CONTRIBUTED PAPER

An Optimization Approach to Locally-Biased Graph Algorithms

This paper investigates a class of locally-biased graph algorithms for finding local or small-scale structures in large graphs.

By Kimon Fountoulakis, David F. Gleich, and Michael W. Mahoney

ABSTRACT | Locally-biased graph algorithms are algorithms that attempt to find local or small-scale structure in a large data graph. In some cases, this can be accomplished by adding some sort of locality constraint and calling a traditional graph algorithm; but more interesting are locally-biased graph algorithms that compute answers by running a procedure that does not even look at most of the input graph. This corresponds more closely to what practitioners from various data science domains do, but it does not correspond well with the way that algorithmic and statistical theory is typically formulated. Recent work from several research communities has focused on developing locally-biased graph algorithms that come with strong complementary algorithmic and statistical theory and that are useful in practice in downstream data science applications. We provide a review and overview of this work, highlighting commonalities between seemingly different approaches, and highlighting promising directions for future work.

KEYWORDS | Clustering; graph algorithms; local algorithms; network flow; partitioning; semi-supervised learning; spectral graph theory

I. INTRODUCTION

Graphs, long popular in computer science and discrete mathematics, have received renewed interest recently in statistics, machine learning, data analysis, and related areas because they provide a useful way to model many types of relational data. In this way, graphs can be used to extract insight from data arising in many application domains. In biology, e.g., graphs are routinely used to generate hypotheses for experimental validation [1]; and in neuroscience, they are used to study the networks and circuits in the brain [2], [3].

Given their ubiquity, graphs and graph-based data have been approached from several different perspectives. In computer science, it is common to develop algorithms, e.g., for connected component, minimum spanning tree, and maximum flow problems, to run on data that are modeled as a precisely specified input graph. These algorithms are then characterized by the number of operations they use for the worst possible input at any size. In statistics and machine learning, on the other hand, it is common to use graphs as models to perform inference about unseen data. In this case, one often hypothesizes an unseen graph with a particular structure, such as block structure, hierarchical structure, low-rank structure in the adjacency matrix, etc. Then, one runs algorithms on the observed data in order to impute entries in the unobserved hypothesized graph. These methods may be characterized in terms of running time, but they are also characterized in terms of the amount of data needed to recover the hidden hypothesized structure.

In many application areas where the end goal is to obtain some sort of domain-specific insight, e.g., such as in social network analysis, neuroscience, medical imaging, etc., one constructs graphs from primary data, and then one runs a computational procedure that does *not* come with either of these traditional types of theoretical guarantees. As an example, consider the GeneRank method [4], where we have a set of genes related to an experimental condition in a microarray study. This set of genes is "refined" via a locally-biased graph algorithm closely related to those we will discuss. Importantly, this operational refinement procedure does *not* come with the sort of theory traditional in statistics, machine learning, or computer science. As another example, e.g., in social network applications, one might run a random walk process for a few steps from a starting node of interest, and if the walk "gets stuck" then one might interpret this as evidence that that region of the graph is meaningful in the application domain [5]. These are examples of the types of

Manuscript received February 29, 2016; revised October 19, 2016; accepted November 29, 2016. Date of current version January 18, 2017.
K. Fountoulakis and **M. W. Mahoney** are with the International Computer Science Institute and the Department of Statistics, University of California Berkeley, Berkeley, CA, USA.
D. F. Gleich is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907 USA (e-mail: dgleich@purdue.edu).

Digital Object Identifier: 10.1109/JPROC.2016.2637349

heuristics commonly used in applications. By heuristic, we mean an algorithm in the sense that it performs a sequence of well-defined steps, but one where precise theory is lacking (although usually heuristics come with strong intuitive motivation and are justified in terms of some downstream application). In particular, typically heuristics do not explicitly optimize a well-defined objective function and typically they do not come with well-defined inferential guarantees.

Note that, in both of these examples, the goal is to find "local" or "small-scale" structure in the data graph. Both examples also correspond to what practitioners interested in downstream applications actually do. Existing algorithmic and statistical theory, however, has challenges with these local or small-scale structures. For instance, a very "good" algorithmic runtime on a graph is traditionally one that is linear in the number of vertices and edges. If the output of interest is only a vanishingly small fraction of a large graph, however, then this theory may not provide strong qualitative guidance on how these locally-biased methods behave in practice. Likewise, inferential methods often assume that the structures inferred constitute a substantial fraction of the graph, and many statistical techniques have challenges differentiating very small structure from random noise.

In this overview, we describe a class of graph algorithms that has proven to be very useful for identifying and interpreting small-scale local structure in large-scale data. For this class of algorithms, however, strong algorithmic and statistical theory has been developed. In particular, these graph algorithms are locally biased in one of several precisely quantified senses. We will describe what we mean by this in more detail below, but, informally, this means that the algorithms are most interested in only a small part of a large graph. As opposed to heuristic operational procedures, however, many of these algorithms do come with strong worst case algorithmic guarantees, and many of these algorithms also do come with statistical guarantees that prove they have implicit regularization properties. This complementary algorithmic-statistical theory helps explain their improved performance in many practical applications.

While the approach of locally-biased graph algorithms is very general, it has been developed most extensively for the fundamental problem of finding locally-biased graph partitions, i.e., clusters or communities, and so we will focus on locally-biased approaches for the graph clustering problem. Of course, this partitioning question is of interest in many more application-driven areas, where one is interested in finding useful or meaningful clusters as part of a data analysis pipeline.

A. The Rationale for Local Analysis in Real-World Data

As a quick example of why local graph analysis is frequently used in data and data science applications, we present in Figure 1 the results of finding the best partition of both a random geometric graph and a more typical data graph. Standard graph partitioning algorithms must operate on, or "touch", each vertex and edge of the graph to identify

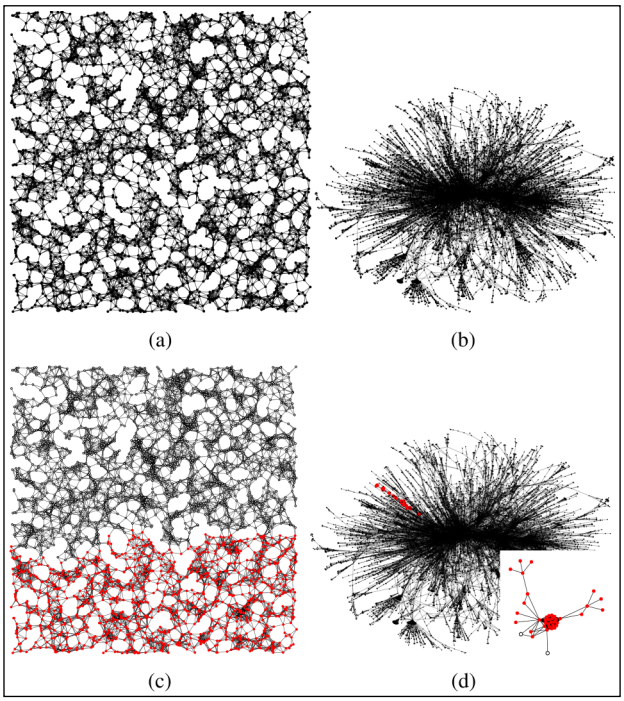

Fig. 1. ***At left, a geometric graph has a pleasing and intuitive layout in the 2-D plane. At right, a more typical data graph has a classic hairball layout that shows little high level structure. Due to the evident lack of global structure in the data graph, locally-biased graph algorithms are often used in these contexts. The solution of the minimum-conductance problem in the geometric graph is a large set of nodes, and it has conductance value 0.00464. The solution of the minimum-conductance problem in the more typical data graph is a small set of nodes, and it has conductance value 0.00589. The inset figure shows that this small graph is very dense and has only three edges leaving the set. (a) A random geometric graph. (b) A typical data graph. (c) The optimal conductance solution for the geometric graph bisects the graph into two large well-balanced pieces. (d) The optimal conductance solution for a typical data graph. (Inset. A zoomed view of the subgraph where the two unfilled notes are the border with the rest of the graph.)***

these partitions. The best partition of the geometric graph is around half the data, where it is reasonable to run an algorithm that touches all the data. On the other hand, the best partition of the data graph is very small, and in this case touching the entire graph to find it can be too costly in terms of computation time. The local graph clustering techniques discussed in this paper can find this cluster when given a small set of nodes inside and touching only edges and nodes that scale in the size of the output cluster, greatly reducing the computation time.

Far from being a pathology or a peculiarity, the finding that optimal partitions of real-world networks are often extremely imbalanced, thus leading to very small optimal clusters, is endemic to many of the graphs arising in large-scale data analysis [6]–[9].[1]

[1]An important applied question has to do with the meaningfulness, usefulness, etc., of such small clusters. We do not consider those questions here, and instead we refer the interested reader to prior work [6]–[9]. Here, we instead focus on the algorithmic and statistical properties of these locally-biased algorithms.

Let us now explain in more detail Fig. 1. Fig. 1(a) shows a graph with 3000 nodes that is typical of many graphs that possess a strong underlying geometry, e.g., those used in computer graphics, computer vision, logistics planning, road network analysis, and so on. This particular graph is produced by generating 3000 random points in the plane and connecting all points within a small radius, such that the final graph is connected. The geometric graph can be nearly bisected by optimizing a measure known as conductance (we will define this shortly), which is designed to balance partition size and quality. In Fig. 1(b), we show a more typical data graph of around 10 680 nodes [10], where this particular data graph is based on the trust relationships in a pretty good privacy (PGP) key chain. Optimizing the same conductance objective function results in a tiny set [Fig. 1(d)] and not the near-bisection as in the geometric case [Fig. 1(c)]. (We were able to use integer optimization techniques to directly solve the NP-hard problems at the cost of months of computation.) Many other examples of this general phenomenon can be found in prior work [6]–[9].

B. Partitioning as a Model Problem

The problem of finding good partitions or clusters is ubiquitous. Representative examples include biomedical applications [11], [12], internet and world wide web [13]–[15], social graphs [6], [16]–[18], human communication graphs [19], human mobility graphs [20], voting graphs in political science [21]–[25], protein interaction graphs [26], material science [27], [28], neuroscience [29]–[31], and collaboration graphs [32].

All of the features of locally-biased computations are present in this model partitioning problem. For example, while some of these algorithms read the entire graph as input but are engineered to return answers that are biased toward and meaningful for a smaller part of the input graph [33]–[36], other algorithms can take as input a "small seed" set of nodes as well as an oracle with which to access neighbors of a node, and they return meaningful answers without even touching the entire graph [37]–[40]. Similarly, while these algorithms are often formulated in the language of theoretical computer science as approximation algorithms, i.e., they come with running time guarantees and can be shown to approximate to within some quality-of-approximation factor some objective function of interest, e.g., conductance, in other cases one can prove statistical results such as that they exactly solve a regularized version of that objective [37], [41]–[43].

Importantly, this statistical regularization is implicit rather than explicit. Typically, regularization is explicitly used in statistics, when fitting models with a large number of parameters, in order to avoid overfitting to the given data. It is also used to select answers biased towards specific sets—for example, sparse solution sets by using the Lasso [44]. In the case of locally-biased graph algorithms, one simply runs a faster algorithm for a problem. In some cases, the nature of the approximation that this fast algorithm makes can be related back to a regularized variant of the objective function.

C. Overview

In this paper, we will consider partitioning from the perspective of conductance, and we will survey a recent class of results about a common localizing construction. These methods will let us find the optimal sets in Fig. 1 without resorting to integer optimization (but also losing the proof of optimality). Depending on the exact construction, they will also come with a variety of helpful statistical properties. We will conclude with a variety of different perspectives on these problems and some open questions.

In Section II, we describe assumptions, notation, and preliminary results which we will use in this paper. In Section III, we discuss two global graph partitioning problems and their spectral relaxation. In Section IV, we describe the local graph clustering application. In Section V, we provide empirical evaluations for the global and local graph clustering algorithms which are described in this paper. Finally, in Section VI, we give our conclusions.

II. PRELIMINARIES AND NOTATION

Graph assumptions: We use the letter $\mathcal{G}$ to denote a given connected graph. We assume that $\mathcal{G}$ is undirected with no self-loops. Many of the constructions we will use operate on weighted graphs and so we assume that each edge may have a positive capacity. Graphs that are unweighted should have all of their capacities set to 1.

Nodes, edges, and cuts: Let $\mathcal{V} = \{v_1, v_2 \ldots, v_n\}$ be a given set of $|\mathcal{V}|$ nodes of graph $\mathcal{G}$. We denote with e_{ij} an edge in the graph between nodes v_i and v_j. Let $\mathcal{E}$ be a given set of $|\mathcal{E}|$ edges of graph $\mathcal{G}$. A subset $S \subset \mathcal{V}$ of nodes can be used to define a partitioning of $\mathcal{V}$ into S and $S^c := \mathcal{V} \backslash S$. We define a cut as subset $E \subset \mathcal{E}$ which partitions the graph $\mathcal{G}$ into two sets. Given a partition $S \subset \mathcal{V}$ and S^c, then $E(S, S^c) = \{e_{ij} \in \mathcal{E} \mid v_i \in S$ and $v_j \in S^c\}$ is the set of edges with one side in S and the other side in S^c. If the partition is clear from the context we write the cut set as E instead of $E(S, S^c)$. Let c_{ij} be a weight of the edge e_{ij}, then we define the cut S as

$$\mathrm{cut}(S) := \mathrm{cut}(E(S, S^c)) := \sum_{e_{ij} \in E(S,S^c)} c_{ij}. \tag{1}$$

The volume of a set S is

$$\mathrm{vol}(S) := \sum_{v_i \in S} \sum_{e_{ij} \in \mathcal{E}} c_{ij}. \tag{2}$$

Matrix notation: We denote with $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ the adjacency matrix for a given graph, where $A_{ij} = c_{ij}\ \forall e_{ij} \in \mathcal{E}$ and zero elsewhere. Let d_i be the degree of node $v_i \in \mathcal{V}$, $D \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ be the degree diagonal matrix $D_{ii} = d_i$, $L = D - A$ be the graph Laplacian, and $\mathcal{L} = D^{-1/2} L D^{-1/2}$ be the symmetric normalized graph Laplacian. Note that the volume of a subset S is $\mathrm{vol}(S) = \sum_{v_i \in S} d_i$. We denote with $B \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{V}|}$ the incidence matrix of the given graph $\mathcal{G}$. Every row of the incidence matrix corresponds to an edge $e_{ij} \in \mathcal{E}$ in $\mathcal{G}$. Assuming arbitrary ordering of the edges of the graph, in this paper we define the rows of the incidence matrix as $B_{e_{ij}} = e_i - e_j\ \forall e_{ij} \in \mathcal{E}$,

where $e_i \in \mathbb{R}^{|\mathcal{V}|}$ is equal to one at the ith position and zero elsewhere. Finally, $C \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ is a diagonal matrix of the weights of each edge, i.e., $C_{ij} = c_{ij}$ $\forall i,j$. In this notation, the Laplacian matrix $L = B^T CB$.

Norms: For all $x \in \mathbb{R}^{|\mathcal{E}|}$, we define the weighted ℓ_1 and ℓ_2 norms $||x||_{1,C} := \sum_{e_{ij}\in E} c_{ij}|x_{ij}|$ and $||x||^2_{2,C} := \sum_{e_{ij}\in E} c_{ij}|x_{ij}|^2$, respectively. The order of elements of the vector x is identical to the order of edges in the incidence matrix. Given a partition S, S^c and a vector $x \in \{0,1\}^{|\mathcal{V}|}$ such that $x_i = 1$ if $v_i \in S$ and $x_i = 0$ if $v_i \in S^c$, then $\text{cut}(S) = ||Bx||_{1,C}$. Moreover, notice that $\text{cut}(S) = ||Bx||^2_{2,C} = x^T B^T CBx = x^T Lx$.

Miscellaneous: We use $[x;y;z]$ to denote a single column vector where the individual vectors x, y, z are stacked in this order. Finally, the vectors $0_{|\mathcal{V}|}$ and $1_{|\mathcal{V}|}$ are the all zeros and ones vectors of length $|\mathcal{V}|$, respectively.

III. SPARSEST CUT, MINIMUM CONDUCTANCE, AND SPECTRAL RELAXATIONS

In this section, we present two ubiquitous combinatorial optimization problems: sparsest cut and minimum conductance. These problems are NP-hard [45], [46], but they can be relaxed to tractable convex optimization problems [47]. We discuss one of the commonly used relaxation techniques which will motivate part of our discussion for local graph clustering algorithms. Both sparsest cut and minimum conductance give different ways of balancing the size of a partition with its quality.

Sparsest cut finds the partition that minimizes the ratio of the fraction of edges that are removed divided by the scaled product of volumes of the two disjoint sets of nodes defined by removing those edges. In particular, if we have a partition (S, S^c), where S^c is defined in Section II, then $\text{cut}(S)$ is the number of edges that are removed, and the scaled product of volumes $\text{vol}(S)\,\text{vol}(S^c)/\text{vol}(\mathcal{V})$ is the volume of the disjoint sets of nodes. Putting all together in an optimization problem, we obtain

$$\tilde{\varphi}(\mathcal{G}) := \quad \text{minimize } \tilde{\varphi}(S) := \frac{\text{cut}(S)}{\frac{1}{\text{vol}(\mathcal{V})}\text{vol}(S)\,\text{vol}(S^c)} \qquad (3)$$
$$\text{subject to } S \subset \mathcal{V}.$$

We will use the term "expansion of a set" to refer to the ratio $\tilde{\varphi}(S)$, and the term "expansion of the graph" to refer to $\tilde{\varphi}(\mathcal{G})$, in which case this problem is known as the sparsest cut problem.

Another way of balancing the partition is

$$\varphi(\mathcal{G}) := \quad \text{minimize } \varphi(S) := \frac{\text{cut}(S)}{\min(\text{vol}(S), \text{vol}(S^c))} \qquad (4)$$
$$\text{subject to } S \subset \mathcal{V}.$$

In this case, we divide by the minimum of $\text{vol}(S)$ and $\text{vol}(S^c)$, rather than their product. We will use the term "conductance of a set" to refer to the ratio of cut to volume $\varphi(S)$, and the term "conductance of the graph" to refer to $\varphi(\mathcal{G})$, in which case this problem is known as the minimum conductance problem.

The difference between the two problems (3) and (4) is that the former regularizes based on the number of connections lost among pairs of nodes, while the latter regularizes based on the size of the small side of the partition. Optimal solutions to these problems differ by a factor of 2

$$\frac{1}{2}\tilde{\varphi}(S) \le \varphi(S) \le \tilde{\varphi}(S) \qquad (5)$$

leading to the two objectives $\tilde{\varphi}(S)$ and $\varphi(S)$ being almost substitutable from a theoretical computer science perspective [47]. However, this does not mean that the actual obtained solutions by solving (3) and (4) are similar; in general, they are not.

There are three major relaxation techniques for the NP-hard problems (3) and (4): spectral relaxation, all pairs multicommodity flow or linear programming (LP) relaxation, and semidefinite programming (SDP) relaxation. For detailed descriptions about the LP and SDP relaxations we refer the reader to [46] and [47], respectively. We focus here on spectral relaxation since similar relaxations are widely used for the development of local clustering methods, which we discuss in subsequent sections.

A. Spectral Relaxation

Spectral graph partitioning is one of the best known relaxations of the sparsest cut (3) and minimum conductance (4). The relaxation is the same for both problems, although the diversity of derivations of spectral partitioning does not always make this connection clear. For a partition (S, S^c), let us associate a vector $x \in \{c_1, c_2\}^{|\mathcal{V}|}$ such that $x_i = c_1$ if $v_i \in S$ and $x_i = c_2$ if $v_i \in S^c$. (For simplicity, think of $c_1 = 1$ and $c_2 = 0$, so x is the set indicator vector.) The spectral clustering relaxation uses a continuous relaxation of the set indicator vector in problems (3) and (4) to produce an eigenvector. The relaxed problem is

$$\lambda_2 := \quad \text{minimize } \frac{||Bx||^2_{2,C}}{2||x||^2_{2,D}}$$
$$\text{subject to } 1^T_{|\mathcal{V}|} Dx = 0 \qquad (6)$$
$$x \in \mathbb{R}^{|\mathcal{V}|} - \{0_{|\mathcal{V}|}\}.$$

(The denominator $||x||^2_{2,D} = \sum |x_i|^2 d_i$.) To see why (6) is a continuous relaxation of (3) we make two observations.

First, notice that for all x in $\mathbb{R}^{|\mathcal{V}|} - \{0_{|\mathcal{V}|}\}$ such that $1^T_{|\mathcal{V}|} Dx = 0$ the denominator in (6) satisfies $2\text{vol}(\mathcal{V})||x||^2_{2,D} = \sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} d_i d_j |x_i - x_j|^2$. Therefore, λ_2 in (6) is equivalent to

$$\lambda_2 = \begin{array}{ll} \text{minimize} & \dfrac{||Bx||_{2,C}^2}{\frac{1}{\text{vol}(\mathcal{V})}\sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} d_i d_j |x_i - x_j|^2} \\ \text{subject to} & 1_{|\mathcal{V}|}^T Dx = 0 \\ & x \in \mathbb{R}^{|\mathcal{V}|} - \{0_{|\mathcal{V}|}\}. \end{array} \tag{7}$$

The optimal value of the right-hand side in (7) is equivalent to the optimal value of right-hand side in the following expression:

$$\lambda_2 = \begin{array}{ll} \text{minimize} & \dfrac{||Bx||_{2,C}^2}{\frac{1}{\text{vol}(\mathcal{V})}\sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} d_i d_j |x_i - x_j|^2} \\ \text{subject to} & x \in \mathbb{R}^{|\mathcal{V}|} - \{0_{|\mathcal{V}|}, 1_{|\mathcal{V}|}\}. \end{array} \tag{8}$$

To prove this notice that the objective function of the right-hand side in (8) is invariant to constant shifts of x, i.e., x and $x + c1_{|\mathcal{V}|}$ have the same objective function, where c is a constant. Therefore, if $\tilde{x}$ is an optimal solution of the right-hand side in (8), then $\hat{x} = \tilde{x} - \frac{1_{|\mathcal{V}|}^T D\tilde{x}}{vol(\mathcal{V})} 1_{|\mathcal{V}|}$ has the same optimal objective value and also $1_{|\mathcal{V}|}^T D\hat{x} = 0$.

Second, by restricting the solution in (8) in $\{0,1\}^{|\mathcal{V}|}$ instead of $\mathbb{R}^{|\mathcal{V}|}$ we get that $\text{cut}(S) = ||Bx||_{2,C}^2$ and

$$\sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} d_i d_j |x_i - x_j|^2 = \text{vol}(S)\,\text{vol}(S^c).$$

Using these two observations, it is easy to see that (6) is a continuous relaxation of (3). Using (5), it is easy to see that (6) is a relaxation for (4) as well.

The quality of approximation of relaxation (6) to sparsest cut (3) is given by Cheeger's inequality [48], [49]

$$\lambda_2/\text{vol}(\mathcal{V}) \leq \tilde{\varphi}(\mathcal{G}) \leq (8\lambda_2)^{1/2}/\text{vol}(\mathcal{V})$$

while the approximation guarantee for the minimum conductance problem (4) is

$$\lambda_2/2 \leq \varphi(\mathcal{G}) \leq (2\lambda_2)^{1/2}$$

which can be found in [50]. (A generalization of these bounds holds for arbitrary vectors [51].) Both of these approximation ratios can be realized by rounding procedures described below.

Another form of relaxation is the combinatorial model relaxation, which is formulated as problem (6) by ignoring the orthogonality constraint and restricting $x \in \{0,1\}^{|\mathcal{V}|}$ instead of $x \in \mathbb{R}^{|\mathcal{V}|}$. An extensive study of the spectral and the combinatorial model relaxation can be found in [52], while empirical comparisons between these relaxations are discussed in [53].

B. Rounding

In practice, the solution obtained by the spectral relaxation is unlikely to lie in $\{0,1\}^{|\mathcal{V}|}$, i.e., it is unlikely to be the indicator vector of a set. Therefore, it is necessary to have an efficient postprocessing procedure where the solution is rounded to a set. At the same time it is important to guarantee that the rounded solution has good worst case guarantees in terms of the conductance or sparsest cut objective.

One of the most efficient and theoretically justified rounding procedures for spectral relaxation is the sweep cut. The sweep cut procedure is summarized in the following steps.

1) Input: the solution $x \in \mathbb{R}^{|\mathcal{V}|}$ of (6).
2) Sort the indices of x in decreasing order with respect to the values of the components of x. Let $i_1, i_2, \ldots, i_{|\mathcal{V}|}$ be the sorted indices.
3) Using the sorted indices generate a collection of sets $S_j := \{i_1, i_2, \ldots, i_j\}$ for each $j \in \{1, 2, \ldots, |\mathcal{V}|\}$.
4) Compute the conductance or sparsest cut objective for each set S_j and return the minimum.

Notice that sweep cut can be used to obtain approximate solutions for both sparsest cut and minimum conductance. In fact, the proof for the upper inequalities of the approximation guarantees of spectral relaxation to sparsest cut and minimum conductance are obtained by using the sweep cut procedure [48], [49].

IV. LOCALLY-BIASED GRAPH PARTITIONING METHODS

All of the algorithms described in Section III are "global," in that they touch all of the nodes of the input graph at least once, and thus they have a running time that is at least linear in the size of the input graph. Informally, locally-biased graph clustering algorithms find clusters near a specified seed set of vertices, in many cases without even touching the entire input graph. In this section, we will describe several local graph clustering algorithms, each of which has somewhat different properties.

To understand the seemingly-quite-different algorithms we will discuss, we will distinguish local graph clustering algorithms based on three major features.

1) Weakly or strongly local algorithms: Weakly local algorithms are those that are biased toward a local part of the graph but may "touch" the entire input graph during the computation, i.e., they formally have running times that scale with the size of the graph. Strongly local graph algorithms are those that only access a small portion of the entire input graph in order to perform their computation, i.e., they formally have running times that are linear in the size of the output or input set of nodes but independent of the size of the input graph. We show that the difference between weakly and strongly local algorithms often translates to whether we penalize the solution by adding an ℓ_1-norm penalty implicitly to the objective function and/or by restricting the feasible region of the problem by adding implicitly a locality constraint. Both ways result in strongly local algorithms.

2) Localizing bias: Current local graph clustering algorithms are supervised, i.e., one has to give a reference set of seed nodes. We discuss two major ways that this information is incorporated in the problem. First, the bias to the input is incorporated in the objective function of the problem through a localizing construction we will call the reference cut graph. Second, the bias to the input is incorporated to the problem as a constraint.
3) ℓ_1 and ℓ_2 metrics: The final major feature that distinguishes locally-biased algorithms is how the cut measure is treated. Via the cut metric [54], this can be viewed as embedding the vertices of $\mathcal{G}$ and evaluating their distance in the embedding. Two distances are explicitly or implicitly used in locally-biased algorithms: the $\ell_1$and the ℓ_2 metric spaces. The former results in local flow algorithms, and the latter results in local spectral algorithms. The distinction between these two is very important in practice, as we show by empirical evaluation in subsequent sections.

The local graph clustering algorithms that we consider in the following sections and their basic properties with respect to the above three features are given in Table 1.

A. General Localizing Construction

We describe these locally-biased graph methods in terms of an augmented graph we call the "reference cut graph." We should emphasize that this is a conceptual construction to highlight the similarities and differences between locally-biased algorithms in Table I; in particular, these algorithms do *not* explicitly construct the reference cut graph.

Let $h,g \in \mathbb{R}^{|\mathcal{V}|}$, $h,g \geq 0$, and $g - h \geq 0$, and let α, β, and γ be parameters specified below. Then, the reference cut graph is constructed from a simple, undirected graph $\mathcal{G}$ as follows.

1) Add a source and sink node s and t.
2) Add edges from s to each node in $\mathcal{V}$.
3) Add edges from each node in $\mathcal{V}$ to t.
4) Weight each original edge by γ.
5) Weight the edges from s to $\mathcal{V}$ by αh, where $\alpha \geq 0$.
6) Weight the edges from t to $\mathcal{V}$ by $\beta(g - h)$, $\beta \geq 0$

Let $H := diag(h)$, $G := diag(g)$, and $Z = G - H$. Then, we can also view the augmented graph through its incidence matrix and edge weight matrix

TABLE 1 State-of-the-Art Local Graph Clustering Methods and Their Properties With Respect to the Three Features That Are Discussed in Section IV

Methods	Locality	Bias	Metric
Flow-Improve [55]	Weak	Objective	ℓ_1
MOV [33]	Weak	Constraint	ℓ_2
Local Flow-Improve [56]	Strong	Objective	ℓ_1
MQI [57]	Strong	Constraint	ℓ_1
spectral MQI [58]	Strong	Constraint	ℓ_2
ℓ_1-reg. Page-Rank [38], [41]	Strong	Objective	ℓ_2

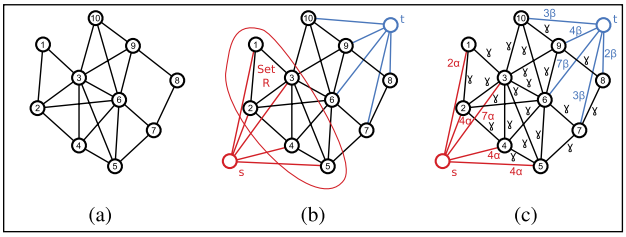

Fig. 2. ***The construction of the reference cut graph begins by adding a source node s connected to the reference set R and a sink node t connected to the rest of the graph. Then, we add weights to the network based on the degrees and three parameters α, β, and γ. Each edge to the source node is weighted by α · degree, each edge to the sink node is weighted by β · degree, and each internal edge is weighted by γ. Note that one of the choices of α, β, or γ will be 1, but various papers adopt different choices, and so we leave it general. (a) A simple graph. (b) Adding the source s and sink t. (c) The reference cut graph, with weights indicated.***

$$\tilde{B} = \begin{bmatrix} 1_{|\mathcal{V}|} & -I_{|\mathcal{V}|} & 0 \\ 0 & B & 0 \\ 0 & I_{|\mathcal{V}|} & -1_{|\mathcal{V}|} \end{bmatrix}, \quad \tilde{C} = \begin{bmatrix} \alpha H & 0 & 0 \\ 0 & \gamma C & 0 \\ 0 & 0 & \beta Z \end{bmatrix}$$

respectively. The above construction might look overly complicated. However, we will see in the following subsections that it simplifies for local spectral and flow graph clustering algorithms with specific settings for h, g, and γ.

B. Weakly Local and Strongly Local Flow Methods

Although finding the set of minimum conductance is NP-hard in general, there are a number of cases and variations that admit polynomial time algorithms and can be solved via max-flow/min-cut or a parametric max-flow method. These algorithms begin with a reference set R of nodes, and they return a smaller (or not much larger) set of nodes that is a better partition in terms of the conductance ratio φ. Typically, the returned value is optimal for a variation on the minimum conductance and/or the sparsest cut objective. The methods themselves are highly flexible and apply to other variations of minimum conductance and sparsest cut. For the sake of simplicity, we will describe them for conductance.

All of the following procedures adopt the following meta-algorithm starting with working set W initialized to an input reference set of nodes R, values $\alpha_1, \beta_1, \gamma_1$ and vectors $h = d_R$, $g = d$, where d_R is a vector of length $|\mathcal{V}|$ with components equal to d_i's for nodes $v_i \in R$ and zeros for nodes $v_i \in R^c$. Fig. 2 illustrates the construction of an augmented graph based on the previous setting of $\alpha, \beta, \gamma$and h, g.

The Local-Flow Meta-Algorithm

1) Initialize $W_1 = R$, $k = 1$, $h = d_R$, $g = d$ and $\alpha_1, \beta_1, \gamma_1$ based on R.
2) Create the reference cut graph $\tilde{B}_k, \tilde{C}_k$ based on R and $\alpha_k, \beta_k, \gamma_k$.
3) Solve the s,t-min-cut problem associated with $\tilde{B}_k, \tilde{C}_k$.
4) Set W_{k+1} to be the s-side of the cut.

5) Check if W_{k+1} has smaller conductance (or some variant of it, as we will make specific in the text below) than before and stop if not and return W_k.
6) Update $\alpha_{k+1}, \beta_{k+1}, \gamma_{k+1}$ based on W_{k+1} and R.
7) Set $k \rightarrow k+1$.
8) Repeat starting from state 2.

Next, we describe several procedures that are instantiations of this basic Local-Flow Meta-Algorithm.

1) *MQI*: The first algorithm we consider is the MQI procedure due to Lang and Rao [57]. This method is designed to take the reference set R with $\text{vol}(R) \leq \text{vol}(G)/2$ and identify a subset of it $S \subseteq R$. The method instantiates the Local-Flow Meta-Algorithm using $\alpha_k = \text{cut}(W_k)$, $\gamma_k = \text{vol}(W_k)$, $\beta_k = \infty$ and $h = d_R$, $g = d$. The idea with this method is that the reference cut graph will have an s,t -min-cut value strictly less than $\alpha_k \gamma_k$ if and only if there is a strict subset $S \subset W_k$ that has conductances less than α_k / γ_k. (See [57] for the proof.) If there is such a set, then the result $\text{vol}(R) \leq \text{vol}(G)/2$ will be a set with conductance less than α_k / γ_k. Since α_k and γ_k are picked based on the current working set W_k, at each step the algorithm monotonically improves the conductance. Also, each step minimizes the objective

$$\begin{aligned} &\text{minimize } ||\tilde{B}x||_{1,\tilde{C}_k} \\ &\text{subject to } x_i = 0 \ \forall v_i \in R^c, x_s = 1, x_t = 0. \end{aligned}$$

In fact, when the algorithm terminates, that means that there is no subset of R with conductance less than α_k / γ_k. Hence, we have solved the following variation on the conductance problem

$$\begin{aligned} &\text{minimize } \varphi(S) = \frac{\text{cut}(S)}{\text{vol}(S)} \\ &\text{subject to } S \subseteq R. \end{aligned}$$

The key difference from the standard conductance problem is that we have restricted ourselves to a subset of the reference set R. Procedurally, this is guaranteed because the edges connecting R^c to t have weight infinity, so they will never be cut. Thus, operationally, MQI is always a strongly local algorithm since it only operates within the input seed set R. Nodes connected to t with weight infinity can be agglomerated or merged into a mega-sink node T. The resulting graph has the same size as R along with the source and sink. (This is how the MQI construction is described in the original paper.) The MQI problem can be solved using the max-flow method on the resulting graph a logarithmic number of times [57]. Therefore, the running time for solving MQI depends on the max-flow algorithm that is used. Details about running times of max-flow algorithms can be found in [59].

2) *FlowImprove*: The FlowImprove method due to Andersen and Lang [55] was inspired by MQI and designed to address the weakness that the algorithm will always find an output set within the reference set R, i.e., that is a subset of R. (As an illustration, see Figure 3.) Again, FlowImprove takes as input

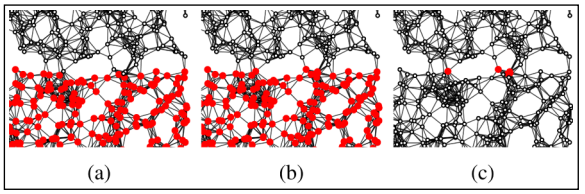

Fig. 3. ***The results of running MQI and FlowImprove on the reference set produced by a spectral partitioning method on the geometric graph (i.e., run global spectral §III-A; then round with a sweep-cut §III-B ; and then refine using MQI and FlowImprove). The FlowImprove method identifies the optimal set from Fig. 1 in this case, whereas MQI cannot because it searches only within the given set R. (a) MQI. (b) FlowImprove. (c) The difference.***

a reference set R with volume less than half the graph. The idea behind FlowImprove is that we want to find a set with conductance at least as good as R and that also is highly correlated with R. To do this, consider the following variant of conductance:

$$\varphi_R(S) = \frac{\text{cut}(S)}{\text{vol}(S \cap R) - \theta \text{vol}(S \cap R^c)}$$

where $\theta = \text{vol}(R)/\text{vol}(R^c)$, and where the value is ∞ if the denominator is negative. For any set S, $\varphi_R(S) \geq \varphi(S)$. Thus, this modified conductance score is an upper bound on the true conductance. Again, we are able to show that the Local-Flow Meta-Algorithm can solve for the exact value of $\varphi_R(S)$ in polynomial time. To do so, instantiate that algorithm with $\alpha_k = \varphi_R(W_k)$, $\beta_k = \theta \varphi_R(W_k)$, $\gamma_k = 1$ and $h = d_R$, $g = d$. The value of a cut on set S in the resulting graph is

$$\text{cut}(S) + \alpha_k \text{vol}(R) - \alpha_k [\text{vol}(S \cap R) - \theta \text{vol}(S \cap R^c)].$$

(See [60] for the justification.) Andersen and Lang show that the algorithm monotonically reduces $\varphi_R(W_k)$ at each iteration as well. Each iteration now solves the s,t-min-cut problem

$$\begin{aligned} &\text{minimize } ||\tilde{B}x||_{1,\tilde{C}_k} \\ &\text{subject to } x_s = 1, x_t = 0. \end{aligned} \tag{9}$$

In order to match precisely their FlowImprove procedure, we would need to modify our meta-algorithm to check the value of $\phi_R(W_k)$, instead of conductance (at Step 5 of the Local-Flow Meta-Algorithm above), for monotonic decrease. The authors also show that this procedure terminates in a finite number of iterations.

At termination, the FlowImprove algorithm has exactly solved minimize $\varphi_R(S), S \subseteq V$. This can be considered a locally-biased variation of the conductance objective, where we penalize departure from the reference set R. Consequently, the solutions will tend to identify small conductance sets nearby R.

FlowImprove is a very useful algorithm, but it has two small weaknesses. The first is that it is a weakly local algorithm. At each step, we have to solve a min-cut problem that is the size of the original graph. The second is that the min-cut problems do not have integer weights.

(Note that θ will almost never be an integer.) Most fast max-flow/min-cut procedures and implementations assume integer weights. For instance, many implementations of the push-relabel method (hipr [61]) only allows integer weights. Boykov and Kolmogorov's solver is a notable exception [62]. Similarly to MQI, the running time of solving the max-flow/min-cut problem depends on the particular solver that is used. A summary of max-flow/min-cut methods can be found in [59].

3) *Local FlowImprove* The Local FlowImprove algorithm due to Orecchia and Zhu [56] sought to address the weak locality of the FlowImprove method and create a strongly local flow based method. This involved two key innovations: a modification to the construction and objective that enables strong locality; and an algorithm to realize that strong locality. This Local FlowImprove method essentially interpolates between MQI and FlowImprove. In one limit, it is strictly local to the reference graph and exactly reproduces the MQI output. In the other limit, it is exactly FlowImprove. To do this, Orecchia and Zhu alter the objective function used for FlowImprove to place an additional penalty on deviating from the set R. They describe this as increasing the weight of connections β_k in the reference cut graph by scaling these by a value $\kappa \geq 1$. If $\kappa = 1$, then their construction is exactly that of FlowImprove. If $\kappa = \infty$, then this construction is equivalent to that of MQI. The effect of κ is illustrated in Fig. 4.

In terms of the optimization framework, their modification corresponds to using

$$\varphi'_R(S;\kappa) = \frac{\mathrm{cut}(S)}{\mathrm{vol}(S \cap R) - \theta\kappa\mathrm{vol}(S \cap R^c)}$$

where $\kappa \geq 1$ and $\theta = \mathrm{vol}(R)/\mathrm{vol}(R^c)$ as in FlowImprove, and again the value is ∞ if the denominator is negative. This result corresponds to instantiating the Local-Flow Meta-Algorithm using $\alpha_k = \varphi'_R(W;\kappa)$, $\beta_k = \varphi'_R(W;\kappa)\theta\kappa$ and $h = d_R$, $g = d$.

The second innovation is that they describe an algorithm to solve the min-cut problem on the reference cut graph that does not need to explore the entire graph. This second piece used a novel modification of Dinic's procedure [63] to compute a max-flow/min-cut that exploited the locality. We refer interested readers back to Orecchia and Zhu for details of this second somewhat complicated construction. In our recent work [42], however, we describe a simplified framework for the local FlowImprove method that shows that the strong locality in their modification results from implicitly regularizing the FlowImprove objective with an ℓ_1-norm regularizer. (This will mirror strongly local spectral results in the forthcoming spectral section.) In fact, our recent work [42] shows that each iteration exactly solves

$$\begin{aligned} &\text{minimize } \|\tilde{B}x\|_{1,\tilde{C}_{k'}} + \varepsilon\|Dx\|_1 \qquad (10)\\ &\text{subject to } x_s = 1, x_t = 0 \end{aligned}$$

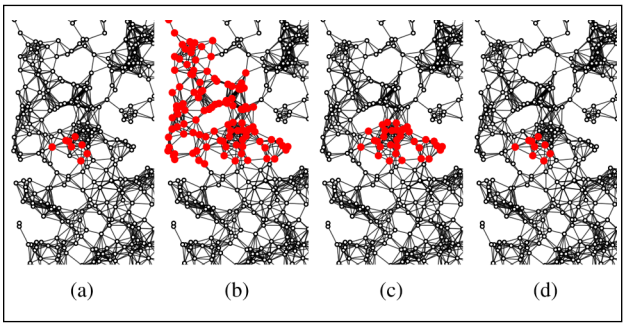

Fig. 4. *The results of running FlowImprove compared with local FlowImprove with a reference set R. The FlowImprove result returns a fairly large set whereas the Local FlowImprove results produce successively smaller sets as the penalty κ increases. When $\kappa = e^5$ then the result simply fills in the hole in the reference set. (a) Reference set R. (b) The FlowImprove result. (c) Local FlowImprove $\kappa = e^3$. (d) Local FlowImprove $\kappa = e^5$.*

where $\tilde{C}_{k'}$ is a small perturbation on the above definition and ε is a locality parameter. The volume of the output cluster S of the method in [42] is bounded $\mathrm{vol}(S) \leq \mathrm{vol}(R)(1 + 2/\varepsilon) + E(R, R^c)$, where $\varepsilon := \mathrm{vol}(R)/\mathrm{vol}(R^c) + \delta$ and $\delta \geq 0$ is a constant.

That work also describes a simple procedure to realize the strong locality that leverages any max-flow/min-cut solver on a sequence of subproblems whose size is bounded independent of the graph.

C. Weakly and Strongly Local Spectral Methods

There are spectral analogs for each of the three flow-based constructions on the augmented graph. The development of these ideas occurred in parallel, largely independently, and it was not obvious until recently that the ideas were very related. Here, we make these connections explicit. Of the three flow constructions, the simplest is the MQI objective. We begin with it.

1) SpectralMQI: The method we call SpectralMQI was proposed as the Dirichlet partitioning problem in [58]. Given a graph $\mathcal{G}$ and a subset of vertices, consider finding the set S of minimal local conductance $\varphi'(S) = \mathrm{cut}(S)/\mathrm{vol}(S)$ such that $S \subseteq R$, where, again, R is a reference set specified in the input. Note that the only difference from conductance is that we do not have the minimum in the denominator. A spectral algorithm to find an approximate minimizer of this is to solve the generalized eigenvector problem

$$\begin{aligned} &\lambda_R = \text{minimize } \frac{\|Bx\|^2_{2,C}}{\|x\|^2_{2,D}}\\ &\text{subject to } x_i = 0 \ \forall v_i \in R^c. \end{aligned}$$

The solution vector x and value λ_R are related to the smallest eigenvalue of the submatrix of the normalized Laplacian

corresponding to the nodes in R. (Note that we take the submatrix of the normalized Laplacian, rather than the normalized Laplacian on the subgraph induced by R.) A sweep cut over the eigenvector x produces a set S that satisfies a Cheeger inequality with respect to the best possible solution [58]. This definition of local conductance is also called NCut′ by Hochbaum [64], who gave a polynomial time algorithm to compute it that is closely related to the MQI procedure. In this case, if R has volume less than half the graph, then this is exactly the spectral analogue of the MQI procedure and the result is a Cheeger-like bound on the overall conductance.

2) MOV: The Mahoney–Orecchia–Vishnoi (MOV) objective [33] is a spectral analog of the FlowImprove method, with a few subtle differences. The goal is to find a small Rayleigh quotient, as in (6), that is highly correlated with an input vector $z \in R^{|V|}$, where z represents the seed or local-bias. Given this, the MOV objective is

$$\begin{aligned} &\text{minimize} \ \frac{\|Bx\|_{2,C}^2}{\|x\|_{2,D}^2} \\ &\text{subject to} \ 1_{|V|}^T Dx = 0 \\ &\qquad (z^T Dx)^2 \geq k \qquad x \in R^V. \end{aligned}$$

The solution of this problem represents an embedding of the nodes in V which is locally biased, i.e., large values for components/nodes that are considered important and small or zero values for the rest.

According to [33], there is a constant ρ, i.e., the optimal dual variable for the locally-biased constraint, such that the solution to the MOV problem satisfies $(L + \rho D)x = \rho Dz$. The null space of L is the vector $1_{|V|}$, and assuming that $1_{|V|}^T Dz = 0$, then the solution to the previous system is unique. One final detail is that the MOV construction fixes $||x||_2 = 1$. Consequently, the MOV solution is

$$x = c(L + \rho D)^{\dagger} Dz \quad c = \left(\left\|(L + \rho D)^{\dagger} Dz\right\|_2\right)^{-1}. \tag{11}$$

In the MOV paper [33], they show that ρ can be chosen such that $x^T Dz = \kappa$, the desired correlation strength with the input vector z, through a simple bisection procedure. Solving the linear system (11) results in a weakly local method that satisfies another Cheeger-like inequality. Recent extensions show that it is possible to get multiple locally-biased vectors that are akin to eigenvectors from this setup [65], [66]. The methodology is able to leverage the large number of Laplacian system solvers [67] that can find an approximate solution to (11) in nearly linear time.

The pseudoinverse allows us to "pass through" $\rho = 0$ and approach $\rho = -\lambda_2$. (This system is singular at $\rho = 0$ and $\rho = \lambda_2$.) What is interesting is that taking the limit $\rho \to -\lambda_2$ directly maps to the spectral relaxation (6). Thus, the ρ parameter interpolates between the global spectral relaxation (6) and a spectral version of the min-cut problem in each step of FlowImprove.

Based on the reference cut graph, the MOV objective is minimize $||\tilde{B}\tilde{x}||_{2,\tilde{C}}^2$, where $\tilde{x} := [1; x; 0]$. The reference graph cut setting is $\gamma = 1$, $g = d$, $h = Dz$ and $\alpha = \beta = \rho \geq 0$ controls the desired strength of the correlation to the input vector z. Notice that the MOV problem is a spectral, i.e., ℓ_2, version of the s,t-min-cut problem. This observation was made first in [41], [68]. If ρ is extremely large, the solution to the above problem would have perfect correlation with the input vector h. As $\rho \to 0$, we decrease the effective correlation with the input vector h. (These arguments are formal in [33].)

3) ℓ_1-Regularized PageRank: The ℓ_1-regularized PageRank problem was initially studied in [41] and then further refined in [37]. In the latter work, the problem is defined as

$$\text{minimize} \ \frac{1}{2}||\tilde{B}\tilde{x}||_{\tilde{C},2}^2 + \varepsilon ||Dx||_1 \tag{12}$$

where $\tilde{x} := [1; x; 0]$. The reference cut graph setting for (12) is $g = d$ and $h \geq 0$ is a vector that satisfies $||h||_1 = 1$ and $||h||_\infty \geq \varepsilon$. The latter condition is to guarantee that the solution to (12) is not the zero vector. Moreover, $\alpha = \beta$ and $\gamma = (1 - \alpha)/2$. Similarly to z for MOV, the vector h controls the input seed set and the weights of nodes in that set. The larger the weights the more the solution will be correlated with the corresponding nodes in the input seed set. The solution vector to problem (12) is component-wise nonnegative and the parameter α controls how much energy is concentrated close to the input seed set. Formally, based on theoretical guarantees in [38] the vector h should be an indicator vector for a single seed node, around which there is a target cluster of nodes C. The algorithm is not guaranteed to find the exact target cluster C, but if C has conductance less than $\alpha/10$ then it is guaranteed to return a cluster with conductance of $\mathcal{O}(\sqrt{\alpha \log(\text{vol}(C))})$. We refer the reader to [38] for a detailed description of the theoretical graph clustering guarantees.

The idea of ℓ_1-regularized PageRank graph clustering initially appeared in [38] in the form of implicit regularization. In particular, the authors in [38] suggest solving a personalized PageRank linear system approximately. In [37] and [41], the authors noticed that the termination criteria in [38] are related to the first-order optimality conditions of the above ℓ_1-regularized PageRank problem, and they draw the connection to explicit ℓ_1- regularization. It is shown in [37] that solving the ℓ_1-regularized PageRank problem has the same Cheeger-like worst case approximation guarantees to the minimum conductance problem as the original procedure in [38]. However, there is an important technical difference: one advantage of solving the ℓ_1-regularized problem is that the locality of the solution is a property of the optimization problem as opposed to a property of an algorithm. In particular, by solving the ℓ_1-regularized problem it is guaranteed to obtain the same solution regardless of the algorithm used. In comparison, applying the procedure in [38], where

the output depends on the setting of the procedure, i.e., the strategy for choosing nodes to be updated at every iteration, leads to somewhat different solutions, depending on the specific settings chosen.

Let x^* be the optimal solution of (12) and S^* be the set of nodes where x^* is nonzero. In [37], it is shown that many standard optimization algorithms such as iterative soft thresholding or block iterative soft thresholding solve (12) with running time $\mathcal{O}(\mathrm{vol}(S^*)/\alpha)$, which can be independent of the volume of the whole graph $\mathrm{vol}(\mathcal{V})$. This opens up the possibility of the use of these algorithms more generally. For details about the algorithms, we refer the reader to [37].

V. EMPIRICAL EVALUATION

In this section, we illustrate differences among global, weakly local, and strongly local solutions to the problems discussed in Section IV. Additionally, we discuss differences between spectral and flow methods (which is equivalently ℓ_2 versus ℓ_1 metrics for node distances).

To do so, we make use of the following real-world undirected and unweighted networks.

- US-Senate. Each node in this network is a Senator that served in a single term (two years) of Congress. Our data cover the period from year 1789 to 2008. Senators appear as multiple nodes if they served in multiple terms. Edges are based on the similarity of voting records between Senators and thresholded at the maximum similarity such that the graph remains connected. Edge weights are discarded. For a detailed discussion of this data set we refer the reader to [22]. This graph has 8974 nodes and 153 804 edges. This graph has two large clusters with small conductance ratio, i.e., downward-slopping network community profile; see [9, Fig. 6] for details. The first cluster consists of all the nodes before the year 1913 and the second cluster consists of nodes after that year.
- CA-GrQc. The data for this graph are a general relativity and quantum cosmology collaboration network. Details can be found in the Stanford Network Analysis Project.[2] This graph has 4158 nodes and 13 422 edges. This graph has many clusters of small size with small conductance ratio, while large clusters have large conductance ratio, i.e., upward-slopping network community profile; see [9, Fig. 6] for details.
- FB-Johns55. This graph is a Facebook anonymized data set on a particular day in September 2005 for a student social network at John Hopkins University. The graph is unweighted and it represents "friendship" ties. The data form a subset of the

[2] http://snap.stanford.edu/data

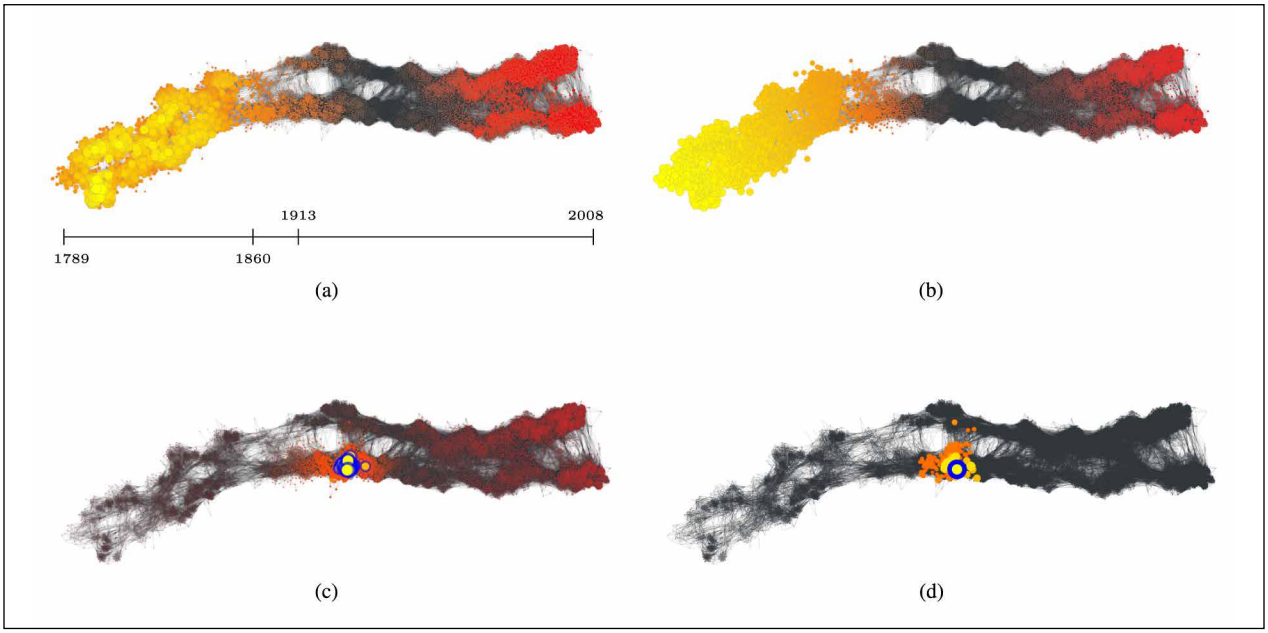

Fig. 5. *US-Senate. This figure shows the solutions of (a) spectral relaxation; (b) MOV with global bias; (c) MOV with local bias; and (d) strongly local ℓ_1-regularized PageRank. We use a heat map to represent the weights of the nodes. For spectral relaxation and MOV bright yellow means large positive and bright red means large negative. For ℓ_1-regularized PageRank bright yellow means large positive and bright red means small positive. The blue halo around a node in (c) and (d) means that this node is included in the seed set. The size of the nodes shows the weights of the solution in absolute value. If a weight is nearly zero then the corresponding node is barely visible. (a) Global spectral relaxation. (b) MOV with global seed. (c) MOV with local seed. (d) ACL ℓ_1-regularized PageRank.*

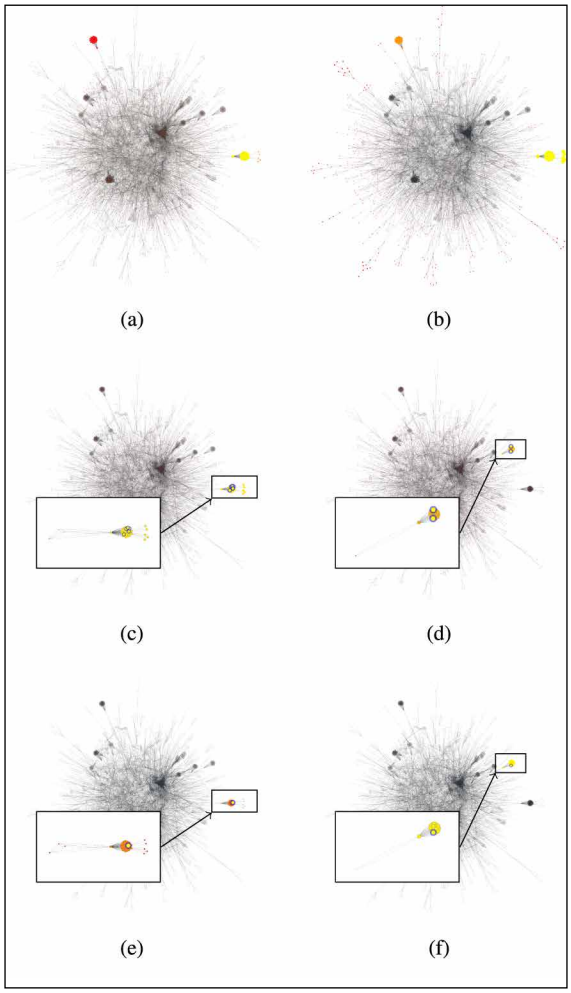

Fig. 6. ***CA-GrQc. This figure shows the solutions of spectral relaxation, MOV with global input, MOV with local input, and ℓ_1-regularized PageRank. The meaning of the colors of the nodes and its sizes is the same as in Fig. 5. (a) Spectral relaxation. (b) MOV global. (c) MOV local, seed 1. (d) MOV local, seed 2. (e) ℓ_1-regularized PageRank, Seed 1. (f) ℓ_1-regularized PageRank, Seed 2.***

Facebook100 data set from [24] and [69]. This graph has 5157 nodes and 186 572 edges. This is an expander-like graph, all small and large clusters have about the same conductance ratio, i.e., flat network community profile; see [9, Fig. 6] for details.

- US-Roads. The data for this graph is from the National Highway Planning Network [6]. Each node in this network is an intersection between two highways and the edges represent segments of the highways themselves.

Note that the small-scale versus large-scale clustering properties of the first three networks have been characterized previously [9]. In addition, it is known that US-Roads has a downward-sloping network community profile.

A. Global, Weakly Local, and Strongly Local Solutions

We first demonstrate differences among global, weakly local, and strongly local algorithms. Let us start with a comparison among spectral algorithms. By comparing algorithms that use that same metric, i.e., ℓ_2, to measure distances among nodes we minimize factors that can affect the solution, and we focus on weak versus strong locality. In all figures we show the solution obtained by an algorithm without applying any rounding procedure. We illustrate the importance of the nodes by coloring and size; details are explained in the captions of the figures and in the text. The layout for all graphs has been obtained using the force-directed algorithm [70], which is available from the graph-tool project.[3]

For US-Senate, the comparison is shown in Fig. 5. Fig. 5(a) and (b) shows the solutions of global algorithms, spectral relaxation, and MOV global ($z = 1_{|\mathcal{V}|}$ and then we orthogonalize z with respect to $D1_{|\mathcal{V}|}$), respectively. As expected, the US-Senate graph has two large clusters, i.e., before the year 1913 and after that year, that partition along the 1-D time axis. This global structure is nicely captured by spectral relaxation and MOV global in Fig. 5(a) and (b), respectively.

Given an input seed set, Fig. 5(c) and (d) illustrates the weakly and strongly local solutions by MOV and ℓ_1-regularized PageRank, respectively. For MOV in Fig. 5(c) we set $z_i = 1$ for all i in the input seed set and $z_i = 0$ for all i outside the input seed set. Then, we orthogonalize z with respect to $D \cdot 1_{|\mathcal{V}|}$. For ℓ_1-regularized PageRank, we only give a single node as an input seed set, i.e., $h_i = 1$ where i is the input node and $h_i = 0$ for all other nodes. Moreover, we set the locality parameter ε large enough such that the solution is very sparse, i.e., strongly local. In Fig. 5(c) and (d), we demonstrate the input seed sets by nodes with a blue halo around them. In Fig. 5(c), the cluster which is found by MOV consists of the nodes which have large mass concentration around the input seed set, i.e., the nodes around the input seed set that have large size and are colored with a bright red shade. MOV recovers this cluster by examining the whole graph; each node has a weight assigned to it in Fig. 5(c). On the other hand, a similar cluster is found in Fig. 5(d) by using ℓ_1-regularized PageRank without examining the whole graph. This is possible because nodes of the graph have zero weight assigned and need not be considered. This speed and data advantage, along with the sparsity-based implicit regularization [41], are some of the reasons that strongly local algorithms, such as ℓ_1-regularize PageRank, are used so often in practice [7], [9].

In Fig. 6, we present global, weakly local, and strongly local solutions for the less well-partitionable and thus less easily-visualizable CA-GrQc graph. As already mentioned in the description of this data set, this graph has many small clusters with small conductance ratio and large clusters have large ratio. This is also justified in our experiment

[3]https://graph-tool.skewed.de

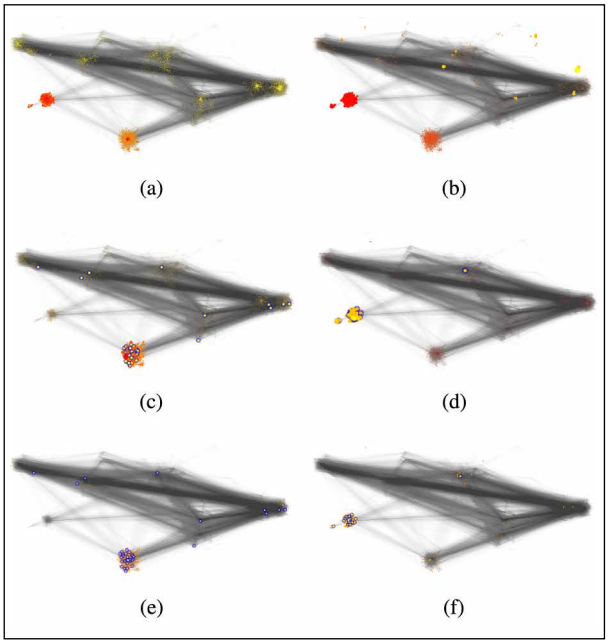

Fig. 7. *FB-Johns55. This figure shows the solutions of spectral relaxation, MOV with global input, MOV with local input, and ℓ_1-regularized PageRank. The meaning of the colors of the nodes and its sizes is the same as in Fig. 5. (a) Spectral relaxation. (b) MOV global. (c) MOV local. (d) MOV local, seed 2. (e) ℓ_1-regularized PageRank. (f) ℓ_1-regularized PageRank, Seed 2.*

by the fact that global methods, such as the spectral relaxation and MOV global in Fig. 6(a) and (b), respectively, recover small clusters. The two global procedures find small clusters which are presented in Fig. 6(a) and (b) with red, orange, and yellow colors. However, since there are many small clusters of small conductance ratio, one might want to find different clusters than the ones obtained by spectral relaxation and MOV global. This is possible using localized procedures such as MOV and ℓ_1-regularized PageRank. Given two different seed sets we demonstrate in Fig. 6(c) and (d) that MOV successfully finds other clusters than the ones obtained by the global methods. The same is shown in Fig. 6(e) and (f) for ℓ_1-regularized PageRank. Notice that MOV assigns weights (perhaps small) to all the nodes of the graph; on the other hand, ℓ_1-regularized PageRank, as a strongly local procedure, assigns weights only to a small number of nodes, without examining all of the graph.

We now use the FB-Johns55 graph which has an expander-like behavior at all size scales, i.e., all small and large clusters have large conductance ratio. See [9, Fig. 6] for details. We present the results of this experiment in Fig. 7. Notice that in Fig. 7(a) and (b) the global methods identify three main clusters, one small (red nodes), one medium size (orange nodes), and one large (yellow nodes). All these clusters have similar conductance ratio. In Fig. 7(c) and (d), we show that MOV can recover the medium or small size clusters, respectively, by giving a localized seed set. In Fig. 7(e) and (f), we illustrate that using ℓ_1-regularized PageRank one can find very similar clusters while exploiting the strongly local running time of the method.

Let us now present the performance of flow-based algorithms on the same graphs. We begin with US-Senate in Fig. 8. In this figure, the red nodes are part of the solution of FlowImprove or Local FlowImprove, depending on the experiment; the yellow nodes are part of the seed set only; and the orange nodes are in both the solution of the flow algorithm and the input seed set. In Fig. 8(a), we used as an input seed set to FlowImprove the cluster obtained by applying sweep cut with respect to the conductance ratio on the spectral relaxation solution. Fig. 8(b) and (c) presents a clear distinction between FlowImprove and Local FlowImprove, weakly and strongly local algorithms, respectively. For both figures, the input seed set is located at about the middle of the graph. FlowImprove as a weakly local algorithm examines the whole graph and returns a cluster which includes the period before 1913. Also, it includes big part of the input seed set in the cluster due

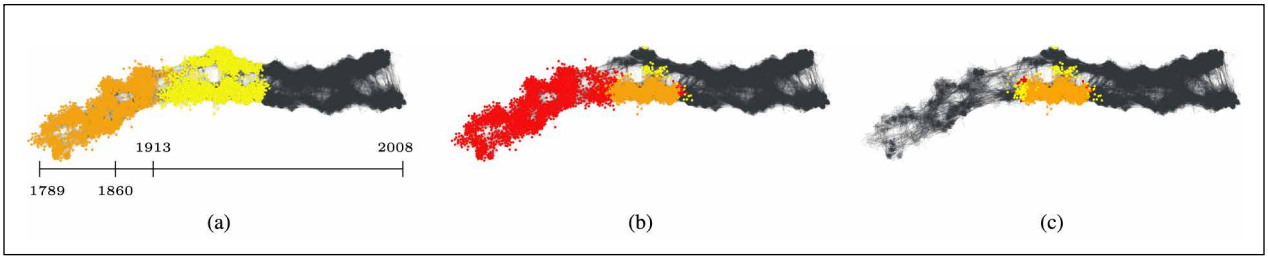

Fig. 8. *US-Senate. This figure shows the solutions of FlowImprove and Local FlowImprove for various input seed sets. Red nodes are only FlowImprove or Local FlowImprove, depending on the experiment; yellow nodes are only seed set; and orange nodes are both part of the flow algorithm and the seed set. (a) Local FlowImprove, seed: Spectral relaxation + sweep cut. (b) FlowImprove, local seed set. (c) Local FlowImprove, local seed set.*

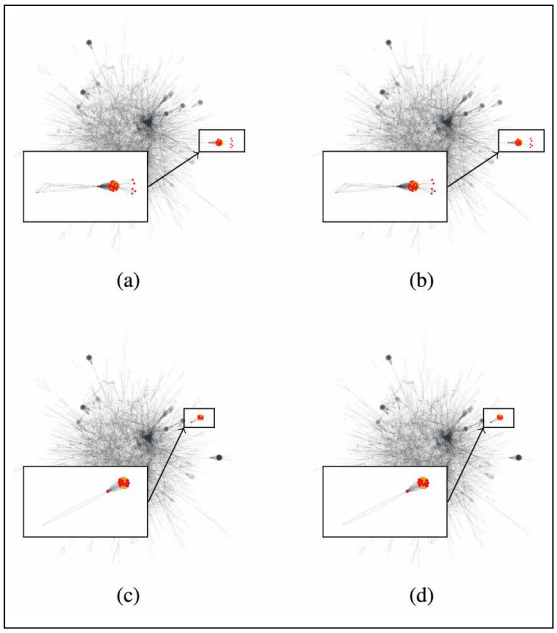

Fig. 9. ***CA-GrQc. This figure shows the solutions of FlowImprove and Local FlowImprove for various input seed sets. Red nodes are only FlowImprove or Local FlowImprove, depending on the experiment; yellow nodes are only seed set; and orange nodes are both part of the flow algorithm and the seed set. (a) FlowImprove, seed 1. (b) Local FlowImprove, seed 1. (c) FlowImprove, local seed 2. (d) Local FlowImprove, seed 2.***

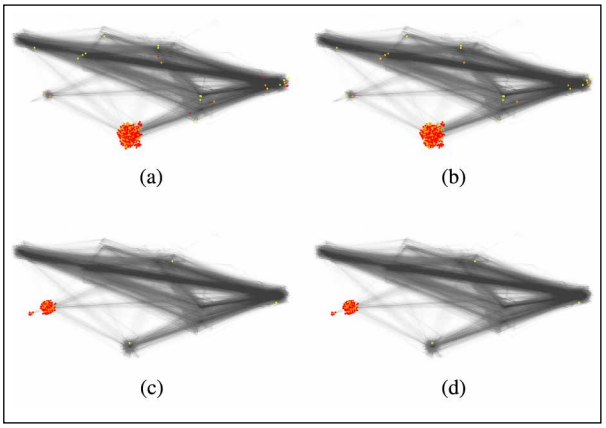

Fig. 10. ***FB-Johns55. This figure shows the solutions of FlowImprove and Local FlowImprove for various input seed sets. Red nodes are only FlowImprove or Local FlowImprove, depending on the experiment; yellow nodes are only seed set; and orange nodes are both part of the flow algorithm and the seed set. (a) FlowImprove, seed 1. (b) Local FlowImprove, seed 1. (c) FlowImprove, seed 2. (d) Local FlowImprove, seed 2.***

to the overlapping regularization term in the denominator of its objective function. See the definition of the objective function ϕ_R for FlowImprove in Section IV. On the other hand, in Fig. 8(c) Local FlowImprove as a strongly local algorithm does not examine the whole graph and its solution is concentrated only around the input seed set.

The distinction that we discussed in the previous paragraph between FlowImprove and Local FlowImprove is easy to visualize in the relatively well-structured US-Senate, but it is not so clear in all graphs. For example, in Fig. 9, we present the performance of these two algorithms for the CA-GrQc graph. Since this graph has only small clusters of small conductance ratio, FlowImprove and Local FlowImprove find the same clusters. This is clearly shown by comparing Fig. 9(a) and (b) and Fig. 9(c) and (d). A similar performance is observed for the FB-Johns55 graph in Fig. 10, except that the solutions of FlowImprove and Local FlowImprove are not exactly the same but only very similar.

B. Flow Versus Spectral or ℓ_1 Versus ℓ_2

Spectral algorithms measure distances of the nodes based on the ℓ_2-norm. Generally this means that the nodes of the graph are embedded on the real line. On the other hand, flow algorithms measure distances of the nodes based on the ℓ_1-norm. The solution to flow-based algorithms that we discussed is binary, either a node is added in the solution with weight 1 or it is not and it has weight 0. In this case, the objective function $\|Bx\|_{1,C}$ of the flow algorithms is a locally-biased variation on cut($\mathcal{S}$), where $\mathcal{S}$ is constructed based on the binary x. Therefore, the flow algorithms aim to find a balance between finding good cuts and identifying the input seed set. This implies that the flow algorithms try to minimize the absolute number of edges that cross the partition, but at the same time they try to take into account the volume regularization effect of the denominator in the objective function.

In this section, we will try to isolate the effect of ℓ_1 and ℓ_2 metrics in the output solution. We do this by employing MQI and spectral MQI, which are flow (i.e., ℓ_1) and spectral (i.e., ℓ_2) problems, respectively. The first set of results is shown in Fig. 11. Notice in Fig. 11(a) and (b) that MQI and spectral MQI + sweep cut recover the large clusters, i.e., before and after the year 1913. There are only minor differences between the two solutions. Moreover, observe that spectral MQI returns a solution which is not binary. This is illustrated in Fig. 11(c), where the weights of the nodes are real numbers. Then sweep cut is applied on the solution of spectral MQI to obtain a binary solution with small conductance ratio, i.e., Fig. 11(b).

The previous example did not reveal any difference between MQI and spectral MQI other than the fact that spectral MQI has to be combined with the sweep cut rounding procedure to obtain a binary solution. In Fig. 12, we present a result showing where the solutions have substantial differences. The graph that we used for this is the US-Roads, and the input seed set consists of nodes near Minneapolis together with some suburban areas around the city. Notice in Fig. 12(a) that MQI, i.e., ℓ_1 metric, shrinks the boundaries of the input seed set. However, MQI does not accurately recover Minneapolis. The reason is the volume regularization

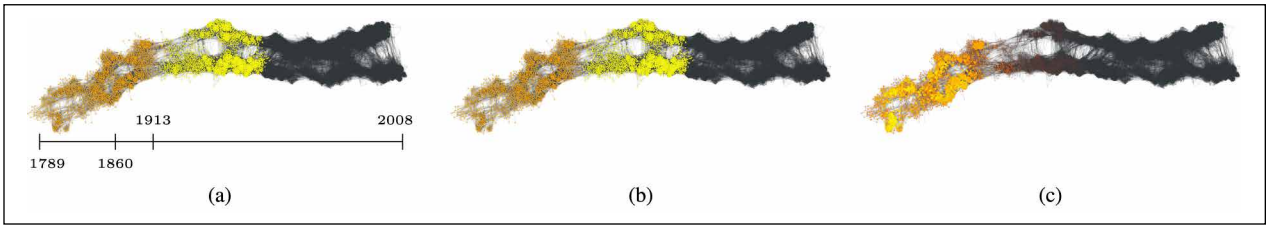

Fig. 11. ***US-Senate. This figure shows the solutions of MQI, spectral MQI, spectral MQI + sweep cut given the solution of spectral relaxation + sweep cut as an input seed set. For (a) and (b) the red nodes are only MQI or spectral MQI + sweep cut depending on the experiment; yellow nodes are only seed set; and orange nodes are both part of the flow or spectral algorithm and the seed set. (c) shows the solution of spectral MQI without sweep cut. For (c), we use a heat map to represent the weights of the nodes. Bright yellow means large positive and bright red means small positive. The size of the nodes shows the weights of the solution in absolute value. (a) MQI, seed: Spectral relaxation + sweep cut. (b) Spectral MQI + sweep cut, seed: Spectral relaxation + sweep cut. (c) Spectral MQI, seed: Spectral relaxation + sweep cut.***

which is imposed by the denominator of the objective function of MQI. This regularization forces the solution to have large volume. On the other hand, spectral MQI + sweep cut in Fig. 12(b) recovers Minneapolis. The reason is that for spectral MQI the regularization effect of the denominator is unimportant since the objective function is invariant to scalar multiplications of the solution vector. It is the solution of spectral MQI, i.e., the eigenvector of smallest eigenvalue, which is presented in Fig. 12(c), that is concentrated closely around Minneapolis. Due to this concentration of the eigenvector around Minneapolis, the sweep is successful. Briefly, spectral MQI, which is a continuous relaxation of MQI, implicitly offers an additional level of volume regularization, which turns out to be useful in this example.

Finally, we present another set of results using the FB-Johns55 graph in Fig. 13. As we saw before, notice that for this less well-structured graph the solutions of MQI and spectral MQI + sweep cut are nearly the same. This happens because the regularization effect of the denominator of MQI and the regularization imposed by spectral MQI have nearly the same effect on this graph. This is also justified by the fact that MQI in Fig. 13(a) and spectral MQI without sweep cut in Fig. 13(c) recover nearly the same cluster.

VI. DISCUSSION AND CONCLUSION

Although the optimization approach we have adopted is designed to highlight similarities between different variants of locally-biased graph algorithms, it is also worth emphasizing that there are a number of quite different and complementary perspectives people in different research communities have adopted thus far on these methods. Here are the examples.

1) Theoretical and empirical. The theoretical implications of these locally-biased algorithms are often used to improve the performance of long-standing problems in theoretical computer science by improving runtimes, improving approximation constants, and handling special cases. Empirically, these algorithms are used to study real-world data and to accelerate and improve performance on discovery and prediction tasks. Due to the strong locality, the fast runtimes for theory often manifest as extremely fast algorithms in practice. Well-implemented strongly-local algorithms often have runtimes in milliseconds even on billion-edge graphs [37], [39].

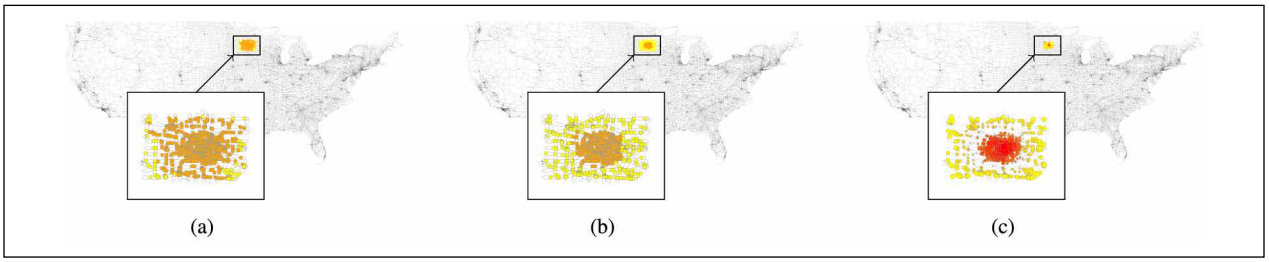

Fig. 12. ***US-Roads. This figure shows the solutions of MQI, spectral MQI, spectral MQI + sweep cut given Minneapolis and its suburban areas as an input seed set. The meaning of the colors of the nodes and its sizes is the same as in Fig. 11. (a) MQI, seed: Minneapolis and suburban areas. (b) Spectral MQI + sweep cut, seed: Minneapolis and suburban areas. (c) Spectral MQI, seed: Minneapolis and suburban areas.***

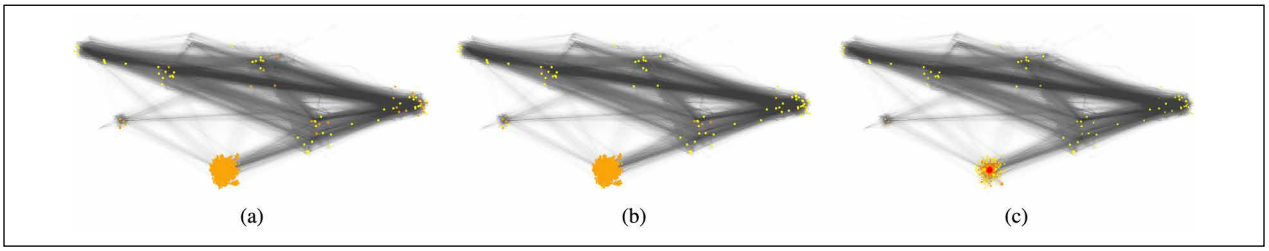

Fig. 13. ***FB-Johns55. This figure shows the solutions of MQI, spectral MQI, spectral MQI + sweep cut for a given input seed set. The meaning of the colors of the nodes and its sizes is the same as in Fig. 11. (a) MQI. (b) Spectral MQI + sweep cut. (c) Spectral MQI.***

2) Algorithmic and statistical. Some of the work is motivated by having better algorithmic results, e.g., being fast and/or being a rigorous approximation algorithm, i.e., worst case guarantees in terms of approximating the optimal solution of a combinatorial problem, while other work has provided an interpretation in terms of statistical properties, e.g., explicitly optimizing a regularized objective [71] or implicitly having output that are nice in some sense, i.e., well-connected output cluster [72]. Often, locally-biased algorithms alone suffice as the result is an improvement to some downstream activity that will necessarily look at all the data anyway.
3) Optimization and operational. The locally-biased methods tend to result from stating an optimization problem and solving it with some sort of black box or white box. Strongly local algorithms often arise by studying a specific procedure on a graph and showing that it satisfies some condition, e.g., that it terminates so quickly that it cannot explore the entire graph, that it leads to a solution with certain quality-of-approximation guarantees, etc. See, for instance, the spectral algorithms [38]–[40], [60], [73]–[75] and the flow-based algorithms [56], [76], [77].

In light of these complementary approaches as well as the ubiquity with which graphs are used to model data, we expect that locally-biased graph algorithms and our optimization perspective on locally-biased graph algorithms will find increasing usefulness in many application areas. ■

REFERENCES

[1] N. Tuncbag *et al.*, "Network modeling identifies patient-specific pathways in glioblastoma," *Sci. Rep.*, vol. 6, p. 28668, Jun. 2016.

[2] D. S. Bassett and E. Bullmore, "Small-world brain networks," *Neuroscientist*, vol. 12, no. 6, pp. 512–523, Dec. 2006.

[3] O. Sporns, "Network analysis, complexity, and brain function," *Complexity*, vol. 8, no. 1, pp. 56–60, Sep. 2002.

[4] J. L. Morrison, R. Breitling, D. J. Higham, and D. R. Gilbert, "GeneRank: Using search engine technology for the analysis of microarray experiments," *Bioinformatics*, vol. 6, no. 1, p. 233, Sep. 2005.

[5] E. Estrada and N. Hatano, "Communicability graph and community structures in complex networks," May 2009. [Online]. Available: https://arxiv.org/abs/0905.4103

[6] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney, "Statistical properties of community structure in large social and information networks," in *Proc. 17th Int. Conf. World Wide Web*, Apr. 2008, pp. 695–704.

[7] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Math.*, vol. 6, no. 1, pp. 29–123, 2009.

[8] J. Leskovec, K. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proc. 19th Int. Conf. World Wide Web*, Apr. 2010, pp. 631–640.

[9] L. G. S. Jeub, P. Balachandran, M. A. Porter, P. J. Mucha, and M. W. Mahoney, "Think locally, act locally: Detection of small, medium-sized, and large communities in large networks," *Phys. Rev. E*, vol. 91, Jan. 2015, Art. no. 012821.

[10] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, "Models of social networks based on social distance attachment," *Phys. Rev. E*, vol. 70, no. 5, Nov. 2004, Art. no. 056122.

[11] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki, "A comprehensive two-hybrid analysis to explore the yeast protein interactome," *Proc. Nat. Acad. Sci.*, vol. 98, no. 8, pp. 4569–4574, 2001.

[12] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, Apr. 2002.

[13] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," in *Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, Aug. 1999, pp. 251–262.

[14] A. Réka, H. Jeong, and A.-L. Barabási, "Internet: Diameter of the world wide web," *Nature*, vol. 401, pp. 130–131, Sep. 1999.

[15] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins, "The Web as a graph: Measurements, models, and methods," in *Proc. 5th Annu. Int. Conf. Comput. Combinatorics*, 1999, pp. 1–17.

[16] J. Scott, *Social Network Analysis*. London, U.K.: Sage, 2013.

[17] A. L. Traud, P. J. Mucha, and M. A. Porter, "Social structure of Facebook networks," 2011. [Online]. Available: https://arxiv.org/abs/1102.2166

[18] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the Facebook social graph," 2011. [Online]. Available: https://arxiv.org/abs/1111.4503

[19] J. P. Onnela *et al.*, "Structure and tie strengths in mobile communication networks," *Proc. Nat. Acad. Sci.*, vol. 104, no. 18, pp. 7332–7336, 2007.

[20] P. Expert, T. S. Evans, V. D. Blondel, and R. Lambiotte, "Uncovering space-independent communities in spatial networks," *Proc. Nat. Acad. Sci.*, vol. 108, no. 19, pp. 7663–7668, 2011.

[21] 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. USA*, vol. 102, no. 20, pp. 7057–7062, 2005.

[22] P. Mucha, T. Richardson, K. Macon, M. Porter, and J. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *Sciences*, vol. 328, no. 5980, pp. 876–878, 2010.

[23] K. T. Macon, P. J. Mucha, and M. A. Porter, "Community structure in the united nations general assembly," *Phys. A, Stat. Mech. Appl.*, vol. 391, nos. 1–2, pp. 343–361, 2012.

[24] 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.*, vol. 53, no. 3, pp. 526–543, 2011.

[25] M. C. González, H. J. Herrmann, J. Kertész, and T. Vicsek, "Community structure and ethnic preferences in school friendship networks," *Phys. A, Stat. Mech. Appl.*, vol. 379, no. 1, pp. 307–316, 2007.

[26] A. C. Lewis, N. S. Jones, M. A. Porter, and C. M. Deane, "The function of communities in protein interaction networks at multiple scales," *Syst. Biol.*, vol. 4, no. 100, pp. 1–14, 2010.

[27] D. S. Basset, E. T. Owens, K. E. Daniels, and M. A. Porter, "Influence of network topology on sound propagation in granular materials," *Phys. Rev. E*, vol. 86, Oct. 2012, Art. no. 041306.

[28] P. Ronhovde *et al.*, "Detecting hidden spatial and spatio-temporal structures in glasses and complex physical systems by multiresolution network clustering," *Eur. Phys. J. E*, vol. 34, no. 105, pp. 1–24, 2012.

[29] D. S. Bassett, N. F. Wymbs, M. A. Porter, P. J. Mucha, J. M. Carlson, and S. T. Grafton, "Dynamic reconfiguration of human brain networks during learning," *Proc. Nat. Acad. Sci.*, vol. 108, no. 18, pp. 7641–7646, 2011.

[30] N. F. Wymbs, D. S. Bassett, P. J. Mucha, M. A. Porter, and S. T. Grafton, "Differential recruitment of the sensorimotor putamen and frontoparietal cortex during motor chunking in humans," *Neuron*, vol. 74, no. 5, pp. 936–946, 2012.

[31] D. S. Basset, N. F. Wymbs, M. P. Rombach, M. A. Porter, P. J. Mucha, and S. T. Grafton, "Task-based core-periphery organization of human brain dynamics," *PLoS Comput. Biol.*, vol. 9, no. 9, p. e1003171, 2013.

[32] T. S. Evans, R. Lambiotte, and P. Panzarasa, "Community structure and patterns of scientific collaboration in business and management," *Scientometrics*, vol. 89, no. 1, pp. 381–396, 2011.

[33] M. W. Mahoney, L. Orecchia, and N. K. Vishnoi, "A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally," *J. Mach. Learn. Res.*, vol. 13, pp. 2339–2365, Aug. 2012.

[34] R. Andersen and K. Lang, "An algorithm for improving graph partitions," in *Proc. 19th ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2008, pp. 651–660.

[35] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Annu. Adv. Neural Inf. Process. Syst.*, 2004, pp. 321–328.

[36] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu, "Automatic multimedia cross-modal correlation discovery," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 653–658.

[37] K. Fountoulakis, X. Cheng, J. Shun, F. Roosta-Khorasani, and M. W. Mahoney, "Exploiting optimization for local graph clustering." 2016. [Online]. Available: https://arxiv.org/abs/1602.01886

[38] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *Proc. 47th Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2006, pp. 475–486.

[39] K. Kloster and D. F. Gleich, "Heat kernel based community detection," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1386–1395.

[40] D. A. Spielman and S. H. Teng, "A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning," *SIAM J. Sci. Comput.*, vol. 42, no. 1, pp. 1–26, 2013.

[41] D. F. Gleich and M. W. Mahoney, "Anti-differentiating approximation algorithms: A case study with min-cuts, spectral, and flow," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1018–1025.

[42] L. N. Veldt, D. F. Gleich, and M. W. Mahoney, "A simple and strongly-local flow-based method for cut improvement," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1938–1947.

[43] M. W. Mahoney and L. Orecchia, "Implementing regularization implicitly via approximate eigenvector computation," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 121–128.

[44] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc., B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[45] F. Shahrokhi, "The maximum concurrent flow problem," *J. ACM*, vol. 37, no. 2, pp. 318–334, 1990.

[46] T. Leighton and S. Rao, "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms," *J. ACM*, vol. 46, no. 6, pp. 787–832, 1999.

[47] S. Arora, S. Rao, and U. V. Vazirani, "Expander flows, geometric embeddings and graph partitioning," *J. ACM*, vol. 56, no. 2, 2009, Art. no. 5.

[48] G. F. Lawler and A. D. Sokal, "Bounds on the L^2 spectrum for Markov chains and Markov processes: A generalization of Cheeger's inequality," *Trans. Amer. Math. Soc.*, vol. 309, no. 2, pp. 557–580, 1988.

[49] A. Sinclair and M. Jerrum, "Approximate counting, uniform generation and rapidly mixing Markov chains," *Inf. Comput.*, vol. 82, no. 1, pp. 93–133, 1989.

[50] B. Mohar, "Isoperimetric numbers of graphs," *J. Combinatorial Theory, B*, vol. 47, no. 3, pp. 274–291, 1989.

[51] M. Mihail, "Conductance and convergence of Markov chains—A combinatorial treatment of expanders," in *Proc. 30th Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 1989, pp. 526–531.

[52] D. S. Hochbaum, "A polynomial time algorithm for rayleigh ratio on discrete variables: Replacing spectral techniques for expander ratio, normalized cut, and Cheeger constant," *Oper. Res.*, vol. 61, no. 1, pp. 184–198, 2013.

[53] D. S. Hochbaum, C. Lu, and E. Bertelli, "Evaluating performance of image segmentation criteria and techniques," *EURO J. Comput. Optim.*, vol. 1, nos. 1–2, pp. 155–180, 2013.

[54] M. M. Deza and M. Laurent, *Geometry Cuts Metrics*. New York, NY, USA: Springer-Verlag, 1997.

[55] R. Andersen and K. J. Lang, "An algorithm for improving graph partitions," in *Proc. 19th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2008, pp. 651–660.

[56] L. Orecchia and Z. A. Zhu, "Flow-based algorithms for local graph clustering," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2014, pp. 1267–1286.

[57] K. Lang and S. Rao, "A flow-based method for improving the expansion or conductance of graph cuts," in *Proc. 10th Int. IPCO Conf. Integer Program. Combinatorial Optim.*, 2004, pp. 325–337.

[58] F. Chung, "Random walks and local cuts in graphs," *Linear Algebra Appl.*, vol. 423, pp. 22–32, May 2007.

[59] A. V. Goldberg and R. E. Tarjan, "Efficient maximum flow algorithms," *Commun. ACM*, vol. 57, no. 8, pp. 82–89, 2014.

[60] N. Ailon and E. Liberty, "Fast dimension reduction using Rademacher series on dual BCH codes," in *Proc. 19th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2008, pp. 1–9.

[61] B. Cherkassky and A. Goldberg, "On implementing push-relabel method for the maximum flow problem," in *Proc. 4th Int. IPCO Conf. Integer Program. Combinatorial Optim.*, 1995, pp. 157–171.

[62] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1124–1137, Sep. 2004.

[63] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in a network with power estimation," *Soviet Math. Docladi*, vol. 11, pp. 1277–1280, 1970.

[64] D. S. Hochbaum, "Polynomial time algorithms for ratio regions and a variant of normalized cut," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 889–898, May 2010.

[65] T. J. Hansen and M. W. Mahoney, "Semi-supervised eigenvectors for large-scale locally-biased learning," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3691–3734, 2014.

[66] D. Lawlor, T. Budavari, and M. W. Mahoney, "Mapping the similarities of spectra: Global and locally-biased approaches to SDSS galaxy data," 2016. [Online]. Available: https://arxiv.org/abs/1609.03932

[67] N. K. Vishnoi, "$Lx = b$ Laplacian solvers and their algorithmic applications," *Theor. Comput. Sci.*, vol. 8, nos. 1–2, pp. 1–141, 2012.

[68] D. F. Gleich and M. W. Mahoney, "Using local spectral methods to robustify graph-based learning algorithms," in *Proc. 21st Annu. ACM SIGKDD Conf.*, 2015, pp. 359–368.

[69] A. L. Traud, P. J. Mucha, and M. A. Porter, "Social structure of Facebook networks," *Phys. A, Stat. Mech. Appl.*, vol. 391, no. 16, pp. 4165–4180, 2012.

[70] Y. Hu, "Efficient and high quality force-directed graph," *Math. J.*, vol. 10, no. 1, pp. 37–71, 2005.

[71] Y. X. Wang, J. Sharpnack, A. Smola, and R. Tibshirani, "Trend filtering on graphs," in *Proc. 18th Int. Conf. Artif. Intell. Statist.*, Oct. 2014, pp. 1042–1050.

[72] Z. A. Zhu, S. Lattanzi, and V. Mirrokni, "A local algorithm for finding well-connected clusters," in *Proc. 30th Int. Conf. Mach. Learn.*, Dec. 2013, pp. 396–404.

[73] R. Andersen and Y. Peres, "Finding sparse cuts locally using evolving sets," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 235–244.

[74] F. Chung, "A local graph partitioning algorithm using heat kernel pagerank," *Internet Math.*, vol. 6, no. 3, pp. 315–330, 2009.

[75] F. Chung and O. Simpson, "Computing heat kernel pagerank and a local clustering algorithm," in *Proc. 25th Int. Workshop IWOCA*, May 2014, pp. 110–121.

[76] R. Khandekar, S. Khot, L. Orecchia, and N. Vishnoi, "On a cut-matching game for the sparsest cut problem," Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2007-177, Dec. 2007.

[77] L. Orecchia, L. Schulman, U. Vazirani, and N. Vishnoi, "On partitioning graphs via single commodity flows," in *Proc. 40th Annu. ACM Symp. Theory Comput.*, 2008, pp. 461–470.

ABOUT THE AUTHORS

Kimon Fountoulakis received the B.Sc. degree from Athens University of Economics and Business, Athens, Greece and the M.Sc. and Ph.D. degrees from University of Edinburgh, Edinburgh, U.K.

He is a Postdoctoral Fellow of the International Computer Science Institute and the Department of Statistics, University of California Berkeley, Berkeley, CA, USA. His research is on numerical optimization, network and graph algorithms, and parallel and distributed computing.

David F. Gleich received the B.Sc. degree from Harvey Mudd College, Claremont, CA, USA and the Ph.D. degree from Stanford University, Stanford, CA, USA.

He is an Assistant Professor of Computer Science at Purdue University, West Lafayette, IN, USA. His research is on matrix computations, network and graph algorithms, and parallel and distributed computing.

Prof. Gleich has been awarded a Microsoft Research Graduate fellowship, the John von Neumann postdoctoral fellowship, an NSF CAREER award, and a Sloan Research Fellowship.

Michael W. Mahoney received the Ph.D. degree from Yale University, New Haven, CT, USA, with a dissertation in computational statistical mechanics.

He is currently with the Department of Statistics and at the International Computer Science Institute, University of California Berkeley, Berkeley, CA, USA. He works on algorithmic and statistical aspects of modern large-scale data analysis. Much of his recent research has focused on large-scale machine learning, including randomized matrix algorithms and randomized numerical linear algebra, geometric network analysis tools for structure extraction in large informatics graphs, scalable implicit regularization methods, and applications in genetics, astronomy, medical imaging, social network analysis, and internet data analysis. He has worked and taught at the Mathematics Department, Yale University; at Yahoo Research; and at the Mathematics Department, Stanford University. Among other things, he is on the national advisory committee of the Statistical and Applied Mathematical Sciences Institute (SAMSI), he was on the National Research Council's Committee on the Analysis of Massive Data, he runs the biennial MMDS Workshops on Algorithms for Modern Massive Data Sets, and he spent fall 2013 at UC Berkeley co-organizing the Simons Foundation's program on the Theoretical Foundations of Big Data Analysis.