The MBO solver uses a very similar pseudospectral scheme, although it does not include convex splitting. Unlike in the AC scheme, we need to estimate two time steps automatically in our code, instead of tuning them by hand. The first is the inner-loop step, which we determined using a restriction (which one can show is necessary for stability[2]) that the time step should not exceed twice the reciprocal of the largest eigenvalue of the linear operator that maps U to $\dfrac{1}{m} k k^T U e^{-W}$. The time step between thresholdings of U is given by the reciprocal of the geometric mean of the largest and smallest eigenvalues of the operator that maps $U \to LUW$. The associated intuition is that linear diffusion should have enough time to evolve (to avoid getting stuck) but not enough time to evolve to equilibrium (because the equilibrium does not depend on the initial condition, so it carries no information about it). The reciprocal of the smallest eigenvalue gives an estimate of the time that it takes to

[1] The number $2\hat{n}$ is somewhat arbitrary; we choose it to exceed $\hat{n}$, but for computational convenience, we do not want it to be too large.

[2] See, e.g., [100] for the necessary techniques, which are standard in the numerical analysis of ordinary differential equations.

reach equilibrium, and the reciprocal of the largest eigenvalue gives an estimate of the fastest evolution of the system. We choose the geometric mean between these values to produce a number between these two extremes. References [20] and [156] proved bounds (although in a simpler setting) that support these time-step choices for MBO schemes. Some of these proofs are reproduced in Appendix B.

References

[1] G. Agarwal and D. Kempe, *Modularity-maximizing graph communities via mathematical programming*, Eur. Phys. J. B, 66 (2008).

[2] R. Albert and A.-L. Barabási, *Statistical mechanics of complex networks*, Rev. Mod. Phys., 74 (2002), pp. 47–97.

[3] L. A. N. Amaral, A. Scala, M. Barthélemy, and H. E. Stanley, *Classes of small-world networks*, Proc. Nat. Acad. Sci. USA, 97 (2000), pp. 11149–11152.

[4] L. Ambrodio and V. Tortorelli, *On the approximation of free discontinuity problems*, Boll. Un. Mat. Ital. B, 7 (1992), pp. 105–123.

[5] A. A. Amini, A. Chen, P. J. Bickel, and E. Levina, *Pseudo-likelihood methods for community detection in large sparse networks*, Ann. Statist., 41 (2013), pp. 2097–2122.

[6] C. Anderson, *A Rayleigh-Chebyshev procedure for finding the smallest eigenvalues and associated eigenvectors of large sparse Hermitian matrices*, J. Comput. Phys., 229 (2010), pp. 7477–7487.

[7] A. Arenas, A. Fernández, and S. Gómez, *Analysis of the structure of complex networks at different resolution levels*, New J. Phys., 10 (2008), 053039.

[8] N. W. Ashcroft and N. D. Mermin, *Solid State Physics*, Brooks Cole, Pacific Grove, CA, 1st ed., 1976.

[9] M. Ayati, S. Erten, M. R. Chance, and M. Koyuturk, *MOBAS: Identification of disease-associated protien subnets using modularity-based scoring*, EURASIP J. Bioinf. Sys. Bio., 1 (2015), pp. 1–14.

[10] B. Baingana, G. Mateos, and G. B. Giannakis, *Proximal-gradient algorithms for tracking cascades over social networks*, IEEE J. Selected Topics in Signal Processing, 8 (2014), p. 563.

[11] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–512.

[12] M. J. Barber, *Modularity and community detection in bipartite networks*, Phys. Rev. E, 76 (2007), 066102.

[13] D. S. Bassett, E. T. Owens, M. A. Porter, M. L. Manning, and K. E. Daniels, *Extraction of force-chain network architecture in granular materials using community detection*, Soft Matter, 11 (2015), pp. 2731–2744.

[14] A. L. Bertozzi and A. Flenner, *Diffuse interface models on graphs for classification of high dimensional data*, Multiscale Model. Simul., 10 (2012), pp. 1090–1118.

[15] A. L. Bertozzi and A. Flenner, *Diffuse interface models on graphs for classification of high dimensional data*, SIAM Rev., 58 (2016), pp. 293–328.

[16] A. L. Bertozzi, X. Luo, A. M. Stuart, and K. C. Zygalakis, *Uncertainty quantification in the classification of high dimensional data*, SIAM/ASA J. Uncertain. Quantif., 6 (2018), pp. 568–595.

[17] R. F. Betzel and D. S. Bassett, *Multi-scale brain networks*, NeuroImage, 160 (2017), pp. 73–83.

[18] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, *Fast unfolding of communities in large networks*, J. Stat. Mech. Theory Exp., (2008), P10008.

[19] W. J. Boettinger, J. A. Warren, C. Beckermann, and A. Karma, *Phase-field simulation of solidification*, Ann. Rev. Mater. Res., 32 (2002), pp. 163–194.

[20] Z. Boyd, E. Bae, X.-C. Tai, and A. L. Bertozzi, *Simplified energy landscape for modularity using total variation*, Accepted at SIAM J. Appl. Math. (arXiv:1707.09285), (2018).

[21] Y. Boykov and V. Kolmogorov, *Computing geodesics and minimal surfaces via graph cuts*, in Proceedings of the Ninth IEEE International Conference on Computer Vision, vol. 2 of ICCV '03, Washington, DC, USA, 2003, IEEE Computer Society, pp. 26–33.

[22] Y. Boykov, V. Kolmogorov, D. Cremers, and A. Delong, *An integral solution to surface evolution PDEs via geo-cuts*, in Computer Vision — ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7–13, 2006, Proceedings, Part III, A. Leonardis, H. Bischof, and A. Pinz, eds., Springer-Verlag, Berlin, Germany, 2006, pp. 409–422.

[23] Y. Boykov, O. Veksler, and R. Zabih, *Fast approximate energy minimization via graph cuts*, IEEE Trans. Patt. Anal. Mach. Intel., 23 (2001), pp. 1222–1239.

[24] U. Brandes, D. Delling, M. Gaertler, D. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner, *On modularity clustering*, IEEE Trans. Knowl. Data Eng., 20 (2008), pp. 172–188.

[25] X. Bresson, T. Laurent, D. Uminsky, and J. H. von Brecht, *Multiclass total variation clustering*, arXiv:1306.1185, (2013).

[26] A. Broido and A. Clauset, *Scale free networks are rare*, arXiv:1801.03400, (2018).

[27] A. Buades, B. Coll, and J. M. Morel, *A non-local algorithm for image denoising*, in Computer Vision and Pattern Recognition, vol. 2, 2005, pp. 60–65.

[28] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory, 52 (2006), pp. 489–509.

[29] Cenna, *Grgr3_small.gif*. Wikimedia Commons `https://commons.wikimedia.org/wiki/File:Grgr3d_small.gif`.

[30] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock, *An introduction to total variation for image analysis*, Theor. Found. Numer. Methods Sparse Recovery, 9 (2010), p. 227.

[31] A. Chambolle and J. Darbon, *On total variation minimization and surface evolution using parametric maximum flows*, Int. J. Comput. Vis., 84 (2009), pp. 288–307.

[32] E. Y. Chan and D.-Y. Yeung, *A convex formulation of modularity maximization for community detection*, in Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI), Barcelona, Spain, 2011, pp. 2218–2225.

[33] T. F. Chan and L. A. Vese, *Active contours without edges*, IEEE Trans. on Image Process., (2001).

[34] Y. Chen, X. Li, and J. Xu, *Convexified modularity for degree-corrected stochastic block models*, Accepted at Ann. Stat. (arXiv:1604.03993v1), (2017).

[35] Y. Chen and X. Ye, *Projection onto a simplex*, arXiv:1101.6081, (2011).

[36] F. Chung, *Spectral Graph Theory*, AMS, 1992.

[37] S.-Y. Chung, Y.-S. Chung, and J.-H. Kim, *Diffusion and elastic equations on networks*, Publications of the Research Institute for Mathematical Sciences, 43 (2007), pp. 699–726.

[38] A. Clauset, M. E. J. Newman, and C. Moore, *Finding community structure in very large networks*, Phys. Rev. E, 70 (2004), 066111.

[39] A. Clauset, C. R. Shalizi, and M. E. J. Newman, *Power-law distributions in empirical data*, SIAM Rev., 51 (2009), pp. 661–703.

[40] F. Cleri, S. R. Phillpot, and D. Wolf, *Atomistic simulations of intergranular fracture in symmetric-tilt grain boundaries*, Interface Sci., 7 (1999), pp. 45–55.

[41] A. Condon and R. M. Karp, *Algorithms for graph partitioning on the planted partition model*, Random Structures Algorithms, 18 (2001), pp. 116–140.

[42] G. Csardi and T. Nepusz, *The igraph software package for complex network research*, InterJournal, Complex Systems, (2006), p. 1695, `http://igraph.org`.

[43] P. Csermely, A. London, L.-Y. Wu, and B. Uzzi, *Structure and dynamics of core–periphery networks*, J. Complex Netw., 1 (2013), pp. 93–123.

[44] G. Dal Maso, *An Introduction to Gamma-Convergence*, Birkhäuser, Boston, 1993.

[45] J. Darbon and M. Sigelle, *A fast and exact algorithm for total variation minimization*, in Iberian Conference on Pattern Recognition and Image Analysis, Springer Berlin Heidelberg, 2005, pp. 351–359.

[46] E. Davis and S. Sethuraman, *Consistency of modularity clustering on random geometric graphs*, Accepted at Ann. Appl. Probab. (arXiv:1604.0399v1), (2016).

[47] S. Esedoglu and F. Otto, *Threshold dynamics for networks with arbitrary surface tensions*, Comm. Pure Appl. Math., 68 (2015), pp. 808–864.

[48] S. Esedoglu and Y.-H. R. Tsai, *Threshold dynamics for the piecewise constant Mumford-Shah functional*, J. Comput. Phys., 211 (2006), pp. 367–384.

[49] L. Evans, *Convergence of an algorithm for mean curvature motion*, Indiana Univ. Math. J., 42 (1993), pp. 553–557.

[50] P. Expert, T. S. Evans, V. D. Blondel, and R. Lambiotte, *Uncovering space-independent communities in spatial networks*, Proc. Nat. Acad. Sci. U.S.A, 108 (2011), pp. 7663–7668.

[51] D. J. Eyre, *An unconditionally stable one-step scheme for gradient systems*. `https://www.math.utah.edu/~eyre/research/methods/stable.ps`, 1998.

[52] S. E. Fienberg and S. S. Wasserman, *Categorical data analysis of single sociometric relations*, Sociol. Meth., 12 (1981), pp. 156–192.

[53] S. Fortunato, *Community detection in graphs*, Phys. Rep., 486 (2010), pp. 75–174.

[54] S. Fortunato and M. Barthélemy, *Resolution limit in community detection*, Proc. Natl. Acad. Sci. USA, 104 (2007), pp. 36–41.

[55] S. Fortunato and D. Hric, *Community detection in networks: A user guide*, Phys. Rep., 659 (2016), pp. 1–44.

[56] B. K. Fosdick, D. B. Larremore, J. Nishimura, and J. Ugander, *Configuring random graph models with fixed degree sequences*, In press at SIAM Rev. (arXiv:1608.00607), (2016).

[57] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, *Spectral grouping using the Nyström method*, IEEE Trans. Pattern Anal. Mach. Intell., 24 (2004), pp. 214–225.

[58] C. Fowlkes, S. Belongie, and J. Malik, *Efficient spatiotemporal grouping using the Nyström method*, in CVPR, Hawaii, 2001.

[59] O. Frank and F. Harary, *Cluster inference by using transitivity indices in empirical graphs*, J. Am. Stat. Soc., 77 (1982), pp. 835–840.

[60] H. J. Frost, C. V. Thompson, and D. T. Walton, *Simulation of thin film grain structures I: Grain growth stagnation*, Acta Metall. Mater., 38 (1990), pp. 1455–1462.

[61] H. Gao, M. Guo, R. Li, and L. Xing, *4DCT and 4D cone-beam CT reconstruction using temporal regularization*, in Graphics Processing Unit-Based High Performance Computing in Radiation Therapy, X. Jia and S. B. Jiang, eds., CRC Press, 2015, pp. 63–82.

[62] C. Garcia-Cardona, E. Merkurjev, A. L. Bertozzi, A. Percus, and A. Flenner, *Multiclass segmentation using the Ginzburg-Landau functional and the MBO scheme*, IEEE Trans. Pattern Anal. Mach. Intell., 36 (2014), pp. 1600–1614.

[63] T. Gerhart, J. Sunu, L. Lieu, E. Merkurjev, J.-M. Chang, J. Gilles, and A. L. Bertozzi, *Detection and tracking of gas plumes in LWIR hyperspectral video sequence data*, in SPIE Defense, Security, and Sensing, International Society for Optics and Photonics, 2013, pp. 87430J–87430J.

[64] G. Gilboa, *A total variation spectral framework for scale and texture analysis*, SIAM J. Imag. Sci., 7 (2014), pp. 1937–1961.

[65] G. Gilboa and S. Osher, *Nonlocal operators with applications to image processing*, Multiscale Model. Simul., 7 (2008), pp. 1005–1028.

[66] R. Glowinski, T.-W. Pan, and X.-C. Tai, *Some facts about operator-splitting and alternating direction methods*, in Splitting Methods in Communication, Imaging, Science, and Engineering, R. Glowinski, S. J. Osher, and W. Yin, eds., Springer International Publishing, Cham, 2016, pp. 19–94.

[67] D. Goldfarb and W. Yin, *Second-order cone programming methods for total variation-based image restoration*, SIAM J. Sci. Comput., 27 (2005), pp. 622–645.

[68] T. Goldstein and S. Osher, *The split Bregman method for L1-regularized problems*, SIAM J. Imaging Sci., 2 (2009), pp. 323–343.

[69] B. H. Good, Y.-A. de Montjoye, and A. Clauset, *Performance of modularity maximization in practical contexts*, Phys. Rev. E, 81 (2010), 046106.

[70] H. Grahn and P. Geladi, *Techniques and Applications of Hyperspectral Image Analysis*, John Wiley & Sons, 2007.

[71] R. Guimerè, M. Sales-Pardo, and L. A. N. Amaral, *Modularity from fluctuations in random graphs and complex networks*, Phys. Rev. E, 70 (2004), 025101.

[72] A. Harten, *High resulution schemes for hyperbolic conservation laws*, J. Comput. Phys., 49 (1983), pp. 357–393.

[73] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. Murray, *From molecular to modular cell biology*, Nature, 402 (1999).

[74] R. A. Hegemann, L. M. Smith, A. B. Barbaro, A. L. Bertozzi, S. E. Reid, and G. E. Tita, *Geographical influences of an emerging network of gang rivalries*, Phys. A, 390 (2011), pp. 3894–3914.

[75] P. W. Holland, K. B. Laskey, and S. Leinhardt, *Stochastic blockmodels: First steps*, Social Netw., 5 (1983), pp. 109–137.

[76] P. Holme, *Modern temporal network theory: A colloquium*, Eur. Phys. J. B, 88 (2015), 234.

[77] P. Holme and J. Saramäki, *Temporal networks*, Phys. Rep., 519 (2012), pp. 97–125.

[78] D. Hric, T. P. Peixoto, and S. Fortunato, *Network structure, metadata, and the prediction of missing nodes and annotations*, Physical Review X, 6 (2016), p. 031038.

[79] H. Hu, T. Laurent, M. A. Porter, and A. L. Bertozzi, *A method based on total variation for network modularity optimization using the MBO scheme*, SIAM J. Appl. Math., 73 (2013), pp. 2224–2246.

[80] K. Ikehara and A. Clauset, *Characterizing the structural diversity of complex networks across domains*, arXiv:1710.11304, (2017).

[81] M. Jacobs, *Algorithms for multiphase partitioning*, PhD thesis, University of Michigan, Ann Arbor, 2017.

[82] M. Jacobs, E. Merkurjev, and S. Esedoglu, *Auction dynamics: A volume-constrained MBO scheme*, J. Comp. Phys., 354 (2018), pp. 288–310.

[83] L. G. S. Jeub, P. Balachandran, M. A. Porter, P. J. Mucha, and M. W. Mahoney, *Think locally, act locally: The detection of small, medium-sized, and large communities in large networks*, Phys. Rev. E, 91 (2015), 012821.

[84] I. S. Jutla, L. G. S. Jeub, and P. J. Mucha, *A generalized Louvain method for community detection implemented in* MATLAB. http://netwiki.amath.unc.edu/GenLouvain, 2011.

[85] B. Karrer and M. E. J. Newman, *Stochastic blockmodels and community structure in networks*, Phys. Rev. E, 83 (2011), 016107.

[86] W. O. Kermack and A. G. McKendrick, *Contributions to the mathematical theory of epidemics. ii. —the problem of endemicity*, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 138 (1932), pp. 55–83.

[87] B. W. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, Bell Syst. Tech. J., 49 (1970), pp. 291–307.

[88] M. Kim and J. Leskovec, *Inferring missing nodes and edges in networks*, in Proceedings of the 2011 SIAM International Conference on Data Mining, N. Chawla and W. Wang, eds., 2011, pp. 47–58.

[89] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, *An interior-point method for large-scale l1-regularized least squares*, IEEE J. Sel. Topics in Signal Process., 1 (2007), pp. 606–617.

[90] D. Kinderlehrer, I. Livshits, and S. Ta'asan, *A variational approach to modeling and simulation of grain growth*, SIAM J. Sci. Comput., 28 (2006), pp. 1694–1715.

[91] M. Kivelä, A. Arenas, M. Barthélemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, *Multilayer networks*, J. Complex Netw., 2 (2014), pp. 203–271.

[92] I. M. Kloumann, J. Ugander, and J. Kleinberg, *Block models and personalized PageRank*, Proc. Nat. Acad. Sci. USA, 114 (2017), pp. 33–38.

[93] R. V. Kohn and P. Sternberg, *Local minimizers and singular perturbations*, Proc. Roy. Soc. Edinburgh Sect. A, (1989).

[94] V. Kolmogorov, Y. Boykov, and C. Rother, *Applications of parametric maxflow in computer vision*, in 2007 IEEE 11th International Conference on Computer Vision, IEEE, 2007, pp. 1–8.

[95] A. Lancichinetti and S. Fortunato, *Limits of modularity maximization in community detection*, Phys. Rev. E, 84 (2011), 066122.

[96] A. Lancichinetti, S. Fortunato, and F. Radicchi, *Benchmark graphs for testing community detection algorithms*, Phys. Rev. E., 78 (2008), 056117.

[97] Y. LeCun and C. Cortes, *The MNIST database of handwritten digits*, 1998.

[98] E. A. Leicht and M. E. J. Newman, *Community structure in directed networks*, Phys. Rev. Lett., 100 (2008), 118703.

[99] J. Leskovec, L. A. Adamic, and B. A. Huberman, *The dynamics of viral marketing*, ACM Trans. on the Web, 1 (2007), p. 5.

[100] R. J. LeVeque, *Finite difference methods for differential equations*. `https://pdfs.semanticscholar.org/8ec6/d5e07121fb25213657d89c3bfb523e1e4721.pdf`.

[101] X. Luo and A. L. Bertozzi, *Convergence of the graph Allen–Cahn scheme*, J. Stat. Phys., 167 (2017), pp. 934–958.

[102] D. Manolakis, C. Siracusa, and G. Shaw, *Adaptive matched subspace detectors for hyperspectral imaging applications*, in 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 5, 2001, pp. 3153–3156.

[103] C. Mantegazza, *Lecture Notes on Mean Curvature Flow*, Springer-Verlag, Berlin, Germany, 2011.

[104] N. Masuda, M. A. Porter, and R. Lambiotte, *Random walks and diffusion on networks*, Phys. Rep., 716-717 (2017), pp. 1–58.

[105] Z. Meng, E. Merkurjev, A. Koniges, and A. L. Bertozzi, *Hyperspectral image classification using graph clustering methods*, IPOL J. Image Process. Online, 7 (2017), pp. 218–245.

[106] Z. Meng, J. Sanchez, J.-M. Morel, A. L. Bertozzi, and P. J. Brantingham, *Ego-motion classification for body-worn videos*, in Proceedings of the 2016 Conference on Imaging, Vision, and Learning Based on Optimization and PDEs, 2017.

[107] E. Merkujev, T. Kostic, and A. Bertozzi, *MBO scheme on graphs for segmentation and image processing*, SIAM J. Imaging Sci., 6 (2013), pp. 1903–1930.

[108] E. Merkurjev, E. Bae, A. L. Bertozzi, and X.-C. Tai, *Global binary optimization on graphs for classification of high-dimensional data*, J. Math. Imaging Vision, 52 (2015), pp. 414–435.

[109] E. Merkurjev, A. L. Bertozzi, and F. Chung, *A semi-supervised heat kernel PageRank MBO algorithm for data classification*, Accepted at Comm. Math. Sci., (2018).

[110] E. Merkurjev, J. Sunu, and A. L. Bertozzi, *Graph MBO method for multiclass segmentation of hyperspectral stand-off detection video*, in IEEE International Conference on Image Processing, 2014.

[111] B. Merriman, J. Bence, and S. Osher, *Diffusion generated motion by mean curvature*, Proc. Comput. Crystal Growers Workshop, (1992), pp. 73–83.

[112] L. Modica, *The gradient theory of phase transitions and the minimal interface criterion*, Arch. Ration. Mech. Anal., 98 (1987), pp. 123–142.

[113] C. Moore, *The computer science and physics of community detection: Landscapes, phase transitions, and hardness*, arXiv:1702.00467, (2017).

[114] P. J. Mucha, T. Richardson, K. Macon, M. Porter, and J.-P. Onnela, *Community structure in time-dependent, multiscale, and multiplex networks*, Science, 328 (2010), pp. 876–878.

[115] W. W. Mullins, *Two-dimensional motion of idealized grain boundaries*, J. Appl. Phys., 27 (1956), pp. 900–904.

[116] M. E. J. Newman, *Spread of epidemic disease on networks*, Phys. Rev. E, 66 (2002), 016128.

[117] M. E. J. Newman, *Finding community structure in networks using the eigenvectors of matrices*, Phys. Rev. E, 74 (2006), 036104.

[118] M. E. J. Newman, *Modularity and community structure in networks*, Proc. Nat. Acad. Sci. USA, 103 (2006), pp. 8577–8582.

[119] M. E. J. Newman, *Equivalence between modularity optimization and maximum likelihood methods for community detection*, Phys. Rev. E, 94 (2016), 052315.

[120] M. E. J. Newman, *Networks: an Introduction*, Oxford University Press, 2018.

[121] M. E. J. Newman and A. Clauset, *Structure and inference in annotated networks*, Nature Commun., 7 (2016), 11863.

[122] M. E. J. Newman and M. Girvan, *Finding and evaluating community structure in networks*, Phys. Rev. E, 69 (2004), 026113.

[123] M. E. J. Newman and G. Reinert, *Estimating the number of communities in a network*, Phys. Rev. Lett., 117 (2016), 078301.

[124] A. Noack, *Modularity clustering is force-directed layout*, Phys. Rev. E, 79 (2009), 026102.

[125] O. A. Oleinik, *Discontinuous solutions of nonlinear differential equations*, Uspekhi Mat. Nauk, 12 (1957), pp. 3–73.

[126] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.

[127] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, *A roadmap for the computation of persistent homology*, EPJ Data Sci., 6 (2017), pp. 1–38.

[128] L. Papadopoulos, M. A. Porter, K. E. Daniels, and D. S. Bassett, *Network analysis of particles and grains*, J. Complex Netw., advanced access (arXiv:1708.08080) (2018).

[129] R. Pastor-Satorras and C. Castellano, *Eigenvector localization in real networks and its implications for epidemic spreading*, J. Stat. Phys., (2018), pp. 1–14.

[130] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, *Epidemic processes in complex networks*, Rev. Mod. Phys., 87 (2015), pp. 925–979.

[131] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, *Epidemic processes in complex networks*, Rev. Mod. Phys., 87 (2015), pp. 925–979.

[132] L. Peel, D. B. Larremore, and A. Clauset, *The ground truth about metadata and community detection in networks*, Sci. Adv., 3 (2017), e1602548.

[133] T. P. Peixoto, *Hierarchical block structures and high-resolution model selection in large networks*, Phys. Rev. X, 4 (2014), 011047.

[134] T. P. Peixoto, *Inferring the mesoscale structure of layered, edge-valued, and time-varying networks*, Phys. Rev. E, 92 (2015), 042807.

[135] T. P. Peixoto, *Model selection and hypothesis testing for large-scale network models with overlapping groups*, Phys. Rev. X, 5 (2015), 011033.

[136] T. P. Peixoto, *Bayesian stochastic blockmodeling*, arXiv:1705.10225, (2018). Chapter in "Advances in Network Clustering and Blockmodeling", edited by P. Doreian, V. Batagelj, A. Ferligoj, (John Wiley & Sons, New York City, USA [forthcoming]).

[137] M. A. Porter, P. J. Mucha, M. E. J. Newman, and C. M. Warmbrand, *A network analysis of committees in the U.S. House of Representatives*, Proc. Nat. Acad. Sci. U.S.A., 102 (2005), pp. 7057–7062.

[138] M. A. Porter, J.-P. Onnela, and P. J. Mucha, *Communities in networks*, Notic. Amer. Math. Soc., 56 (2009), pp. 1082–1097.

[139] J. Reichardt and S. Bornholdt, *Statistical mechanics of community detection*, Phys. Rev. E, 74 (2006), 016110.

[140] M. A. Riolo, G. T. Cantwell, G. Reinert, and M. E. J. Newman, *Efficient method for estimating the number of communities in a network*, Phys. Rev. E, 96 (2017), 032310.

[141] P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha, *Core-periphery structure in networks (revisited)*, SIAM Rev., 59 (2017), pp. 619–646.

[142] R. A. Rossi and N. K. Ahmed, *Role discovery in networks*, IEEE Trans. Knowl. Data Eng., 27 (2015), pp. 1112–1131.

[143] A. Roxana Pamfil, S. D. Howison, R. Lambiotte, and M. A. Porter, *Relating modularity maximization and stochastic block models in multilayer networks*, arXiv:1804.01964, (2018).

[144] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation noise removal algorithm*, Phys. D, 60 (1992), pp. 259–268.

[145] S. J. Ruuth, B. Merriman, and S. Osher, *A fixed grid method for capturing the motion of self-intersecting wavefronts and related PDEs*, J. Comput. Phys., 193 (2000), pp. 1–21.

[146] H.-W. Shen, X.-Q. Cheng, and J.-F. Guo, *Quantifying and identifying the overlapping community structure in networks*, J. Stat. Mech. Theory Exp., (2009), P07042.

[147] C. S. Smith, *Metal Interfaces*, American Society for Metals, Cleveland, 1952, ch. Grain shapes and other metallurgical applications of topology, pp. 65–113.

[148] T. A. B. Snijders and K. Nowicki, *Estimation and prediction for stochastic blockmodels for graphs with latent block structure*, J. Classif., 14 (1997), pp. 75–100.

[149] J. Solomon, *PDE approaches to graph analysis*, arXiv:1505.00185, (2015).

[150] S. H. Strogatz, *Exploring complex networks*, Nature, 410 (2001), pp. 268–276.

[151] A. L. Traud, E. D. Kelsic, P. J. Mucha, and M. A. Porter, *Comparing community structure to characteristics in online collegiate social networks*, SIAM Rev., 53 (2011), pp. 526–543.

[152] A. L. Traud, P. J. Mucha, and M. A. Porter, *Social structure of Facebook networks*, Phy. A, 391 (2012), pp. 4165–4180.

[153] N. G. Trillos, D. Slepcev, J. Von Brecht, T. Laurent, and X. Bresson, *Consistency of Cheeger and ratio graph cuts*, J. Mach. Learn. Res., 17 (2016), pp. 1–46.

[154] F. Tudisco, P. Mercado, and M. Hein, *Community detection in networks via nonlinear modularity eigenvectors*, arXiv:1708.05569, (2017).

[155] Y. van Gennip and A. L. Bertozzi, *Gamma-convergence of graph Ginzburg-Landau functionals*, Adv. Differential Equations, 17 (2012), pp. 1115–1180.

[156] Y. van Gennip, N. Guillen, B. Osting, and A. L. Bertozzi, *Mean curvature, threshold dynamics, and phase field theory on finite graphs*, Milan J. Math., 82 (2014), pp. 3–65.

[157] A. Vedaldi and B. Fulkerson, *VLFeat: An open and portable library of computer vision algorithms*. `http://www.vlfeat.org`, 2008.

[158] N. Veldt, D. F. Gleich, and A. Wirth, *A correlation clustering framework for community detection*, in Proceedings of the 2018 World Wide Web Conference, WWW '18, Republic and Canton of Geneva, Switzerland, 2018, International World Wide Web Conferences Steering Committee, pp. 439–448.

[159] B. P. Vollmayr-Lee and A. D. Rutenberg, *Fast and accurate coarsening simulation with an unconditionally stable time step*, Phys. Rev. E, 68 (2003), 032310.

[160] U. von Luxborg, *A tutorial on spectral clustering*, Statist. Comput., 17 (2007), pp. 395–416.

[161] Y. Wang, J. Yang, W. Yin, and Y. Zhang, *A new alternating minimization algorithm for total variation image reconstruction*, SIAM J. Imag. Sci., 1 (2008), pp. 248–272.

[162] D. J. Watts and S. H. Strogatz, *Collective dynamics of "small-world" networks*, Nature, 393 (1998), pp. 440–442.

[163] D. Weaire and J. P. Kermode, *Computer simulation of a two-dimensional soap froth I: Method and motivation*, Phil. Mag. B, 48 (1983), pp. 245–259.

[164] M. Welk, J. Weickert, and G. Gilboa, *A discrete theory and efficient algorithms for forward-and-backward diffusion filtering*, Preprint NI17005, Isaac Newton Institute for Mathematical Sciences, 2017.

[165] X. Yan, C. Shalizi, J. E. Jensen, F. Krzakala, C. Moore, L. Zdeborová, P. Zhang, and Y. Zhu, *Model selection for degree-corrected block models*, J. Stat. Mech. Theory Exp., 5 (2014), 05007.

[166] J. Yang, Y. Zhang, and W. Yin, *A fast alternating direction method for TVL1-L2 signal reconstruction from partial Fourier data*, IEEE J. Sel. Topics in Signal Process., 4 (2010), pp. 288–297.

[167] J. Yuan, E. Bae, and X.-C. Tai, *A study on continuous max-flow and min-cut approaches*, in CVPR, IEEE, 2010, pp. 2217–2224.

[168] W. W. Zachary, *An information flow model for conflict and fission in small groups*, J. Anthropological Res., 33 (1977), pp. 452–473.

[169] L. Zelnik-Manor and P. Perona, *Self-tuning spectral clustering*, in NIPS, 2005, pp. 1601–1608.

[170] P. Zhang and C. Moore, *Scalable detection of statistically significant communities and hierarchies, using message passing for modularity*, Proc. of the Nat. Acad. Sci., 111 (2014), pp. 18144–18149.

[171] W. Zhu, V. Chayes, A. Tiard, S. Sanchez, D. Dahlberg, A. L. Bertozzi, S. Osher, D. Zosso, and D. Kuang, *Unsupervised classification in hyperspectral imagery with nonlocal total variation and primal-dual hybrid gradient algorithm*, IEEE Trans. Geosci. Remote Sensing, 55 (2017), pp. 2786–2798.