SIAM J. APPL. MATH.
Vol. 73, No. 6, pp. 2224–2246

A METHOD BASED ON TOTAL VARIATION FOR NETWORK MODULARITY OPTIMIZATION USING THE MBO SCHEME*

HUIYI HU†, THOMAS LAURENT‡, MASON A. PORTER§, AND ANDREA L. BERTOZZI†

Abstract. The study of network structure is pervasive in sociology, biology, computer science, and many other disciplines. One of the most important areas of network science is the algorithmic detection of cohesive groups of nodes called "communities." One popular approach to finding communities is to maximize a quality function known as *modularity* to achieve some sort of optimal clustering of nodes. In this paper, we interpret the modularity function from a novel perspective: we reformulate modularity optimization as a minimization problem of an energy functional that consists of a total variation term and an ℓ_2 balance term. By employing numerical techniques from image processing and ℓ_1 compressive sensing—such as convex splitting and the Merriman–Bence–Osher (MBO) scheme—we develop a variational algorithm for the minimization problem. We present our computational results using both synthetic benchmark networks and real data.

Key words. social networks, community detection, data clustering, graphs, modularity, MBO scheme

AMS subject classifications. 62H30, 91C20, 91D30, 94C15

DOI. 10.1137/130917387

1. Introduction.

Networks provide a useful representation for the investigation of complex systems, and they have accordingly attracted considerable attention in sociology, biology, computer science, and many other disciplines [51, 52]. Most of the networks that people study are graphs, which consist of nodes (i.e., vertices) to represent the elementary units of a system, and edges to represent pairwise connections or interactions between the nodes.

Using networks makes it possible to examine intermediate-scale structure in complex systems. Most investigations of intermediate-scale structures have focused on *community structure*, in which one decomposes a network into (possibly overlapping) cohesive groups of nodes called *communities* [54].[1] There is a higher density of connections within communities than between them.

In some applications, communities have been related to functional units in networks [54]. For example, a community might be closely related to a functional module in a biological system [39] or a group of friends in a social system [62]. Because community structure can yield important insights in real networks [22, 25, 52, 54], it is

*Received by the editors April 17, 2013; accepted for publication July 31, 2013; published electronically December 17, 2013. This work was supported by UC Lab Fees Research grant 12-LR-236660, ONR grant N000141210838, ONR grant N000141210040, AFOSR MURI grant FA9550-10-1-0569, and NSF grant DMS-1109805.

http://www.siam.org/journals/siap/73-6/91738.html

†Department of Mathematics, UCLA, Los Angeles, CA (huiyihu@math.ucla.edu, bertozzi@math.ucla.edu).

‡Department of Mathematics, University of California, Riverside, Riverside, CA (laurent@math.ucr.edu).

§Oxford Centre for Industrial and Applied Mathematics, Mathematical Institute, and CABDyN Complexity Centre, University of Oxford, Oxford, UK (porterm@maths.ox.ac.uk). This author's work was supported by a research award (220020177) from the James S. McDonnell Foundation, the EPSRC (EP/J001759/1), and the FET-Proactive project PLEXMATH (FP7-ICT-2011-8; grant 317614) funded by the European Commission.

[1]Other important intermediate-scale structures to investigate include core-periphery structure [58] and block models [16].

useful to study algorithmic methods for detecting communities. Such efforts have proved fruitful in studies of the social organization in friendship networks [62], legislation cosponsorships in the United States Congress [64], functional modules in biology networks [28, 39], and many other situations.

To perform community detection, one needs a quantitative definition of what constitutes a community, though this depends on the goal and application that one has in mind. Perhaps the most popular approach is to optimize a quality function known as *modularity* [47, 48, 50], and numerous computational heuristics have been developed for optimizing modularity [22, 54]. The modularity of a network partition measures the fraction of total edge weight within communities versus what one might expect if edges were placed randomly according to some null model. We give a precise definition of modularity in (2.1) in section 2.1. Modularity gives one definition of the "quality" of a partition, and maximizing modularity is supposed to yield a reasonable partitioning of a network into disjoint communities.

Community detection is related to *graph partitioning*, which has been applied to problems in numerous areas (such as data clustering) [41,53,60]. In graph partitioning, a network is divided into disjoint sets of nodes. Graph partitioning usually requires the number of clusters to be specified to avoid trivial solutions, whereas modularity optimization does not require one to specify the number of clusters [54]. This is a desirable feature for applications such as social and biological networks.

Because modularity optimization is an NP-hard problem [7], efficient algorithms are necessary to find good locally optimal network partitions with reasonable computational costs. Numerous methods have been proposed [22, 54]. These include greedy algorithms [12, 49], extremal optimization [6, 17], simulated annealing [29, 33], spectral methods (which use eigenvectors of a modularity matrix) [50, 57], and more. The locally greedy algorithm by Blondel et al. [5] is arguably the most popular computational heuristic; it is a very fast algorithm, and it also yields high modularity values [22, 37].

In this paper, we interpret modularity optimization (using the Newman–Girvan null model [48,52]) from a novel perspective. Inspired by the connection between graph cuts and the total variation (TV) of a graph partition, we reformulate the problem of modularity optimization as a minimization of an energy functional that consists of a graph cut (i.e., TV) term and an ℓ_2 balance term. By employing numerical techniques from image processing and ℓ_1 compressive sensing—such as convex splitting and the Merriman–Bence–Osher (MBO) scheme [44]—we propose a variational algorithm to perform the minimization on the new formula. We apply this method to both synthetic benchmark networks and real data sets, and we achieve performance that is competitive with the state-of-the-art modularity optimization algorithms.

The rest of this paper is organized as follows. In section 2, we review the definition of the modularity function, and we then derive an equivalent formula of modularity optimization as a minimization problem of an energy functional that consists of a TV term and an ℓ_2 balance term. In section 3, we explain the MBO scheme and convex splitting, which are numerical schemes that we employ to solve the minimization problem in section 2. In section 4, we test our algorithms on several benchmark and real-world networks. We then review the similarity measure known as the *normalized mutual information* (NMI) and use it to compare network partitions with ground-truth partitions. We also evaluate the speed of our method, which we compare to classic spectral clustering [41, 60], modularity-based spectral partitioning [50, 57], and the GenLouvain code [32] (which is an implementation of a Louvain-like algorithm [5]).

In section 5, we summarize and discuss our results.

2. Method. Consider an N-node network, which we can represent as a weighted graph (G,E) with a node set $G = \{n_1, n_2, \dots, n_N\}$ and an edge set $E = \{w_{ij}\}_{i,j=1}^{N}$. The quantity w_{ij} indicates the closeness (or similarity) of the tie between nodes n_i and n_j, and the array of all w_{ij} values forms the graph's *adjacency matrix* $\mathbf{W} = [w_{ij}]$. In this work, we consider only undirected networks, so $w_{ij} = w_{ji}$.

2.1. Review of the modularity function. The modularity of a graph partition measures the fraction of total edge weight within each community minus the edge weight that would be expected if edges were placed randomly using some null model [54]. The most common null model is the Newman–Girvan (NG) model [48], which assigns the expected edge weight between n_i and n_j to be $\frac{k_i k_j}{2m}$, where $k_i = \sum_{s=1}^{N} w_{is}$ is the strength (i.e., weighted degree) of n_i and $2m = \sum_{i=1}^{N} k_i$ is the total volume (i.e., total edge weight) of the graph (G,E). When a network is unweighted, then k_i is the degree of node i. The NG null model preserves the expected strength distribution of the network.

A *partition* $g = \{g_i\}_{i=1}^{N}$ of the graph (G,E) consists of a set of disjoint subsets of the node set G whose union is the entire set G. The quantity $g_i \in \{1, 2, \dots, \hat{n}\}$ is the community assignment of n_i, where there are $\hat{n}$ communities ($\hat{n} \leq N$). The *modularity* of the partition g is defined as

$$Q(g) = \frac{1}{2m} \sum_{i,j=1}^{N} \left(w_{ij} - \gamma \frac{k_i k_j}{2m} \right) \delta(g_i, g_j)\,, \tag{2.1}$$

where γ is a resolution parameter [56]. The term $\delta(g_i, g_j) = 1$ if $g_i = g_j$, and $\delta(g_i, g_j) = 0$ otherwise. The resolution parameter can change the scale at which a network is clustered [22, 54]. A network breaks into more communities as one increases γ.

By maximizing modularity, one expects to obtain a reasonable partition of a network. However, this maximization problem is NP-hard [7], so considerable effort has been put into the development of computational heuristics for obtaining network partitions with high values of Q.

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

We start by defining the *total variation* (TV), weighted ℓ_2-norm, and weighted mean of a function $f : G \to \mathbb{R}$:

$$\begin{aligned} |f|_{TV} &:= \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} \left| f_i - f_j \right| \,, \\ \|f\|_{\ell_2}^2 &:= \sum_{i=1}^{N} k_i \left| f_i \right|^2 \,, \\ \text{mean}(f) &:= \frac{1}{2m} \sum_{i=1}^{N} k_i f_i \,, \end{aligned} \tag{2.2}$$

where $f_i = f(n_i)$. The quantity $\frac{1}{2}\sum_{i,j=1}^{N} w_{ij}|f_i - f_j|$ is called the TV because it enjoys many properties of the classical TV $\int |\nabla f|$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ [11]. For a vector-valued function $f = (f^{(1)}, \dots, f^{(\hat{n})})\colon G \to \mathbb{R}^{\hat{n}}$, we define

$$|f|_{TV} := \sum_{l=1}^{\hat{n}} |f^{(l)}|_{TV}\,,$$

$$\|f\|_{\ell_2}^2 := \sum_{l=1}^{\hat{n}} \|f^{(l)}\|_{\ell_2}^2\,, \tag{2.3}$$

and $\operatorname{mean}(f) := \big(\operatorname{mean}(f^{(1)}), \dots, \operatorname{mean}(f^{(\hat{n})})\big)$.

Given the partition $g = \{g_i\}_{i=1}^N$ defined in section 2.1, let $A_l = \{n_i \in G, g_i = l\}$, where $l \in \{1, 2, \dots, \hat{n}\}$ ($\hat{n} \leq N$). Thus, $G = \cup_{l=1}^{\hat{n}} A_l$ is a partition of the network (G, E) into disjoint communities. Note that A_l is allowed to be empty, so g is a partition into *at most* $\hat{n}$ communities. Let $f^{(l)} : G \to \{0, 1\}$ be the indicator function of community l; in other words, $f_i^{(l)}$ equals 1 if $g_i = l$, and it equals 0 otherwise. The function $f = (f^{(1)}, \dots, f^{(\hat{n})})$ is then called the *partition function* (associated with g). Because each set A_l is disjoint from all of the others, it is guaranteed that only a single entry of f_i equals 1 for any node i. Therefore, $f : G \to V^{\hat{n}} \subset \mathbb{R}^{\hat{n}}$, where

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

is the standard basis of $\mathbb{R}^{\hat{n}}$.

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

THEOREM 2.1. *Maximizing the modularity functional Q over all partitions that have at most $\hat{n}$ communities is equivalent to minimizing*

$$|f|_{TV} - \gamma \|f - \operatorname{mean}(f)\|_{\ell_2}^2 \tag{2.4}$$

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

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

$$\begin{aligned} Q(g) &= \frac{1}{2m}\left[\left(2m - \sum_{g_i \neq g_j} w_{ij}\right) - \frac{\gamma}{2m}\sum_{l=1}^{\hat{n}}\left(\sum_{n_i \in A_l, n_j \in A_l} k_i k_j\right)\right] \\ &= 1 - \frac{1}{2m}\left(\sum_{l=1}^{\hat{n}} \operatorname{Cut}(A_l, A_l^c) + \frac{\gamma}{2m}\sum_{l=1}^{\hat{n}} \operatorname{vol}(A_l)^2\right) \\ &= 1 - \gamma - \frac{1}{2m}\left(\sum_{l=1}^{\hat{n}} \operatorname{Cut}(A_l, A_l^c) - \frac{\gamma}{2m}\left(\sum_{l=1}^{\hat{n}} \operatorname{vol}(A_l) \cdot \operatorname{vol}(A_l^c)\right)\right)\,, \end{aligned} \tag{2.5}$$

where the sum $\sum_{g_i \neq g_j} w_{ij}$ includes both w_{ij} and w_{ji}. Note that if $\chi_A : G \to \{0, 1\}$ is the indicator function of a subset $A \subset G$, then $|\chi_A|_{TV} = \operatorname{Cut}(A, A^c)$ and

$$\begin{aligned} \|\chi_A - \operatorname{mean}(\chi_A)\|_{\ell_2}^2 &= \sum_{i=1}^{N} k_i \left|\chi_A(n_i) - \frac{\operatorname{vol}(A)}{2m}\right|^2 \\ &= \operatorname{vol}(A)\left(1 - \frac{\operatorname{vol}(A)}{2m}\right)^2 + \operatorname{vol}(A^c)\left(\frac{\operatorname{vol}(A)}{2m}\right)^2 \\ &= \frac{\operatorname{vol}(A) \cdot \operatorname{vol}(A^c)}{2m}\,. \end{aligned}$$

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

$$|f|_{TV} - \gamma\|f - \mathrm{mean}(f)\|_{\ell_2}^2 = \sum_{l=1}^{\hat{n}} \left\{ |f^{(l)}|_{TV} - \gamma\|f^{(l)} - \mathrm{mean}(f^{(l)})\|_{\ell_2}^2 \right\}$$

$$= \sum_{l=1}^{\hat{n}} \left\{ \mathrm{Cut}(A_l, A_l^c) - \gamma \frac{\mathrm{vol}(A_l) \cdot \mathrm{vol}(A_l^c)}{2m} \right\} . \tag{2.6}$$

Combining (2.5) and (2.6), we conclude that maximizing Q is equivalent to minimizing (2.4). □

With the above argument, we have reformulated the problem of modularity maximization as the minimization problem (2.4), which corresponds to minimizing the TV of the function f along with a balance term. This yields a novel view of modularity optimization that uses the perspective of compressive sensing (see the references in [40]). In the context of compressive sensing, one seeks a function f that is compressible under the transform of a linear operator Φ. That is, we want Φf to be well approximated by sparse functions. (A function is considered to be "sparse" when it is equal to or approximately equal to zero on a "large" portion of the whole domain.) Minimizing $\|\Phi f\|_{\ell^1}$ promotes sparsity in Φf. When Φ is the gradient operator (on a continuous domain) or the finite-differencing operator (on a discrete domain) ∇, then the object $\|\Phi f\|_{\ell^1} = \|\nabla f\|_{\ell^1}$ becomes the total variation $|f|_{TV}$ [40,46]. The minimization of TV is also common in image processing and computer vision [10,40,46,59].

The expression in (2.5) is interesting because its geometric interpretation of modularity optimization contrasts with existing interpretations (e.g., probabilistic ones [35] or in terms of the Potts model from statistical physics [56]). For example, we see from (2.5) that finding the bipartition of the graph $G = A \cup A^c$ with maximal modularity is equivalent to minimizing

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

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

$$\frac{\mathrm{Cut}\,(A, A^c)}{\mathrm{vol}(A) \cdot \mathrm{vol}\,(A^c)} . \tag{2.7}$$

In recent papers [8,9,30,31,55,61], various TV-based algorithms were proposed to minimize ratios similar to (2.7).

3. Algorithm.

Directly optimizing (2.4) over all partition functions $f : G \to V^{\hat{n}}$ is difficult due to the discrete solution space. A continuous relaxation is thus needed to simplify the optimization problem.

3.1. Ginzburg–Landau relaxation of the discrete problem.

Let

$$X^p = \{f \mid f : G \to V^{\hat{n}}\}$$

denote the space of partition functions. Minimizing (2.4) over X^p is equivalent to minimizing

$$H(f) = \begin{cases} |f|_{TV} - \gamma\|f - \mathrm{mean}(f)\|_{\ell_2}^2 & \text{if } f \in X^p, \\ +\infty & \text{otherwise} \end{cases} \tag{3.1}$$

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

The Ginzburg–Landau (GL) functional has been used as an alternative for the TV term in image processing (see the references in [4]) due to its Γ-convergence to the TV of the characteristic functions in Euclidean space [34]. Reference [4] developed a graph version of the GL functional and used it for graph-based high-dimensional data segmentation problems. The authors of [23] generalized the two-phase graphical GL functional to a multiphase one.

For a graph (G, E), the (combinatorial) *graph Laplacian* [11] is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \tag{3.2}$$

where $\mathbf{D}$ is a diagonal matrix with node strengths $\{k_i\}_{i=1}^N$ on the diagonal and $\mathbf{W}$ is the weighted adjacency matrix. The operator $\mathbf{L}$ is linear on $\{z|z : G \to \mathbb{R}\}$ and satisfies

$$\langle z, \mathbf{L}z \rangle = \frac{1}{2}\sum_{i,j} w_{ij}(z_i - z_j)^2\,,$$

where $z_i = z(n_i)$ and $i \in \{1, 2, \ldots, N\}$.

Following the idea in [4, 23], we define the *GL relaxation* of H as follows:

$$H_\epsilon(f) = \frac{1}{2}\sum_{l=1}^{\hat{n}}\langle f^{(l)}, \mathbf{L}f^{(l)}\rangle + \frac{1}{\epsilon^2}\sum_{i=1}^{N} W_{\text{multi}}(f_i) - \gamma\|f - \text{mean}(f)\|_{\ell_2}^2\,, \tag{3.3}$$

where $\epsilon > 0$. In (3.3), $W_{\text{multi}} : \mathbb{R}^{\hat{n}} \to \mathbb{R}$ is a multiwell potential (see [23]) with equal-depth wells. The minima of W_{multi} are spaced equidistantly, take the value 0, and correspond to the points of $V^{\hat{n}}$. The specific formula for W_{multi} does not matter for the present paper, because we will discard it when we implement the MBO scheme. Note that the purpose of this multiwell term is to force f_i to go to one of the minima, so that one obtains an approximate phase separation.

Our next theorem states that modularity optimization with an upper bound on the number of communities is well approximated (in terms of Γ-convergence) by minimizing H_ϵ over all $f : G \to \mathbb{R}^{\hat{n}}$. Therefore, the *discrete* modularity optimization problem (2.4) can be approximated by a *continuous* optimization problem. We give the mathematical definition and relevant proofs of Γ-convergence in the appendix.

Theorem 3.1 (Γ-convergence of H_ϵ towards H). *The functional H_ϵ Γ-converges to H on the space $X = \{f \mid f : G \to \mathbb{R}^{\hat{n}}\}$.*

Proof. As shown in Theorem A.2 (in the appendix), $H_\epsilon + \gamma\|f - \text{mean}(f)\|_{\ell_2}^2$ Γ-converges to $H + \gamma\|f - \text{mean}(f)\|_{\ell_2}^2$ on X. Because $\gamma\|f - \text{mean}(f)\|_{\ell_2}^2$ is continuous on the metric space X, it is straightforward to check that H_ϵ also Γ-converges to H according to the definition of Γ-convergence. ∎

By definition of Γ-convergence, Theorem 3.1 directly implies the following.

Corollary 3.2. *Let f^ϵ be the global minimizer of H_ϵ. Any convergent subsequence of f^ϵ then converges to a global maximizer of the modularity Q with at most $\hat{n}$ communities.*

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

In [44], an efficient algorithm (which is now called the *MBO scheme*) was proposed to approximate the gradient descent of the GL functional using threshold dynamics.

See [2, 18, 20] for discussions of the convergence of the MBO scheme. Inspired by the MBO scheme, the authors of [19] developed a method using a PDE framework to minimize the piecewise-constant Mumford–Shah functional (introduced in [45]) for image segmentation. Their algorithm was motivated by the Chan–Vese level-set method [10] for minimizing certain variants of the Mumford–Shah functional. Note that the Chan–Vese method is related to our reformulation of modularity, because it uses the TV as a regularizer along with ℓ_2-based fitting terms. The authors of [23, 43] applied the MBO scheme to graph-based problems.

The gradient-descent equation of (3.3) is

$$\frac{\partial f}{\partial t} = -(\mathbf{L}f^{(1)}, \ldots, \mathbf{L}f^{(\hat{n})}) - \frac{1}{\epsilon^2}\nabla W_{\text{multi}}(f) + \frac{\delta}{\delta f}\left(\gamma\|f - \text{mean}(f)\|^2_{\ell_2}\right), \tag{3.4}$$

where $\nabla W_{\text{multi}}(f) : G \to \mathbb{R}^{\hat{n}}$ is the composition of the functions ∇W_{multi} and f. Thus, one can follow the idea of the original MBO scheme to split (3.4) into two parts and replace the forcing part $\frac{\partial f}{\partial t} = -\frac{1}{\epsilon^2}\nabla W_{\text{multi}}(f)$ by an associated thresholding.

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

Step 1.

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

$$\frac{\partial f}{\partial t} = -(\mathbf{L}f^{(1)}, \ldots, \mathbf{L}f^{(\hat{n})}) + \frac{\delta}{\delta f}\left(\gamma\|f - \text{mean}(f)\|^2_{\ell_2}\right). \tag{3.5}$$

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

Step 2.

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

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

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

To solve (3.5), we implement a *convex-splitting* scheme [21, 63]. Equation (3.5) is the gradient flow of the energy $H_1 + H_2$, where $H_1(f) := \frac{1}{2}\sum_{l=1}^{\hat{n}}\langle f^{(l)}, \mathbf{L}f^{(l)}\rangle$ is convex and $H_2(f) := -\gamma\|f - \text{mean}(f)\|^2_{\ell_2}$ is concave. In a discrete time-stepping scheme, the convex part is treated implicitly in the numerical scheme, whereas the concave part is treated explicitly. Note that the convex-splitting scheme for gradient-descent equations is an unconditionally stable time-stepping scheme.

The discretized time-stepping formula is

$$\begin{aligned}\frac{\hat{f} - f^n}{\tau_n} &= -\frac{\delta H_1}{\delta f}(\hat{f}) - \frac{\delta H_2}{\delta f}(f^n)\\ &= -(\mathbf{L}\hat{f}^{(1)}, \ldots, \mathbf{L}\hat{f}^{(\hat{n})}) + 2\gamma\vec{k}\odot(f^n - \text{mean}(f^n))\,,\end{aligned} \tag{3.6}$$

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

$$\Big((1 + \tau_n\mathbf{L})\hat{f}^{(1)}, \ldots, (1 + \tau_n\mathbf{L})\hat{f}^{(\hat{n})}\Big) = f^n + 2\gamma\tau_n\vec{k}\odot(f^n - \text{mean}(f^n))\,. \tag{3.7}$$

For the purpose of computational efficiency, we utilize the low-order (leading) eigenvectors (associated with the smallest eigenvalues) of the graph Laplacian $\mathbf{L}$ to approximate the operator **L**. The eigenvectors with higher order are more oscillatory and resolve finer-scale features. Leading eigenvectors provide a basis set to approximately represent graph functions. We can resolve progressively finer scales by using more leading eigenvectors. In the graph-clustering literature, scholars usually use a small fraction of the leading eigenvectors of **L** to find useful structural information in a graph [3, 11, 13, 50, 60]. (Note, however, that some recent work has explored the use of other eigenvectors [14].) In contrast, one typically uses many more modes when solving partial differential equations numerically (e.g., consider a pseudospectral scheme), because one needs to resolve the solution at much finer scales.

Motivated by the known utility and many successes of using leading eigenvectors (and discarding higher-order eigenvectors) in studying graph structure, we project f onto the space of the N_{eig} leading eigenvectors to approximately solve (3.7). Assume that $f^n = \sum_s \phi_s \mathbf{a}_s^n$, $\hat{f} = \sum_s \phi_s \hat{\mathbf{a}}_s$, and $2\gamma\tau_n \vec{k} \odot (f^n - \text{mean}(f^n)) = \sum_s \phi_s \mathbf{b}_s^n$, where $\{\lambda_s\}$ are the N_{eig} smallest eigenvalues of the graph Laplacian **L**. We denote the corresponding eigenvectors (eigenfunctions) by $\{\phi_s\}$. Note that $\mathbf{a}_s^n$, $\hat{\mathbf{a}}_s$, and $\mathbf{b}_s^n$ all belong to $\mathbb{R}^{\hat{n}}$. With this representation, we obtain

$$\hat{\mathbf{a}}_s = \frac{\mathbf{a}_s^n + \mathbf{b}_s^n}{1 + \tau_n \lambda_s}\,, \quad s \in \{1, 2, \ldots, N_{\text{eig}}\} \tag{3.8}$$

from (3.7). We are thereby able to solve (3.7) very efficiently.

We summarize our modularity MBO scheme in Algorithm 1. Note that the time complexity of each MBO iteration step is $O(N)$. Unless specified otherwise, the

Algorithm 1. The modularity MBO scheme.

```
Set values for $\gamma$, $\hat{n}$, $\eta$, and $\tau_n = dt$.
Input $\leftarrow$ an initial function $f^0 : G \to V^{\hat{n}}$ and the eigenvalue-eigenvector pairs
$\{(\lambda_s, \phi_s)\}$ of the graph Laplacian $\mathbf{L}$ corresponding to the $N_{\text{eig}}$ smallest eigenvalues.
Initialize:
$\mathbf{a}_s^0 = \langle f^0, \phi_s \rangle$;
$\mathbf{b}_s^0 = \langle 2\gamma dt\vec{k} \odot (f^0 - \text{mean}(f^0)), \phi_s \rangle$.
while $f^n \neq f^{n-1}$ and $n \leq 500$: do
  Diffusion:
  for $i = 1 \to \eta$ do
    $\mathbf{a}_s^n \leftarrow \frac{\mathbf{a}_s^n + \mathbf{b}_s^n}{1 + dt\lambda_s}$ for $s \in \{1, 2, \ldots, N_{\text{eig}}\}$;
    $f^n \leftarrow \sum_s \phi_s \mathbf{a}_s^n$;
    $\mathbf{b}_s^n = \langle 2\gamma dt\vec{k}.*(f^n - \text{mean}(f^n)), \phi_s \rangle$;
    $i = i + 1$;
  end for
  Thresholding:
    $f_i^{n+1} = \vec{e}_{g_i} \in V^{\hat{n}},$ where $g_i = \text{argmax}_{\{1 \leq l \leq \hat{n}\}} \{\hat{f}_i^{(l)}\}$.
  $n = n + 1$;
end while
Output $\leftarrow$ the partition function $f^n$.
```

numerical experiments in this paper use a random initial function f^0. (It takes its value in $V^{\hat{n}}$ with uniform probability by using the command `rand` in MATLAB.)

3.3. Two implementations of the modularity MBO scheme. Given an input value of the parameter $\hat{n}$, the modularity MBO scheme partitions a graph into at most $\hat{n}$ communities. In many applications, however, the number of communities is usually not known in advance [22, 54], so it can be difficult to decide what values of $\hat{n}$ to use. Accordingly, we propose two implementations of the modularity MBO scheme. The *recursive modularity MBO (RMM)* scheme is particularly suitable for networks in which one expects a large number of communities, whereas the *multiple input-$\hat{n}$ modularity MBO (multi-$\hat{n}$ MM)* scheme is particularly suitable for networks in which one expects a small number of communities.

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

Importantly, the minimization problem (2.4) needs a slight adjustment for the recursion steps. Assume for a particular recursion step that we perform the modularity MBO partitioning with parameter $\hat{n}$ on a network $S \subset G$ containing a subset of the nodes of the original graph. Our goal is to increase the modularity for the global network instead of the subnetwork S. Hence, the target energy to minimize is

$$H^{(S)}(f) := |f|_{TV}^{(S)} - \gamma \frac{m^{(S)}}{m} \left\| f - \mathrm{mean}^{(S)}(f) \right\|_{\ell_2}^2 ,$$

where $f : S \to V^{\hat{n}} \subset \mathbb{R}^{\hat{n}}$, the TV norm is $|f|_{TV}^{(S)} = \frac{1}{2} \sum_{i,j \in S} w_{ij} |f_i - f_j|_{\ell_1}$, the total edge weight of S is $2m^{(S)} = \sum_{i \in S} k_i$, and $\mathrm{mean}^{(S)}(f) = \frac{1}{2m^{(S)}} \sum_{i \in S} k_i f_i$. The rest of the minimization procedures are the same as described previously.

Note that the eigenvectors of the Laplacian of the successive subnetworks need to be recalculated for each recursive step. However, because the scales that get resolved become finer as the recursion progresses, the number of eigenvectors N_{eig} that are calculated for each subnetwork need not be very large.

Implementation 2. For the multi-$\hat{n}$ MM scheme, one sets a search range T for $\hat{n}$, runs the modularity MBO scheme for each $\hat{n} \in T$, and then chooses the resulting partition with the highest modularity score. It works well if one knows the approximate maximum number of communities and if that number is reasonably small. One can then set the search range T to be all integers between 2 and the maximum number. Even though the multi-$\hat{n}$ MM scheme allows partitions with fewer than $\hat{n}$ clusters, it is still necessary to include small values of $\hat{n}$ in the search range to better avoid local minima. (See the discussion of the MNIST "4-9" digits network in section 4.2.1.) For different values of $\hat{n}$, one can reuse the previously computed eigenvectors because $\hat{n}$ does not affect the graph Laplacian. Inputting multiple choices for the random initial function f^0 (as described at the end of section 3) also helps to reduce the chance of getting stuck in a minimum and thereby helps to achieve a good optimal solution for the modularity MBO scheme. Because this initial function is used after the computation of eigenvectors, it takes only a small amount of time to rerun the MBO steps.

In section 4, we test these two schemes on several real and synthetic networks.

4. Numerical results.

In this section, we present the numerical results of experiments that we conducted using both synthetic and real network data sets. Unless otherwise specified, our modularity MBO schemes are all implemented in MATLAB.

(Our MATLAB code is not optimized for speed.) In the following tests, we set the parameters of the modularity MBO scheme to be $\eta = 5$ and $\tau_n = 1$.

4.1. LFR benchmark. In [36], Lancichinetti, Fortunato, and Radicchi (LFR) introduced an eponymous class of synthetic benchmark graphs to provide tougher tests of community-detection algorithms than previous synthetic benchmarks. Many real networks have heterogeneous distributions of node degree and community size, so the LFR benchmark graphs incorporate such heterogeneity. They consist of unweighted networks with a predefined set of nonoverlapping communities. As described in [36], each node is assigned a degree from a power-law distribution with power ξ; additionally, the maximum degree is given by $k_{\max}$, and the mean degree is $\langle k \rangle$. Community sizes in LFR graphs follow a power-law distribution with power β, subject to the constraint that the sum of the community sizes must equal the number of nodes N in the network. Each node shares a fraction $1-\mu$ of its edges with nodes in its own community and a fraction μ of its edges with nodes in other communities. (The quantity μ is called the *mixing parameter*.) The minimum and maximum community sizes, $q_{\min}$ and $q_{\max}$, are also specified. We label the LFR benchmark data sets by $(N, \langle k \rangle, k_{\max}, \xi, \beta, \mu, q_{\min}, q_{\max})$. The code used to generate the LFR data is publicly available, provided by the authors in [36].

The LFR benchmark graphs have become a popular choice for testing community-detection algorithms, and [37] used them to test the performance of several community-detection algorithms. The authors concluded, for example, that the locally greedy Louvain algorithm [5] is one of the best heuristics for maximizing modularity. They based their conclusion on the evaluation of the *normalized mutual information* (NMI) (discussed later in this section). Note that the time complexity of this Louvain algorithm is $O(M)$ [22], where M is the number of nonzero edges in the network. In our tests, we use the GenLouvain code (in MATLAB) from [32]; this is an implementation of a Louvain-like algorithm. The GenLouvain code is a modification of the Louvain locally greedy algorithm [5], but it was not designed to be optimal for speed. We implement our RMM scheme on the LFR benchmark, and we compare our results with those from running the GenLouvain code. We use the recursive version of the modularity MBO scheme because the LFR networks that we use contain about $0.04N$ communities.

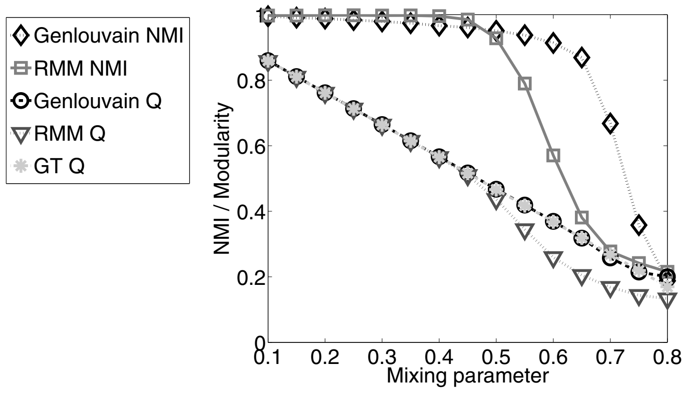
We implement the modularity-optimization algorithms on several sets of LFR benchmark data. We then compare the resulting partitions with the known community assignments of the benchmarks (i.e., the ground truth) by computing NMI [15].

NMI is a similarity measure for comparing two partitions based on information entropy, and it is often used for testing community-detection algorithms [36, 37]. The NMI equals 1 when two partitions are identical, and it has an expected value of 0 when they are independent. For an N-node network with two partitions, $C = \{C_1, C_2, \ldots, C_K\}$ and $\hat{C} = \{\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_{\hat{K}}\}$, that consist of nonoverlapping communities, the NMI is

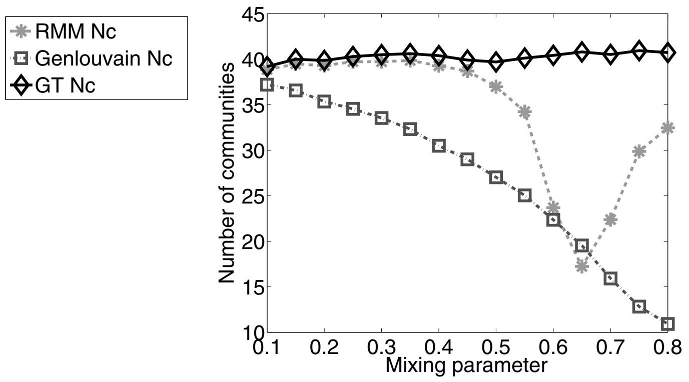
$$\text{NMI}(C,\hat{C}) = \frac{2\sum_{k=1}^{K}\sum_{\hat{k}=1}^{\hat{K}} P(k,\hat{k})\log\left[\frac{P(k,\hat{k})}{P(k)P(\hat{k})}\right]}{-\sum_{k=1}^{K} P(k)\log\left[P(k)\right] - \sum_{\hat{k}=1}^{\hat{K}} P(\hat{k})\log\left[P(\hat{k})\right]}, \tag{4.1}$$

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

We examine two types of LFR networks. One is the 1000-node ensembles used



(a) NMI and modularity (Q).



(b) Number of communities (N_c).

FIG. 4.1. *Tests on LFR1k networks with RMM and GenLouvain. The ground-truth communities are denoted by GT.*

in [37]:

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

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

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

In Figure 4.1, we plot the mean maximized modularity score (Q), the number of communities (N_c), and the NMI of the partitions compared with the ground truth (GT) communities as a function of the mixing parameter μ. As one can see from panel (a), the RMM scheme performs very well for $\mu < 0.5$. Both its NMI score and modularity score are competitive with the results of GenLouvain. However, for $\mu \geq 0.5$, its performance drops with respect to both NMI and the modularity scores of its network partitions. From panel (b), we see that RMM tends to give partitions with more communities than GenLouvain, and this provides a better match to the ground truth. However, it is only trustworthy for $\mu < 0.5$, when its NMI score is very close to 1.

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

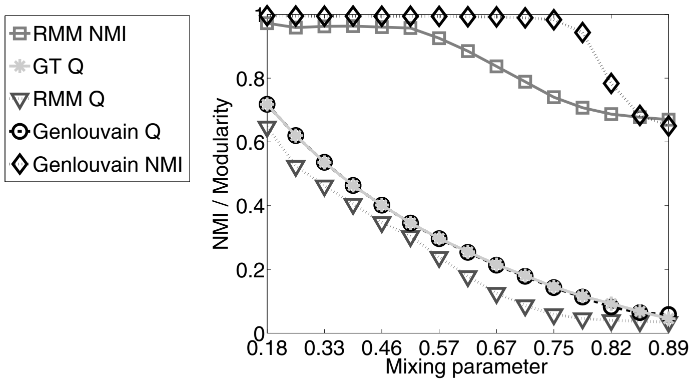
4.1.2. LFR50k networks. To examine the performance of our scheme on larger networks, we construct synthetic networks (LFR50k) with 50,000 nodes. To construct an LFR50k network, we start with 50 different LFR1k networks $N_1, N_2, \dots, N_{50}$ with mixing parameter μ, and we connect each node in N_s ($s \in \{1, 2, \dots, 50\}$) to 20μ nodes in N_{s+1} uniformly at random (where we note that $N_{51} = N_1$). We thereby obtain an LFR50k network of size $50,000$. Each LFR1k network N_s has about 40 planted communities, and each of these communities is a planted community in the LFR50k network (which thus has about 2,000 planted communities in total). We build four such LFR50k networks for each value of $\mu = 0.1, 0.15, \dots, 0.8$. The mixing parameter of the LFR50k network constructed from LFR1k(μ) is approximately $\frac{2\mu}{1+\mu}$.

By construction, the LFR50k network has a similar structure to an LFR1k network. Importantly, simply increasing N in LFR$(N, \langle k\rangle, k_{\max}, \xi, \beta, \mu, q_{\min}, q_{\max})$ to 50,000 is insufficient to preserve similarity of the network structure. A large N results in more communities, so if the mixing parameter μ is held constant, then the edges of each node that terminate at nodes outside of its community are dispersed among a larger number of communities (and the intercommunity edges are thus distributed more sparsely among the communities). In other words, the mixing parameter does not entirely reflect the balance between a node's connections within its own community and its connections to other communities, as there is also a dependence on the total number of communities.

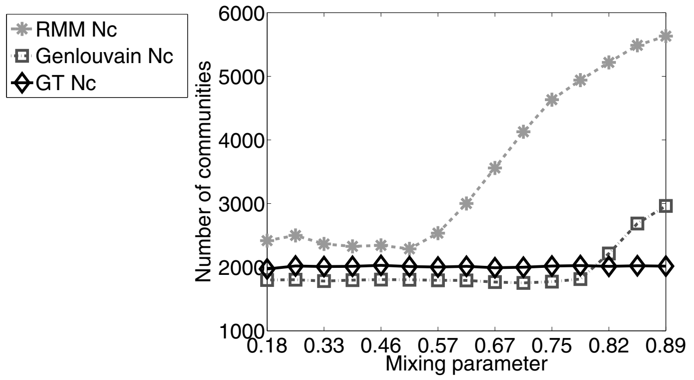
Because of the additional edges in our construction, the strength of each node in an LFR50k network is scaled approximately by a factor of $(1 + 2\mu)$ compared to an LFR1k network, and the total number of edges in LFR50k is scaled approximately by a factor of $50(1 + 2\mu)$. Therefore, the probability null model term $\frac{k_i k_j}{2m}$ in modularity (2.1) is also scaled by a factor of $\frac{(1+2\mu)}{50}$. Hence, in order to probe LFR50k with a resolution scale similar to that in LFR1k, we use the resolution $\gamma = 50$ to try to minimize issues with modularity's resolution limit [56]. We then implement the RMM scheme ($N_{\text{eig}} = 100$) and the GenLouvain code. Note that we also implemented the RMM scheme with $N_{\text{eig}} = 500$, but there is no obvious improvement in the result even though there are about 2000 communities. This is because the eigenvectors of the graph Laplacian of the subnetworks are recalculated at each recursive step, so we can resolve progressively finer-scale structures as the recursion step proceeds.

We average the network diagnostics over the four LFR50k networks for each value of the mixing parameter. In Figure 4.2, we plot the network diagnostics versus the mixing parameter $\frac{2\mu}{1+\mu}$ for $\mu \in \{0.1, 0.15, \dots, 0.8\}$. In panel (a), we see that the performance of RMM is good only when the mixing parameter is less than 0.5, though it is not as good as that of GenLouvain. It seems that the recursive modularity MBO scheme has some difficulties in dealing with networks with a very large number of clusters.

However, the computational time of RMM is lower than that of the GenLouvain



(a) NMI and modularity (Q).



(b) Number of communities (N_c).

FIG. 4.2. *Tests on LFR50k data with RMM and GenLouvain.*

code [32] (though we note that it is an implementation that was not optimized for speed). The mean computational time for an ensemble of LFR50k networks, which includes 15 networks corresponding to 15 values of μ, is 690 seconds for GenLouvain and 220 seconds for the RMM scheme. In Table 4.1, we summarize the mean computational time (in seconds) for each ensemble of LFR data.

TABLE 4.1
Computation times for the LFR1k and LFR50k networks.

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

4.2. MNIST handwritten digit images. The MNIST database consists of 70,000 images of size 28×28 pixels containing the handwritten digits "0" through "9" [38]. The digits in the images have been normalized with respect to size and centered in a fixed-size grey image. In this section, we use two networks from this database. We construct one network using all samples of the digits "4" and "9," which are difficult to distinguish from each other and which constitute 13,782 images of the 70,000. We construct the second network using all images. In each case, our goal is to separate the distinct digits into distinct communities.

We construct the adjacency matrices (and hence the graphs) $\mathbf{W}$ for these two data sets as follows. First, we project each image (a 28^2-dimensional datum) onto 50 principal components. For each pair of nodes n_i and n_j in the 50-dimensional space, we then let $w_{ij} = \exp(-\frac{d_{ij}^2}{3\sigma_i^2})$ if n_j is among the 10 nearest neighbors of n_i; otherwise, we let $w_{ij} = 0$. To construct a symmetric matrix, we take $\tilde{\mathbf{W}} = \frac{\mathbf{W}+\mathbf{W}'}{2}$, where the $'$ symbol represents matrix transposition. We now drop the tilde symbol and henceforth write $\tilde{\mathbf{W}}$ as $\mathbf{W}$. The quantity d_{ij} is the ℓ_2 distance between n_i and n_j; the parameter σ_i is the mean of the distances between n_i and its 10 nearest neighbors.

In this data set, the maximum number of communities is 2 when considering only the digits 4 and 9, and it is 10 when considering all digits. We can thus choose a small search range for $\hat{n}$ and use the multi-$\hat{n}$ modularity MBO scheme.

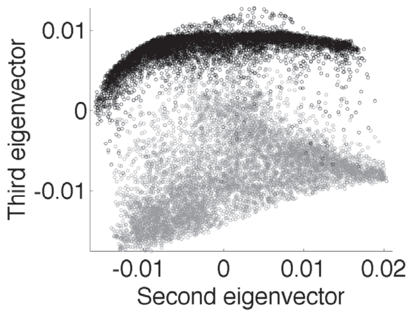
4.2.1. MNIST "4-9" digits network. This weighted network has 13,782 nodes and 194,816 weighted edges. We use the labeling of each digit image as the ground truth. There are two groups of nodes: ones containing the digit 4 and ones containing the digit 9. We use these two digits because they tend to look very similar when they are written by hand. In Figure 4.3(a), we show a visualization of this network, where we have depicted the data projected onto the second and third leading eigenvectors of the graph Laplacian $\mathbf{L}$. The difficulty of separating the 4 and 9 digits has been noted in the graph-partitioning literature (see, e.g., [31]). For example, there is a near-optimal partition of this network using traditional spectral clustering [41, 60] (see below) that splits both the 4-group and the 9-group roughly in half.

The modularity-optimization algorithms that we discuss for the 4-9 network use $\gamma = 0.1$. We choose this resolution-parameter value so that the network is partitioned into two groups by the GenLouvain code. The question about what value of γ to choose is beyond the scope of this paper, but it has been discussed at some length in the literature on modularity optimization [22]. Instead, we focus on evaluating the performance of our algorithm with the given value of the resolution parameter. We implement the modularity MBO scheme with $\hat{n} = 2$ and the multi-$\hat{n}$ MM scheme, and we compare our results with those of the GenLouvain code as well as a traditional spectral-clustering method [41, 60].

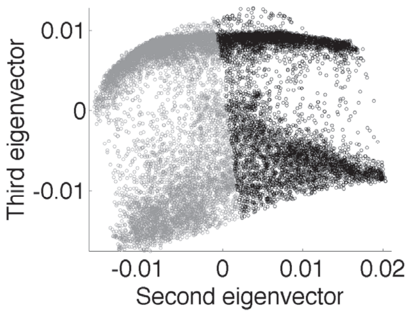
Traditional spectral clustering is an efficient clustering method that has been used widely in computer science and applied mathematics because of its simplicity. One calculates the first k nontrivial eigenvectors $\phi_1, \phi_2, \ldots, \phi_k$ (corresponding to the smallest eigenvalues) of the graph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

On this MNIST 4-9 digits network, we specify $k = 2$ and implement spectral clustering to obtain a partition into two communities. As we show in Figure 4.3(b), we obtain a near-optimal solution that splits both the 4-group and the 9-group roughly in half. This differs markedly from the ground-truth partition in panel (a).

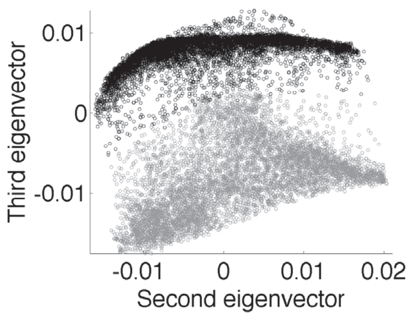
For the multi-$\hat{n}$ MM scheme, we use $N_{\text{eig}} = 80$ and the search range $\hat{n} \in \{2, 3, \ldots, 10\}$. We show visualizations of the partition for $\hat{n} = 2$ and $\hat{n} = 8$ in Figures 4.3(c,d). For this method, computing the spectrum of the graph Laplacian takes a significant portion of the run time (9 seconds for this data set). Importantly, however, this information can be reused for multiple $\hat{n}$, which saves time. In Figure 4.3(e), we



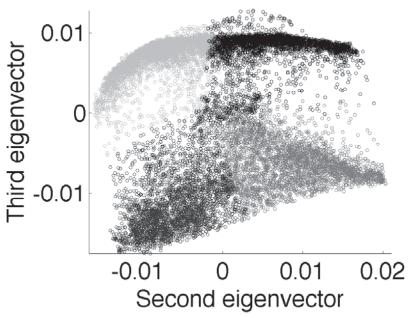
(a) Ground truth.



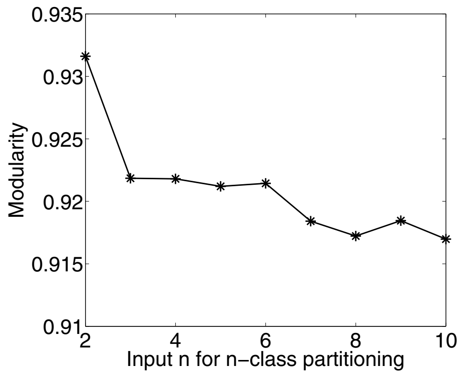
(b) Spectral clustering with k-means.



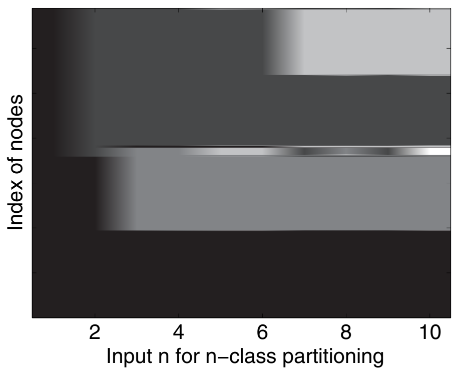
(c) Modularity MBO scheme with $\hat{n} = 2$.



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



(e) Modularity score.



(f) Group assignments visualization.

FIG. 4.3. (a)–(d) *Visualization of partitions on the MNIST* 4-9 *digit image network by projecting it onto the second and third leading eigenvectors of the graph Laplacian. Shading indicates the community assignment.* (e)–(f) *Implementation results of the multi-$\hat{n}$ modularity MBO scheme on the MNIST* 4-9 *digit images. In panel* (e)*, we plot the optimized modularity score as a function of the input $\hat{n}$. In panel* (f)*, shading indicates the community assignment. The horizontal axis represents the input $\hat{n}$ (i.e., the maximum number of communities), and the vertical axis gives the (sorted) index of nodes.*

show a plot of this method's optimized modularity scores versus $\hat{n}$. Observe that the optimized modularity score achieves its maximum when we choose $\hat{n} = 2$, which yields the best partition that we obtain using this method. In Figure 4.3(f), we show how the partition evolves as we increase the input $\hat{n}$ from 2 to 10. When $\hat{n} = 2$, the network is partitioned into two groups (which agrees very well with the ground truth). For $\hat{n} > 2$, however, the algorithm starts to select worse local optima, and either the 4-group or the 9-group gets split roughly in half. Starting from $\hat{n} = 7$, the number of communities stabilizes at about 4 instead of increasing with $\hat{n}$. This indicates that the modularity MBO scheme allows one to obtain partitions with $N_c < \hat{n}$ (in addition to ones with $N_c = \hat{n}$, of course).

In Table 4.2, we show computational time and some network diagnostics for all of the resulting partitions. The (unoptimized) modularity of the ground-truth partition is $Q_{GT} \approx 0.9277$. Our schemes yield partitions with high modularity values and NMI scores that are comparable to those obtained using the GenLouvain code (which was not intended by its authors to be optimized for speed). The number of iterations for the modularity MBO scheme ranges approximately from 15 to 35 for $\hat{n} \in \{2, 3, \ldots, 10\}$.

Table 4.2

Computation times and network diagnostics for partitions of the 4-9 *MNIST data set. (The quantity Q denotes the value of optimized modularity, so there is no such value for spectral clustering.)*

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

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

$$\operatorname{Prt}(C, \hat{C}) = \frac{1}{N} \sum_{k=1}^{K} \max_{l \in \{1, \ldots, \hat{K}\}} |C_k \cap \hat{C}_l| \in [0, 1] \,. \tag{4.2}$$

Intuitively, purity can by viewed as the fraction of nodes that have been assigned to the correct community. However, the purity score is not robust in estimating the quality of a partition. When the partition C breaks the network into communities that consist of single nodes, then the purity score achieves a value of 1. Hence, one needs to consider other diagnostics when interpreting the purity score. In this particular data set, a high purity score does indicate good performance because the ground truth and the partitions each consist of two communities.

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

4.2.2. MNIST 70k network. We test our new schemes further by consider the entire MNIST network of 70,000 samples containing digits from 0 to 9. This network contains about five times as many nodes as the MNIST 4-9 network. However, the node strengths in the two networks are very similar because of how we construct the weighted adjacency matrix. We thus choose $\gamma = 0.5$ so that the modularity optimization is performed at a similar resolution scale in both networks. There are 1,001,664 weighted edges in this network.

We implement the multi-$\hat{n}$ MM scheme with $N_{\text{eig}} = 100$ and the search range $\hat{n} \in \{2, 3, \ldots, 20\}$. Even if N_c is the number of communities in the true optimal solution, the input $\hat{n} = N_c$ might not give a partition with N_c groups. The modularity landscape in real networks is notorious for containing a huge number of nearly degenerate local optima (especially for values of modularity Q near the globally optimum value) [26], so we expect the algorithm to yield a local minimum rather than a global minimum. Consequently, it is preferable to extend the search range to $\hat{n} > N_c$, so that the larger $\hat{n}$ gives more flexibility to the algorithm in trying to find the partition that optimizes modularity.

The best partition that we obtain using the search range $\hat{n} \in \{2, 3, \ldots, 20\}$ contains 11 communities. All of the digit groups in the ground truth except for the 1-group are correctly matched to those communities. In the partition, the 1-group splits into two parts, which is not surprising given the structure of the data. In particular, the samples of the digit 1 include numerous examples that are written like a 7. These samples are thus easily disconnected from the rest of the 1-group. If one considers these two parts as one community associated with the 1-group, then the partition achieves a 96% correctness in its classification of the digits.

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

Table 4.3
Computation times and network diagnostics for partitions of the MNIST 70k data set.

	N_c	Q	NMI	Purity	Time (s)
GenLouvain	11	0.93	0.92	0.97	10900
Multi-$\hat{n}$ MM ($\hat{n} \in \{2, 3, \ldots, 20\}$)	11	0.93	0.89	0.96	290 / 212 *
Modularity MBO 3% GT ($\hat{n} = 10$)	10	0.92	0.95	0.96	94.5 / 16.5 *

* *Calculated with the RC procedure.*

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

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

Another benefit of the modularity MBO scheme is that it allows the possibility of incorporating a small portion of the ground truth into the modularity optimization process. In the present paper, we implement the modularity MBO using 3% of the ground truth by specifying the true community assignments of 2100 nodes in the initial function f^0; we chose the nodes uniformly at random. We also let $\hat{n} = 10$. With the eigenvectors already computed (which took 12 seconds using the RC process), the MBO steps take a subsequent 4.5 seconds to yield a partition with exactly 10 communities and 96.4% of the nodes classified into the correct groups. The authors of [23] also implemented a segmentation algorithm on this MNIST 70k data with 3% of the ground truth, and they obtained a partition with a correctness of 96.9% in 15.4 seconds. In their algorithm, the ground truth was enforced by adding a quadratic fidelity term to the energy functional (i.e., it is semisupervised). The fidelity term is the ℓ_2 distance between the unknown function f and the given ground truth. In our scheme, however, we use the ground truth only in the initial function f^0. Nevertheless, it is also possible to add a fidelity term to the modularity MBO scheme and thereby perform semisupervised clustering.

4.3. Network-science coauthorships. Another well-known graph type in the community-detection literature is the network of coauthorships of network scientists. This benchmark was compiled by Newman and was first used in [50].

In the present paper, we use the graph's largest connected component, which consists of 379 nodes representing authors and 914 weighted edges that indicate coauthored papers. We do not have any so-called ground truth for this network, but it is useful to compare partitions obtained from our algorithm with those obtained using more established algorithms. In this section, we use GenLouvain's result as this pseudo ground truth. In addition to modularity-MBO, RMM, and GenLouvain, we also consider the results of modularity-based spectral partitioning methods that allow the option of either bipartitioning or tripartitioning at each recursive stage [50, 57].

In [50], Newman proposed a spectral partitioning scheme for modularity optimization by using the leading eigenvectors (associated with the largest eigenvalues) of a so-called *modularity matrix* $\mathbf{B} = \mathbf{W} - \mathbf{P}$ to approximate the modularity function Q. In the modularity matrix, $\mathbf{P}$ is the probability null model and $P_{ij} = \frac{k_i k_j}{2m}$ is the NG null model with $\gamma = 1$. Assume that one uses the first p leading eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p\}$, and let β_j denote the eigenvalue of $\mathbf{u}_j$. Let $\mathbf{U} = (\mathbf{u}_1|\mathbf{u}_2|\ldots|\mathbf{u}_p)$. We then define N node vectors $\mathbf{r}_i \in \mathbb{R}^p$ whose jth component is

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

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

$$Q \simeq \hat{Q} = N\alpha + \sum_{l=1}^{\hat{n}} \|\mathbf{R}_l\|_{\ell_2}^2, \tag{4.3}$$

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

A partition that maximizes (4.3) in a given step must satisfy the geometric constraints $\mathbf{R}_l \cdot \mathbf{r}_i > 0$, $g_i = l$, and $\mathbf{R}_l \cdot \mathbf{R}_h < 0$ for all $l, h \in \{1, 2, \ldots, \hat{n}\}$. Hence, if one constructs an approximation $\hat{Q}$ using p eigenvectors, a network component can be split into at most $p + 1$ groups in a given recursive step. The choice $p = 2$ allows either bipartitioning or tripartitioning in each recursive step. Reference [50] discussed the case of general p but reported results for recursive bipartitioning with $p = 1$. Reference [57] implemented this spectral method with $p = 2$ and a choice of bipartitioning or tripartitioning at each recursive step.

In Table 4.4, we report diagnostics for partitions obtained by several algorithms (for $\gamma = 1$). For the recursive spectral bipartitioning and tripartitioning, we use Matlab code that has been provided by the authors of [57]. They informed us that this particular implementation was not optimized for speed, so we expect it to be slow; one can create much faster implementations of the same spectral method. The utility of this method for the present comparison is that [57] includes a detailed discussion of its application to the network of network scientists. Each partitioning step in this spectral scheme either bipartitions or tripartitions a group of nodes. Moreover, as discussed in [57], a single step of the spectral tripartitioning is by itself interesting. Hence, we specify $\hat{n} = 3$ for the modularity MBO scheme as a comparison.

Table 4.4
Computation times and network diagnostics for partitions of the network of network scientists.

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

From Table 4.4, we see that the modularity MBO scheme with $\hat{n} = 3$ gives a higher modularity than a single tripartition from the algorithm in [57], and the former's NMI and purity are both significantly higher. When we do not specify the number of clusters, the RMM scheme achieves a higher modularity score and NMI than recursive bipartitioning/tripartitioning, though the former's purity is lower (which is not surprising due to its larger N_c). The RMM scheme and GenLouvain have similar run times. For any of these methods, one can of course use subsequent postprocessing, such as Kernighan–Lin node-swapping steps [50,54,57], to find higher-modularity partitions.

4.4. A note on computational heuristics and time complexity. Numerous computational heuristics have been employed to optimize network modularity [22,54]. We have compared our results with implementations of a small number of popular methods that others have made available. We report computation times in our discussions, but we note that these available implementations (e.g., the GenLouvain code from [32]) were not optimized for speed. Our results above demonstrate the good speed of our method, but we have not, for example, included a comparison of its speed with the C++ implementations of Blondel et al. [27] of the Louvain method [5]. Importantly, the low computational cost of our method is a direct result of its firm theoretical grounding, and our reformulation of the problem of modularity optimization offers hope for the development of even faster methods in the future. We performed all calculations in this paper using Matlab R2011b on a MacBook Air with a 1.7 GHz processor (Intel Core i5).

5. Conclusion and discussion.

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

One major part of our schemes is to calculate the leading eigenvector-eigenvalue pairs, so one can benefit from the fast numerical Rayleigh–Chebyshev procedure in [1] when dealing with large, sparse networks. Furthermore, for a given network (which is represented by a weighted adjacency matrix), one can reuse previously computed eigendecompositions for different choices of initial functions, different values of $\hat{n}$, and different values of the resolution parameter γ. This provides welcome flexibility, and it can be used to significantly reduce computation time because the MBO step is extremely fast, as each step is $O(N)$ and the number of iterations is empirically small.

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

Appendix A. The notion of Γ-convergence of functionals is now commonly used for minimization problems; see [42] for a detailed introduction. In this appendix, we briefly review the definition of Γ-convergence and then prove the claim that the graphical multiphase Ginzburg–Landau functional Γ-converges to the graph TV. This proof is a straightforward extension of the work in [24] for the two-phase graph GL functional.

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

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

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

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

Reference [23] proposed the following multiphase graph GL functional:

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

where $\hat{f} : G \to \mathbb{R}^{\hat{n}}$ and $W_{\text{multi}}(\hat{f}(n_i)) = \prod_{l=1}^{\hat{n}} \|\hat{f}(n_i) - \vec{e}_l\|_{\ell_1}^2$. See sections 2 and 3 for the definitions of all of the relevant graph notation. Let $X = \{\hat{f} \mid \hat{f} : G \to \mathbb{R}^{\hat{n}}\}$, $X^p = \{f \mid f : G \to V^{\hat{n}}\} \subset X$, and $F_\epsilon = GL_\epsilon^{\text{multi}}$ for all $\epsilon > 0$. Because $\hat{f}$ can be viewed as a matrix in $\mathbb{R}^{N\times\hat{n}}$, the metric for the space X can be defined naturally using the ℓ_2-norm.

THEOREM A.2 (Γ-convergence). *The sequence F_ϵ Γ-converges to F_0 as $\epsilon \to 0^+$, where*

$$F_0(\hat{f}) := \begin{cases} |\hat{f}|_{TV} = \frac{1}{2}\sum_{i,j=1}^N w_{ij}\|\hat{f}(n_i) - \hat{f}(n_j)\|_{\ell_1} & \text{if } \hat{f} \in X^p, \\ +\infty & \text{otherwise.} \end{cases}$$

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

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

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

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

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

Acknowledgments. We thank Marya Bazzi, Yves van Gennip, Blake Hunter, Ekaterina Merkurjev, and Peter Mucha for useful discussions. We also thank Peter Mucha for providing his spectral partitioning code. We have included acknowledgements for data directly in the text and the associated references.

REFERENCES

[1] 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.

[2] G. BARLES AND C. GEORGELIN, *A simple proof of convergence for an approximation scheme for computing motions by mean curvature*, SIAM J. Numer. Anal., 32 (1995), pp. 484–500.

[3] M. BELKIN AND P. NIYOGI, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Comput., 15 (2003), pp. 1373–1396.

[4] 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.

[5] V. D. BLONDEL, J.-L. GUILLAUME, R. LAMBIOTTE, AND E. LEFEBVRE, *Fast unfolding of communities in large networks*, J. Statist. Mech., 10 (2008), P10008.

[6] S. BOETTCHER AND A. G. PERCUS, *Optimization with extremal dynamics*, Complexity, 8 (2002), pp. 57–62.

[7] U. BRANDES, D. DELLING, M. GAERTLER, R. GÖRKE, M. HOEFER, Z. NIKOLOSKI, AND D. WAGNER, *On modularity clustering*, IEEE Trans. Knowledge Data Engrg., 20 (2008), pp. 172–188.

[8] X. BRESSON, T. LAURENT, D. UMINSKY, AND J. VON BRECH, *Convergence and energy landscape for Cheeger cut clustering*, Adv. Neural Inform. Process. Syst., 25 (2012), pp. 1394–1402.

[9] X. BRESSON, X.-C. TAI, T. F. CHAN, AND A. SZLAM, *Multi-class transductive learning based on ℓ^1 relaxations of Cheeger cut and Mumford-Shah-Potts model*, J. Math. Imaging Vision, (2013), to appear; available online at http://link.springer.com/article/10.1007/s10851-013-0452-5.

[10] T. F. Chan and L. A. Vese, *Active contours without edges*, IEEE Trans. Image Process., 10 (2001), pp. 266–277.

[11] F. R. K. Chung, *Spectral Graph Theory*, CBMS Reg. Conf. Ser. Math. 92, AMS, Providence, RI, 1997.

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

[13] R. R. Coifman and S. Lafon, *Diffusion maps*, Appl. Comput. Harm. Anal., 21 (2006), pp. 5–30.

[14] M. Cucuringu, V. D. Blondel, and P. V. Dooren, *Extracting spatial information from networks with low-order eigenvectors*, Phys. Rev. E, 87 (2013), 032803.

[15] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, *Comparing community structure identification*, J. Statist. Mech., 9 (2005), P09008.

[16] P. Doreian, V. Batagelj, and A. Ferligoj, *Generalized Blockmodeling*, Cambridge University Press, Cambridge, UK, 2004.

[17] J. Duch and A. Arenas, *Community detection in complex networks using extremal optimization*, Phys. Rev. E, 72 (2005), 027104.

[18] S. Esedoglu and F. Otto, *Threshold Dynamics for Networks with Arbitrary Surface Tensions*, 2013, submitted; available online at http://www.mis.mpg.de/publications/preprints/2013/prepr2013-2.html.

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

[20] L. C. Evans, *Convergence of an algorithm for mean curvature motion*, Indiana Univ. Math. J., 42 (1993), pp. 533–557.

[21] D. Eyre, *An Unconditionally Stable One-Step Scheme for Gradient Systems*, unpublished paper, University of Utah, 1998, available online at http://www.math.utah.edu/∼eyre/research/methods/stable.ps.

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

[23] C. Garcia-Cardona, E. Merkurjev, A. L. Bertozzi, A. Flenner, and A. Percus, *Fast multiclass segmentation using diffuse interface methods on graphs*, arXiv:1302.3913, 2013.

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

[25] M. Girvan and M. E. J. Newman, *Community structure in social and biological networks*, Proc. Natl. Acad. Sci. USA, 99 (2002), pp. 7821–7826.

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

[27] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, *Louvain Method: Finding Communities in Large Networks*, available online at https://sites.google.com/site/findcommunities/ (2008–2011).

[28] R. Guimerà and L. A. N. Amaral, *Functional cartography of complex metabolic networks*, Nature, 433 (2005), pp. 895–900.

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

[30] M. Hein and T. Bühler, *An inverse power method for nonlinear eigenproblems with applications in* 1*-spectral clustering and sparse PCA*, in Advances in Neural Information Processing Systems (NIPS) 23, 2010, pp. 847–855.

[31] M. Hein and S. Setzer, *Beyond spectral clustering—Tight relaxations of balanced graph cuts*, Adv. Neural Inform. Process. Syst., 24 (2011), pp. 2366–2374.

[32] I. S. Jutla, L. G. S. Jeub, and P. J. Mucha, *A Generalized Louvain Method for Community Detection Implemented in MATLAB*, available online at http://netwiki.amath.unc.edu/GenLouvain (2011–2012), version 1.2.

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

[34] R. V. Kohn and P. Sternberg, *Local minimizers and singular perturbations*, Proc. Roy. Soc. Edinburgh Sec. A Math., 111 (1989), pp. 69–84.

[35] R. Lambiotte, J.-C. Delvenne, and M. Barahona, *Laplacian dynamics and multiscale modular structure in networks*, arXiv:0812.1770 (2009).

[36] A. Lancichinetti, S. Fortunato, and F. Radicchi, *Benchmark graphs for testing community detection algorithms*, Phys. Rev. E, 78 (2008), 046110.

[37] A. Lancichinetti and S. Fortunato, *Community detection algorithms: A comparative analysis*, Phys. Rev. E, 80 (2009), 056117.

[38] Y. LeCun, C. Cortes, and C. J. C. Burges, *MNIST Database*, online at http://yann.lecun.com/exdb/mnist/.

[39] A. C. F. Lewis, N. S. Jones, M. A. Porter, and C. M. Deane, *The function of communities in protein interaction networks at multiple scales*, BMC Syst. Biol., 4 (2010), 100.
[40] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, *Compressed sensing MRI*, IEEE Signal Process. Mag., 25 (2008), pp. 72–82.
[41] U. von Luxburg, *A tutorial on spectral clustering*, Statist. Comput., 17 (2007), pp. 395–416.
[42] G. D. Maso, *An Introduction to* Γ*-Convergence*, Progr. Nonlinear Differential Equations Appl. 8, Springer, New York, 1993.
[43] E. Merkurjev, T. Kostić, and A. L. Bertozzi, *An MBO scheme on graphs for classification and image processing*, SIAM J. Imaging Sci., 6 (2013), pp. 1903–1930.
[44] B. Merriman, J. K. Bence, and S. J. Osher, *Motion of multiple junctions: A level set approach*, J. Comput. Phys., 112 (1994), pp. 334–363.
[45] D. Mumford and J. Shah, *Optimal approximations by piecewise smooth functions and associated variational problems*, Comm. Pure Appl. Math., 42 (1989), pp. 577–685.
[46] D. Needell and R. Ward, *Stable image reconstruction using total variation minimization*, SIAM J. Imaging Sci., 6 (2013), pp. 1035–1058.
[47] M. E. J. Newman and M. Girvan, *Mixing patterns and community structure in networks*, in Statistical Mechanics of Complex Networks, R. Pastor-Satorras, J. Rubi, and A. Diaz-Guilera, eds., Springer, Berlin, 2003, pp. 66–87.
[48] M. E. J. Newman and M. Girvan, *Finding and evaluating community structure in networks*, Phys. Rev. E, 69 (2004), 026113.
[49] M. E. J. Newman, *Fast algorithm for detecting community structure in networks*, Phys. Rev. E, 69 (2004), 066133.
[50] M. E. J. Newman, *Finding community structure in networks using the eigenvectors of matrices*, Phys. Rev. E, 74 (2006), 036104.
[51] M. E. J. Newman, *The physics of networks*, Physics Today, 61 (2008), pp. 33–38.
[52] M. E. J. Newman, *Networks: An Introduction*, Oxford University Press, London, 2010.
[53] A. Y. Ng, M. I. Jordan, and Y. Weiss, *On spectral clustering: Analysis and an algorithm*, Adv. Neural Inform. Process. Syst., 14 (2001), pp. 849–856.
[54] M. A. Porter, J.-P. Onnela, and P. J. Mucha, *Communities in networks*, Notices Amer. Math. Soc., 56 (2009), pp. 1082–1097, 1164–1166.
[55] S. Rangapuram and M. Hein, *Constrained* 1*-spectral clustering*, in Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), 2012, pp. 1143–1151.
[56] J. Reichardt and S. Bornholdt, *Statistical mechanics of community detection*, Phys. Rev. E, 74 (2006), 016110.
[57] T. Richardson, P. J. Mucha, and M. A. Porter, *Spectral tripartitioning of networks*, Phys. Rev. E, 80 (2009), 036111.
[58] M. P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha, *Core-periphery structure in networks*, SIAM J. Appl. Math., to appear; arXiv:1202.2684 (2013).
[59] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation noise removal algorithm*, Phys. D, 60 (1992), pp. 259–268.
[60] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Trans. Pattern Anal. Machine Intell., 22 (2000), pp. 888–905.
[61] A. Szlam and X. Bresson, *A total variation-based graph clustering algorithm for Cheeger ratio cuts*, in Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 1039–1046.
[62] A. L. Traud, P. J. Mucha, and M. A. Porter, *Social structure of Facebook networks*, Phys. A, 391 (2012), pp. 4165–4180.
[63] B. P. Vollmayr-Lee and A. D. Rutenberg, *Fast and accurate coarsening simulation with an unconditionally stable time step*, Phys. Rev. E, 68 (2003), 066703.
[64] Y. Zhang, A. J. Friend, A. .L. Traud, M. A. Porter, J. H. Fowler, and P. J. Mucha, *Community structure in Congressional cosponsorship networks*, Phys. A, 387 (2008), pp. 1705–1712.