Eigenvector Computation and Community Detection in Asynchronous Gossip Models

Frederik Mallmann-Trenn
CSAIL, MIT, US
mallmann@mit.edu

Cameron Musco
CSAIL, MIT, US
cnmusco@mit.edu

Christopher Musco
CSAIL, MIT, US
cpmusco@mit.edu

Abstract

We give a simple distributed algorithm for computing adjacency matrix eigenvectors for the communication graph in an asynchronous gossip model. We show how to use this algorithm to give state-of-the-art asynchronous community detection algorithms when the communication graph is drawn from the well-studied *stochastic block model*. Our methods also apply to a natural alternative model of randomized communication, where nodes within a community communicate more frequently than nodes in different communities.

Our analysis simplifies and generalizes prior work by forging a connection between asynchronous eigenvector computation and Oja's algorithm for streaming principal component analysis. We hope that our work serves as a starting point for building further connections between the analysis of stochastic iterative methods, like Oja's algorithm, and work on asynchronous and gossip-type algorithms for distributed computation.

2012 ACM Subject Classification C.2.4 Distributed Systems

Keywords and phrases block model, community detection, distributed clustering, eigenvector computation, gossip algorithms, population protocols

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.401

Related Version See `https://arxiv.org/abs/1804.08548` for the full version.

Funding This work was supported in part by NSF Award Numbers BIO-1455983, CCF-1461559, CCF-0939370, and CCF-1565235. Cameron Musco was partially supported by an NSF graduate student fellowship.

1 Introduction

Motivated by the desire to process and analyze increasingly large networks—in particular social networks—considerable research has focused on finding efficient distributed protocols for problems like triangle counting, community detection, PageRank computation, and node centrality estimation. Many of the most popular systems for massive-scale graph processing, including Google's Pregel [23] and Apache Giraph [33] (used by Facebook), employ programming models based on the simulation of distributed message passing algorithms, in which each node is viewed as a processor that can send messages to its neighbors.

Apart from computational benefits, distributed graph processing can also be required when privacy constraints apply: for example, EU regulations restrict the personal data sent to countries outside of the EU [9]. Distributed algorithms avoid possibly problematic aggregation of network information, allowing each node to compute a local output based on their own neighborhood and messages received from their neighbors.

One of the main problems of interest in network analysis is the computation of the eigenvectors of a networks' adjacency matrix (or related incidence matrices, such as the graph Laplacian). The extremal eigenvectors have many important applications—from graph partitioning and community detection [15, 26], to embedding in graph-based machine learning [5, 29], to measuring node centrality and computing importance scores like PageRank [6].

Due to their importance, there has been significant work on distributed eigenvector approximation. In *synchronous* message passing systems, it is possible to simulate the well-known power method for iterative eigenvector approximation [21]. However, this algorithm requires that each node communicates synchronously with all of its neighbors in each round .

In an attempt to relax this requirement, models in which a subset of neighbors are sampled in each communication round [22] have been studied. However, the computation of graph eigenvectors in fully asynchronous and gossip-based message passing systems, in which nodes communicate with a single neighbor at a time in an asynchronous fashion, is not well-understood. While a number of algorithms have been proposed, which give convergence to the true eigenvectors as the number of iterations goes to infinity, strong finite iteration approximation bounds are not known [16, 27].

Our contributions.

In this work, we give state-of-the-art algorithms for graph eigenvector computation in asynchronous systems with randomized schedulers, including the classic gossip model [8, 14] and population protocol model [2]. We show that in these models, communication graph eigenvectors can be computed via a very simple adaption of Oja's classic iterative algorithm for principal components analysis [30]. Our analysis leverages recent work studing Oja's algorithm for streaming covariance matrix eigenvector estimation [1, 20].

By making an explicit connection between work on streaming eigenvector estimation and asynchronous computation, we hope to generally expand the toolkit of techniques that can be applied to analyzing graph algorithms in asynchronous systems.

As a motivating application, we use our results to give state-of-the-art distributed community detection protocols, significantly improving upon prior work for the well-studied stochastic-block model and related models where nodes communicate more frequently within their community than outside of it. We summarize our results below.

Asynchronous eigenvector computation. First, we provide an algorithm (Algorithm 2) that approximates the k largest eigenvectors $\mathbf{v}_1, ..., \mathbf{v}_k$ for an arbitrary communication matrix (essentially a normalized adjacency matrix, defined formally in Definition 1).

For an n-node network, the algorithm ensures, with good probability, that each node $u \in [n]$ computes the u^{th} entries of vectors $\tilde{\mathbf{v}}_1, ..., \tilde{\mathbf{v}}_k$ such that for all $i \in [k]$, $\|\tilde{\mathbf{v}}_i - \mathbf{v}_i\|_2^2 \leq \epsilon$. Each message sent by the algorithm requires communicating just $O(k)$ numbers, and the global time complexity is $\tilde{O}(\frac{\Lambda k^3}{\text{gap} \cdot \min(\text{gap}, \gamma_{mix})\epsilon^3})$ local rounds, where gap is the minimal gap between the k largest eigenvalues, γ_{mix} is roughly speaking the spectral gap, i.e., the difference between the largest and second-largest eigenvalue, and Λ is the sum of the k largest eigenvalues. We note that we use $\tilde{O}(\cdot)$ to suppress logarithmic terms, and in particular, factors of poly $\log n$. See Theorem 6 for a more precise statement.

For illustration, consider a communication graph generated via the stochastic block model – $G(n, p, q)$, which has n nodes, partitioned into two equal-sized clusters. Each intracluster edge added independently with probability p and each intercluster edge is added with probability $q < p$. If, for example, $p = \Omega\left(\frac{\log n}{n}\right)$ and $q = p/2$, and $k = 2$, we can bound with high probability $\Lambda = \Theta(1/n)$, gap $= \Theta(1/n)$, and $\gamma_{mix} = \Theta(1/n)$, which yields an eigenvector approximation algorithm running in $\tilde{O}(\frac{n}{\epsilon^3})$ global rounds, or $\tilde{O}(\frac{1}{\epsilon^3})$ local rounds.

Approximate community detection. Second, we harness our eigenvector approximation routine for community detection in the stochastic block model with connection probabilities p, q (we give two natural definitions of this model in an asynchronous distributed system with a random scheduler; see Definitions 2 and 3). After executing our protocol (Algorithm 5), with good probability, all but an ϵ fraction of the nodes output a correct community label in $\tilde{O}\left(1/\epsilon^3\rho^2\right)$ local rounds, where $\rho = \min\left(\frac{q}{p+q}, \frac{p-q}{p+q}\right)$. For example, when $q = p/2$, this complexity is $\tilde{O}\left(1/\epsilon^3\right)$. See Theorem 8 and Theorem 9 for precise bounds.

Exact community detection. Finally, we show how to produce an exact community labeling, via a simple gossip-based error correction scheme. For ease of presentation, here we just state our results in the case when $q = p/2$ and we refer to section 5 (Theorems 10 and 11) for general results. Starting from an approximate labeling in which only a small constant fraction of the nodes are incorrectly labeled, we show that, with high probability, after $O(\log n)$ local rounds, all nodes are labeled correctly.

Related work.

Community detection via graph eigenvector computation and other spectral methods has received ample attention in centralized setting [25, 10, 38]. Such methods are known to recover communities in the stochastic block model close to the information theoretic limit. Interestingly, many state-of-the-art community detection algorithms in this model, which improve upon spectral techniques, are based on message passing (belief propagation) algorithms [12, 28]. However, these algorithms are not known to work in asynchronous contexts.

Community detection in asynchronous distributed systems has received less attention. It has recently been tackled in a beautiful paper by Becchetti et al. [3]. The algorithm studied in this paper is a very simple averaging protocol, originally considered by the authors in a synchronous setting [4]. Each node starts with a random value chosen uniformly in $\{-1, 1\}$. Each time two nodes communicate, they update their values to the average of their previous values. After each round of communication, a node's estimated community is given by the *sign* of the change of its value due to the averaging update in that round.

As discussed in [3], in *regular* graphs, which their analysis is restricted to, this protocol can be viewed as a method for estimating the sign of the second largest adjacency matrix eigenvector. Thus, it has close connections with our protocols, which explicitly estimate this eigenvector and form community labels using the signs of its entries.

The analysis of Becchetti et al. applies to *regular clustered graphs*, under certain eigenvalue gap restrictions. For comparison, we focus on *regular stochastic block model graphs*, in which all nodes have exactly a edges to (randomly selected) nodes in their cluster and exactly $b < a$ edges to nodes outside their cluster. We also focus on the case when the community detection algorithm succeeds with constant probability and a large constant fraction of nodes with correct cluster labels. It is possible to boost probability of success and/or the fraction of nodes with correct labels via the 'community sensitive labeling' approach of Becchetti et al. or our clean up phase, and by repeating the algorithm multiple times.

The results of Becchetti et al. apply with $O(\text{polylog}\, n)$ local rounds of communication when either $\frac{a}{b} = \Omega(\log^2 n)$, or when $a - b = \Omega(\sqrt{a+b})$. In contrast, our results for the (non-regular) stochastic block model give $O(\text{polylog}\, n)$ local runtime when $\frac{p}{q} = \Omega(1)$ or $n(p-q) = \Omega(\sqrt{n(p+q)\log n})$. Here we assume that q is not too small – see Theorem 9 for details.

Note that $n \cdot p$ and $n \cdot q$ can be compared to a and b, since they are the expected number of intra- and inter-cluster edges respectively. Thus, our results give comparable bounds, tightening those of Becchetti et al. in some regimes and holding in the most commonly studied family of stochastic block model graphs, without any assumption of regularity[1].

Outside of community detection, our approach to asynchronous eigenvector approximation is related to work on asynchronous distributed stochastic optimization [37, 11, 31]. Often, it is assumed that many processors update some decision variable in parallel. If these updates are sufficiently sparse, overwrites are rare and the algorithm converges as if it were run in a synchronous manner. Our implementation of Oja's algorithm falls under this paradigm. Each update to our eigenvector estimates is sparse – requiring a modification just by the two nodes that communicate at a given time. In this way, we can fully parallelize the algorithm, even in an asynchronous system.

2 Preliminaries

2.1 Notation

For integer $n > 0$, let $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$. Let $\mathbf{1}_{n,m}$ be an $n \times m$ all-ones matrix and $\mathbf{I}_{n\times n}$ be an $n \times n$ identity. Let $\mathbf{e}_i$ be the i^{th} standard basis vector, with length apparent from context. Let V denote a set of nodes with cardinality $|V| = n$. Let $\mathcal{P}$ be the set of all unordered node pairs (u, v) with $u \neq v$. $|\mathcal{P}| = \binom{n}{2}$.

For vector $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|_2$ is the Euclidean norm. For matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$, $\|\mathbf{M}\|_2 = \max_{\mathbf{x}} \frac{\|\mathbf{Mx}\|_2}{\|\mathbf{x}\|_2}$ is the spectral norm. $\|\mathbf{M}\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m \mathbf{M}_{i,j}^2}$ denotes the Frobenius norm. $\mathbf{M}^T$ is the matrix transpose of $\mathbf{M}$. When $\mathbf{M} \in \mathbb{R}^{n\times n}$ is symmetric we let $\lambda_1(\mathbf{M}) \geq \lambda_2(\mathbf{M}) \geq \ldots \geq \lambda_n(\mathbf{M})$ denote its eigenvalues. $\mathbf{M}$ is positive semidefinite (PSD) if $\lambda_i(\mathbf{M}) \geq 0$ for all i. For symmetric $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{n\times n}$ we use $\mathbf{M} \preceq \mathbf{N}$ to indicate that $\mathbf{N} - \mathbf{M}$ is PSD.

2.2 Computational model

We define an asynchronous distributed computation model that encompasses both the well-studied population protocol [2] and asynchronous gossip models [8]. Computation proceeds in rounds and a random scheduler chooses a single pair of nodes to communicate in each round. The choice is independent across rounds, but may be nonuniform across node pairs.

▶ **Definition 1** (Asynchronous communication model)**.** Let V be a set of nodes with $|V| = n$. Computation proceeds in rounds, with every node $v \in V$ having some state $s(v, t)$ in round t.

Recall that $\mathcal{P}$ denotes all unordered pairs of nodes in V. Let $w : \mathcal{P} \to \mathbb{R}^+$ be a nonnegative weight function. In each round, a random scheduler chooses exactly one $(u, v) \in \mathcal{P}$ with probability $w(u, v)/\left[\sum_{(i,j)\in\mathcal{P}} w(i, j)\right]$ and u, v both update their states according to some common (possibly randomized) transition function σ. Specifically, they set $s(v, t + 1) = \sigma(s(v, t), s(u, t))$ and $s(u, t + 1) = \sigma(s(u, t), s(v, t))$.

Note that in our analysis we often identify the weight function w with a symmetric weight matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ where $\mathbf{W}_{u,u} = 0$ and $\mathbf{W}_{u,v} = \mathbf{W}_{v,u} = w(u, v)/\left[\sum_{(i,j)\in\mathcal{P}} w(i, j)\right]$. Let $\mathbf{D}$ be a diagonal matrix with $\mathbf{D}_{u,u} = \sum_{v\in V} \mathbf{W}_{u,v}$. $\mathbf{D}_{u,u}$ is the probability that node u communicates in any given round. Since two nodes are chosen in each round, $\sum_u \mathbf{D}_{u,u} = 2$. We will refer to $\mathbf{D} + \mathbf{W}$ as the *communication matrix* of the communication model.

[1] We note that the analysis of Bechitti et al seems likely to extend to our alternative communication model (Definition 2), where the communication graph is weighted and regular

▶ Remark (Asynchronous algorithms). Since the transition function σ in Definition 1 is universal, nodes can be seen as identical processes, with no knowledge of w or unique ids. We do assume that nodes can initiate and terminate a protocol synchronously. That is, nodes interact from round 0 up to some round T, after which they cease to interact, or begin a new protocol. This assumption is satisfied if each node has knowledge of the global round number but, in general, is much weaker. For example, in the asynchronous gossip model discussed below, it is sufficient for nodes to have access to a synchronized clock.

We use *algorithm* to refer to a sequence of transition functions, each corresponding to a subroutine run for specified number of rounds. Subroutines are run sequentially. The first has input nodes with identical starting states (as prescribed by Definition 1) but later subroutines start once nodes have updated their states and thus have distinguished inputs.

▶ Remark (Simulation of existing models). The standard *population protocol model* [2] is recovered from Definition 1 by setting $w(u,v) = 1$ for all (u,v) – i.e., pairs of nodes communicate uniformly at random. A similar model over a fixed communication graph $G = (E,V)$ is recovered by setting $w(u,v) = 1$ for all $(u,v) \in E$ and $w(u,v) = 0$ for $(u,v) \notin E$.

Definition 1 also encompasses the *asynchronous gossip model* [8, 14], where each node holds an independent Poisson clock and contacts a random neighbor when the clock ticks. If we identify rounds with clock ticks, let λ_u be the rate of node u's clock, and let $p(u,v)$ be the probability that u contacts v when its clock ticks. Then the probability that nodes u and v interact in a given round is $\frac{1}{2}\left[\frac{\lambda_u}{\sum_{z\in V}\lambda_z}\cdot p(u,v) + \frac{\lambda_v}{\sum_{z\in V}\lambda_z}\cdot p(v,u)\right]$. With $w(u,v)$ set to this value, Definition 1 corresponds exactly to the asynchronous gossip model.

2.3 Distributed community detection problem

This paper studies the very general problem of computing communication matrix eigenvectors with asynchronous protocols run by the nodes in V. One primary application of computing eigenvectors is to detect community structure in G. Below we formalize this application as the *distributed community detection problem* and introduce two specific cases of interest.

In the distributed community detection problem, the weight function w and corresponding weight matrix $\mathbf{W}$ of Definition 1 are clustered: nodes in the same cluster are more likely to communicate than nodes in different clusters. The goal is for each node to independently identify what cluster it belongs to (up to a permutation of the cluster labels).

We consider two models of clustering. In the first (n,p,q)*-weighted communication model*, the weight function directly reflects the increased likelihood of intracluster communication. In the second, $G(n,p,q)$*-communication model*, weights are uniform on a graph sampled from the well-studied planted-partition or *stochastic block model* [19]. For simplicity, we focus on the setting in which there are two equal sized clusters, but believe that our techniques can be extended to handle a larger number of clusters, potentially with unbalanced sizes.

▶ **Definition 2** ((n,p,q)-weighted communication model)**.** An asynchronous model (Definition 1), where node set V is partitioned into disjoint sets V_1, V_2 with $|V_1| = |V_2| = n/2$. For values $q < p$, $w(u,v) = p$ if $u,v \in V_i$ for some i and $w(u,v) = q$ if $u \in V_i$ and $v \in V_j$ for $i \neq j$.

▶ **Definition 3** ($G(n,p,q)$-communication model)**.** An asynchronous model (Definition 1), where node set V is partitioned into disjoint sets V_1, V_2 with $|V_1| = |V_2| = n/2$. The weight matrix $\mathbf{W}$ is a normalized adjacency matrix of a random graph $G(V,E)$ generated as follows: for each pair of nodes $u, v \in V$, add edge (u,v) to edge set E with probability p if u and v are in the same partition V_i and probability $q < p$ if u and v are in different partitions.

Analysis of community detection in the (n, p, q)-weighted communication model is more elegant, and will form the basis of our analysis for the $G(n, p, q)$-communication model, which more closely matches models considered in prior work on in both distributed and centralized settings. Formally, we define the distributed community detection problem as follows:

▶ **Definition 4** (Distributed community detection problem). An algorithm executing in the communication models of Definition 2 and Definition 3 solves community detection in T rounds if for every $t \geq T$, all nodes in V_1 hold some integer state $s_1 \in \{-1, 1\}$, while all nodes in V_2 hold state $s_2 = -s_1$. An algorithm solves the community detection problem in L *local rounds* if every node's state remains fixed after L local interactions with other nodes.

3 Asynchronous Oja's algorithm

Our main contribution is a distributed algorithm for computing eigenvectors of the communication matrix $\mathbf{D}+\mathbf{W}$. These eigenvectors can be used to solve the distributed community detection problem or in other applications. Our main algorithm is a distributed, asynchronous adaptation of Oja's classic iterative eigenvector algorithm [30], described below:

Algorithm 1 OJA'S METHOD (CENTRALIZED)

Input: $\mathbf{x}_0, ..., \mathbf{x}_{T-1} \in \mathbb{R}^n$ drawn i.i.d. from some distribution $\mathcal{D}$ such that for some constant C, $\mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[\|\mathbf{x}\|_2^2 \leq C] = 1$ and $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x}\mathbf{x}^T] = \mathbf{M}$. Rank parameter k and step size η.
Output: Orthonormal $\tilde{\mathbf{V}} \in \mathbb{R}^{n\times k}$ whose columns approximate $\mathbf{M}$'s k top eigenvectors.

1: Choose $\mathbf{Q}_0$ with entries drawn i.i.d. from the standard normal distribution $\mathcal{N}(0, 1)$.
2: **for** $t = 0, ..., T-1$ **do**
3: $\quad\mathbf{Q}_{t+1} := (\mathbf{I} + \eta\mathbf{x}_t\mathbf{x}_t^T)\mathbf{Q}_t$.
4: **end for**
5: **return** $\tilde{\mathbf{V}}_T := \text{orth}(\mathbf{Q}_T)$. ▷ Orthonormalizes the columns of $\mathbf{Q}_T$.

3.1 Approximation bounds for Oja's method

A number of recent papers have provided strong convergence bounds for the *centralized* version of Oja's method [1, 20]. We will rely on the following theorem, which we prove in full verison using a straightforward application of the arguments in [1].

▶ **Theorem 5.** *Let $\mathbf{M} \in \mathbb{R}^{n\times n}$ be a PSD matrix with $\frac{\sum_{i=1}^k \lambda_i(\mathbf{M})}{C} \leq \Lambda$ and $\frac{\lambda_k(\mathbf{M})-\lambda_{k+1}(\mathbf{M})}{C} \geq \text{gap}$ for some values Λ, gap. For any $\epsilon, \delta \in (0, 1)$, let $\xi = \frac{n}{\delta\epsilon\cdot\text{gap}}$, $\eta = \frac{c_1\epsilon^2\cdot\text{gap}\cdot\delta^2}{C\Lambda k\log^3\xi}$ for some sufficiently small constant c_1, and $T = \frac{c_2\cdot(\log\xi+1/\epsilon)}{C\cdot\text{gap}\cdot\eta}$ for sufficiently large c_2. Then with probability $\geq 1-\delta$, Algorithm 1 run with step size η returns $\tilde{\mathbf{V}}_T$ satisfying,*

$$\|\mathbf{Z}^T\tilde{\mathbf{V}}_T\|_F^2 \leq \epsilon.$$

where $\mathbf{Z}$ is an orthonormal basis for the bottom $n-k$ eigenvectors of $\mathbf{M}$.

If $\tilde{\mathbf{V}}_T$ exactly spanned $\mathbf{M}$'s top k eigenvectors, $\|\mathbf{Z}^T\tilde{\mathbf{V}}_T\|_F^2$ would equal 0. To obtain an approximation of ϵ, the number of iterations required by Oja's method naturally depends inversely on ϵ, the failure probability δ, and the gap between eigenvalues $\lambda_k(\mathbf{M})$ and $\lambda_{k+1}(\mathbf{M})$.

3.2 Distributed Oja's method via random edge sampling

Oja's method can be implemented in the asynchronous communication model (Definition 1) to compute top eigenvectors of the communication matrix $\mathbf{D}+\mathbf{W}$, defined in subsection 2.2.

For any pair of nodes (u, v), let $\mathbf{e}_{u,v} = \mathbf{e}_u + \mathbf{e}_v$ be the vector with all zero entries except 1's in its u^{th} and v^{th} positions. Given weight function w and associated matrix $\mathbf{W}$, let $\mathcal{D}_{\mathbf{W}}$ be the distribution in which each $\mathbf{e}_{u,v}$ is selected with probability $\mathbf{W}_{u,v}$. That is, the same distribution by which edges are selected to be active by the scheduler in Definition 1. Noting that $\mathbf{e}_{u,v}\mathbf{e}_{u,v}^T$ is all zero except at its (u, u), (v, v), (u, v), and (v, u) entries, we can see that

$$\mathop{\mathbb{E}}_{\mathbf{e}_{u,v}\sim\mathcal{D}_{\mathbf{W}}}\left[\mathbf{e}_{u,v}\mathbf{e}_{u,v}^T\right] = \sum_{(u,v)\in\mathcal{P}} \mathbf{W}_{u,v} \cdot \mathbf{e}_{u,v}\mathbf{e}_{u,v}^T = \mathbf{D} + \mathbf{W}, \tag{1}$$

where $\mathcal{P}$ denotes the set of unordered node pairs (u, v) with $u \neq v$. So if we run Oja's algorithm with $\mathbf{e}_{u,v}$ sampled according to $\mathcal{D}_{\mathbf{W}}$, we will obtain an approximation to the top eigenvectors of $\mathbf{D}+\mathbf{W}$. Note that this matrix is PSD, by the fact that each $\mathbf{e}_{u,v}\mathbf{e}_{u,v}^T$ is PSD.

Furthermore, the algorithm can be implemented in our communication model as an *extremely simple averaging protocol.* Each iteration of Algorithm 1 requires computing $\mathbf{Q}_{t+1} = (\mathbf{I} + \eta\mathbf{x}_t\mathbf{x}_t^T)\mathbf{Q}_t$. If $\mathbf{x}_t = \mathbf{e}_{u,v}$ for $\mathbf{e}_{u,v} \sim \mathcal{D}_{\mathbf{W}}$, we can see that computing $\mathbf{Q}_{t+1}$ just requires updating the u^{th} and v^{th} rows of $\mathbf{Q}_t$. Thus, if the n rows of $\mathbf{Q}_t$ are distributed across n nodes, this update can be done locally by nodes u and v when they are chosen to interact by the randomized scheduler. Specifically, letting $[q_u^{(1)}, ..., q_u^{(k)}]$ be the u^{th} row of $\mathbf{Q}_t$, stored as the state at node u, applying $(\mathbf{I}+\eta\mathbf{e}_{u,v}\mathbf{e}_{u,v}^T)$ just requires setting for all $i \in [k]$:

$$q_u^{(i)} := (1 + \eta)q_u^{(i)} + \eta q_v^{(i)}. \tag{2}$$

Node v makes a symmetric update, and all other entries of $\mathbf{Q}_t$ remain fixed.

We give the pseudocode for this protocol in Algorithm 2. Along with the main iteration based on the simple update in (2), the nodes need to implement Step 5 of Algorithm 1, where $\mathbf{Q}_T$ is orthogonalized. This can be done with a gossip-based protocol, which we abstract as the routine `AsynchOrth`. We give an implementation of `AsynchOrth` in subsection 3.3.

▶ Remark (Choice of communication matrix). While, as we will show, the eigenvectors of $\mathbf{D}+\mathbf{W}$ are naturally useful in our applications to community detection, the above techniques easily extend to computing eigenvectors of other matrices. For example, if we set $\mathbf{e}_{u,v} = \mathbf{e}_u - \mathbf{e}_v$, $\mathbb{E}_{\mathbf{e}_{u,v}\sim\mathcal{D}_{\mathbf{W}}}[\mathbf{e}_{u,v}\mathbf{e}_{u,v}^T] = \mathbf{D} - \mathbf{W} = \mathbf{L}$, a scaled Laplacian of the communication graph.

Algorithm 2 Asynchronous Oja's (`AsynchOja`(T, T', η))

Input: Time bounds T, T', step size η.
Initialization: $\forall u$, chose $[q_u^{(1)}, ..., q_u^{(k)}]$ independently from standard Gaussian $\mathcal{N}(0, 1)$.

if $t < T$ **then**
  (u, v) is chosen by the randomized scheduler.
  For all $i \in [k]$, $q_u^{(i)} := (1 + \eta)q_u^{(i)} + \eta q_v^{(i)}$. ▷ Computes of $(\mathbf{I} + \eta\mathbf{e}_{u,v}\mathbf{e}_{u,v}^T)\mathbf{Q}_t$.
else
  $[\hat{v}_u^{(1)}, ..., \hat{v}_u^{(k)}] =$ `AsynchOrth`$([q_u^{(1)}, ..., q_u^{(k)}], T')$. ▷ Implements of $\tilde{\mathbf{V}}_T = \mathrm{orth}(\mathbf{Q}_T)$.
end if

Note that in the pseudocode above, when nodes u, v interact in the asynchronous model, they only need to share their respective values of $q_u^{(i)}$ and $q_v^{(i)}$ for $i \in [k]$.

Up to the orthogonalization step, we see that Algorithm 2 *exactly simulates* Algorithm 1 on input $\mathbf{M} = \mathbf{D} + \mathbf{W}$. Thus, assuming that `AsynchOrth`$([q_u^{(1)}, ..., q_u^{(k)}])$ exactly computes $\tilde{\mathbf{V}}_T = \mathrm{orth}(\mathbf{Q}_T)$ as in Step 5 of Algorithm 1, the error bound of Theorem 5 applies directly. Specifically, if we let the local states, $[q_1^{(1)}, \ldots, q_n^{(1)}], \ldots, [q_1^{(k)}, \ldots, q_n^{(k)}]$ correspond to the k length-n vectors in $\tilde{\mathbf{V}}_T$, Theorem 5 shows that $\|\mathbf{Z}^T\tilde{\mathbf{V}}_T\|_F^2 \leq \epsilon$. In subsection 3.3 we show that this bound still holds when `AsynchOrth` computes an approximate orthogonalization.

3.3 Distributed orthogonalization and eigenvector guarantees

In fact, a specific orthogonalization strategy yields a stronger bound, which is desirable in many applications, including community detection: Algorithm 2 can actually well approximate *each* of $\mathbf{D}+\mathbf{W}$'s top k eigenvectors, instead of just the subspace they span.

Specifically, let $\tilde{\mathbf{v}}_i$ denote the i^{th} column of $\tilde{\mathbf{V}}_T$ and $\mathbf{v}_i$ denote the i^{th} eigenvector of $\mathbf{D}+\mathbf{W}$. We want $(\tilde{\mathbf{v}}_i^T\mathbf{v}_i)^2 \geq 1-\epsilon$ for all i. Such a guarantee requires sufficiently large gaps between the top k eigenvalues, so that their corresponding eigenvectors are identifiable. If these gaps exist, the guarantee can by using the following orthogonalization procedure:

Algorithm 3 ORTHOGONALIZATION VIA CHOLESKY FACTORIZATION (CENTRALIZED)

Input: $\mathbf{Q} \in \mathbb{R}^{n\times k}$ with full column rank. **Output**: Orthonormal span for $\mathbf{Q}$, $\tilde{\mathbf{V}} \in \mathbb{R}^{n\times k}$.

$\mathbf{L} := \text{chol}(\mathbf{Q}^T\mathbf{Q})$ ▷ Cholesky decomp. returns lower triangular $\mathbf{L}$ with $\mathbf{L}\mathbf{L}^T = \mathbf{Q}^T\mathbf{Q}$.
return $\tilde{\mathbf{V}} := \mathbf{Q}(\mathbf{L}^T)^{-1}$ ▷ Orthonormalize $\mathbf{Q}_T$'s columns using the Cholesky factor.

▶ Remark. Algorithm 3 requires an input that is *full-rank*, which always includes $\mathbf{Q}_T$ in Algorithms 1 and 2: $\mathbf{Q}_0$'s entries are random Gaussians so it is full-rank with probability 1 and each $(\mathbf{I}+\eta\mathbf{x}_t^T\mathbf{x}_t)$ is full-rank since $\eta < \|\mathbf{x}_t\|$. Thus, $\mathbf{Q}_T = \prod_{t=0}^{T-1}(\mathbf{I}+\eta\mathbf{x}_t^T\mathbf{x}_t)\mathbf{Q}_0$ is too.

Ultimately, our `AsynchOrth` is an asynchronous distributed implementation of Algorithm 3. We first prove an eigenvector approximation bound under the assumption that this implementation is exact and then adapt that result to account for the fact that `AsynchOrth` only outputs an approximate solution.

Pseudocode for `AsynchOrth` is included below. Each node first computes a (scaled) approximation to every entry of $\mathbf{Q}^T\mathbf{Q}$ using a simple averaging technique. Nodes then locally compute $\mathbf{L} = \text{chol}\left(\mathbf{Q}^T\mathbf{Q}\right)$ and the u^{th} row of $\tilde{\mathbf{V}}_T = \mathbf{Q}(\mathbf{L}^T)^{-1}$. In the full version we argue that, due to numerical stability of Cholesky decomposition, each node's output is close to the u^{th} row of an exactly computed $\tilde{\mathbf{V}}_T$, despite the error in constructing $\mathbf{Q}^T\mathbf{Q}$.

Algorithm 4 ASYNCHRONOUS CHOLESKY ORTHOGONALIZATION (`AsynchOrth`(T))

Input: Time bound T.
Initialization: Each node holds $[q_u^{(1)},\ldots,q_u^{(k)}]$. For all $i,j\in[k]$, let $r_u^{(i,j)} := q_u^{(i)}\cdot q_u^{(j)}$.

if $t < T$ **then**
  (u,v) is chosen by the randomized scheduler.
  for all $i,j\in[k]$, $r_u^{(i,j)} := \frac{r_u^{(i,j)}+r_v^{(i,j)}}{2}$. ▷ Estimation of $\frac{1}{n}\mathbf{q}_i^T\mathbf{q}_j$ via averaging.
else
  Form $\mathbf{R}_u \in \mathbb{R}^{k\times k}$ with $(\mathbf{R}_u)_{i,j} = (\mathbf{R}_u)_{j,i} := n\cdot r_u^{(i,j)}$. ▷ Approximation of $\mathbf{Q}^T\mathbf{Q}$.
  $\mathbf{L}_u := \text{chol}(\mathbf{R}_u)$.
  $[\hat{v}_u^{(1)},\ldots,\hat{v}_u^{(k)}] := [q_u^{(1)},\ldots,q_u^{(k)}]\cdot(\mathbf{L}_u^T)^{-1}$. ▷ Approximation of u^{th} row of $\mathbf{Q}(\mathbf{L}_u^T)^{-1}$.
end if

In the full version we prove the following result when Algorithm 4 is used to implement `AsynchOrth` as a subroutine for Algorithm 2, `AsynchOja`(T,T',η):

▶ **Theorem 6** (Asynchronous eigenvector approximation). *Let* $\mathbf{v}_1,\ldots,\mathbf{v}_k$ *be the top* k

eigenvectors of the communication matrix $\mathbf{D}+\mathbf{W}$ *in an asynchronous communication model, and let* $\Lambda, \overline{\text{gap}}, \gamma_{mix}$ *be bounds satisfying:* $\Lambda \geq \sum_{j=1}^{k} \lambda_j(\mathbf{D}+\mathbf{W})$, $\overline{\text{gap}} \leq \min_{j\in[k]}[\lambda_j(\mathbf{D}+\mathbf{W}) - \lambda_{j+1}(\mathbf{D}+\mathbf{W})]$, *and* $\gamma_{mix} \leq \min\left[\frac{1}{n}, \log\left(\lambda_2^{-1}(\mathbf{I}-\frac{1}{2}\mathbf{D}+\frac{1}{2}\mathbf{W})\right)\right]$.

For any $\epsilon, \delta \in (0,1)$, *let* $\xi = \frac{n}{\delta\epsilon\cdot\overline{\text{gap}}}$. *Let* $\eta = \frac{c_1\epsilon^2\cdot\overline{\text{gap}}\cdot\delta^2}{\Lambda k^3\log^3\xi}$ *for sufficiently small* c_1, *and* $T = \frac{c_2\cdot(\log\xi+1/\epsilon)}{\overline{\text{gap}}\cdot\eta}$, $T' = \frac{c_3(\log\xi+1/\epsilon)\cdot\lambda_1(\mathbf{D}+\mathbf{W})}{\overline{\text{gap}}\cdot\gamma_{mix}}$ *for sufficiently large* c_2, c_3. *For all* $u\in[n], i\in[k]$, *let* $\hat{v}_u^{(j)}$ *be the local state computed by Algorithm 2. If* $\hat{\mathbf{V}} \in \mathbb{R}^{n\times k}$ *is given by* $(\hat{\mathbf{V}})_{u,j} = \hat{v}_u^{(j)}$ *and* $\hat{\mathbf{v}}_i$ *is the* i^{th} *column of* $\hat{\mathbf{V}}$, *then with probability* $\geq 1-\delta-e^{-\Theta(n)}$, *for all* $i\in[k]$:

$$\left|\hat{\mathbf{v}}_i^T\mathbf{v}_i\right| \geq 1-\epsilon \quad \textit{and} \quad \|\hat{\mathbf{v}}_i\|_2 \leq 1+\epsilon.$$

4 Distributed community detection

From the results of section 3, we obtain a simple population protocol for distributed community detection that works for many clustered communication models, including the (n,p,q)-weighted communication and $G(n,p,q)$-communication models of Definitions 2 and 3.

In particular, we show that if each node $u \in V$ can locally compute the u^{th} entry of an approximation $\hat{\mathbf{v}}_2$ to the second eigenvector of the communication matrix $\mathbf{D}+\mathbf{W}$, then it can solve the community detection problem locally: u just sets its state to the sign of this entry.

Algorithm 5 ASYNCHRONOUS COMMUNITY DETECTION (`AsynchCD`(T, T', η))

Input: Time bounds T, T', step size η.

1: Run `AsynchOja`(T, T', η) (Algorithm 2) with $k = 2$.
2: Set $\hat{\chi}_u := \text{sign}(\hat{v}_u^{(2)})$.

Here $\hat{\chi}_u \in \{-1, 1\}$ is the final state of node u. We will claim that this state solves the community detection problem of Definition 4. We use the notation $\hat{\chi}_u$ because we will use $\boldsymbol{\chi}$ to denote the true *cluster indicator vector* for communities V_1 and V_2 in a given communication model: $\boldsymbol{\chi}_u = 1$ for $u \in V_1$ and $\boldsymbol{\chi}_u = -1$ for $u \in V_2$.

In particular, we will show that if η is set so that `AsynchOja` outputs eigenvectors with accuracy ϵ, then a $1-O(\epsilon)$ fraction of nodes will correctly identify their clusters. In section 5 we show how to implement a 'cleanup phase' where, starting with ϵ set to a small constant (e.g. $\epsilon = .1$), the nodes can converge to a state with *all* cluster labels correct with high probability.

4.1 Community detection in the (n,p,q)-weighted communication model

We start with an analysis for the (n,p,q)-weighted communication model. Recall that in this model the nodes are partitioned into two sets, V_1 and V_2, each with $n/2$ elements. Without loss of generality we can identify the nodes with integer labels such that $1,\ldots,n/2 \in V_1$ and $n/2+1,\ldots,n \in V_2$. We define the weighted cluster indicator matrix, $\mathbf{C}^{(p,q)} \in \mathbb{R}^{n\times n}$:

$$\mathbf{C}^{(p,q)} \stackrel{\text{def}}{=} \begin{bmatrix} p\cdot\mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & q\cdot\mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \\ q\cdot\mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & p\cdot\mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \end{bmatrix}. \tag{3}$$

p and q can be arbitrary, but we will always take $p > q > 0$. It is easy to check that $\mathbf{C}^{(p,q)}$ is a rank two matrix with eigendecomposition:

$$\mathbf{C}^{(p,q)} = \frac{n}{2}\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix}\begin{bmatrix} p+q & 0 \\ 0 & p-q \end{bmatrix}\begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \end{bmatrix} \quad \text{where} \quad \mathbf{v}_1 = \frac{\mathbf{1}_{n\times 1}}{\sqrt{n}}, \quad \mathbf{v}_2 = \frac{\boldsymbol{\chi}}{\sqrt{n}}. \tag{4}$$

So, if all nodes could compute their corresponding entry in the *second eigenvector* of $\mathbf{C}^{(p,q)}$, then by simply returning the sign of this entry, they would solve the distributed community detection problem (Definition 4). If they compute this eigenvector approximately, then we can still show that a large fraction of them correctly solve community detection. Specifically:

▶ **Lemma 7.** *Let $\mathbf{v}_2$ be the second eigenvector of $\mathbf{C}^{(p,q)}$ for any $p > q > 0$. If $\tilde{\mathbf{v}}_2$ satisfies:*

$$\left|\tilde{\mathbf{v}}_2^T \mathbf{v}_2\right| \geq 1-\epsilon \quad \text{and} \quad \|\tilde{\mathbf{v}}_2\|_2 \leq 1+\epsilon. \tag{5}$$

for $\epsilon \leq 1$, then $\operatorname{sign}(\tilde{\mathbf{v}}_2)$ gives a labeling such that, after ignoring at most $5\epsilon n$ nodes, all remaining nodes in V_1 have the same labeling, and all in V_2 have the opposite.

A proof can be found in the full version. With Lemma 7 in place, we can then apply Theorem 6 to prove the correctness of `AsynchCD` (Algorithm 5) for the (n,p,q)-weighted communication model

▶ **Theorem 8** (ϵ-approximate community detection: (n,p,q)-weighted communication model)**.** *Consider Algorithm 5 in the (n,p,q)-weighted communication model. Let $\rho = \min\left(\frac{q}{p+q}, \frac{p-q}{p+q}\right)$. For sufficiently small constant c_1 and sufficiently large c_2 and c_3, let*

$$\eta = \frac{c_1 \epsilon^2 \delta^2 \rho}{\log^3\left(\frac{n}{\epsilon\delta\rho}\right)}, \quad T = \frac{c_2 n \left(\log^3\left(\frac{n}{\epsilon\delta\rho}\right) + \frac{\log\left(\frac{n}{\epsilon\delta\rho}\right)}{\epsilon}\right)}{\epsilon^2\delta^2\rho^2}, \quad T' = \frac{c_3 n\left(\log\left(\frac{n}{\epsilon\delta\rho}\right) + \frac{1}{\epsilon}\right)}{\rho^2}.$$

With probability $1-\delta$, after ignoring ϵn nodes, all remaining nodes in V_1 terminate in some state $s_1 \in \{-1,1\}$, and all nodes in V_2 terminate in state $s_2 = -s_1$. Suppressing polylogarithmic factors in the parameters, the total number of global rounds and local rounds required are: $T + T' = \tilde{O}\left(\frac{n}{\epsilon^3\delta^2\rho^2}\right)$ and $L = \tilde{O}\left(\frac{1}{\epsilon^3\delta^3\rho^2}\right)$.

Proof. In the (n,p,q)-weighted communication model the weight and degree matrices are:

$$\mathbf{W} = \frac{4}{n^2(p+q)-2np} \cdot (\mathbf{C}^{(p,q)} - p \cdot \mathbf{I}_{n\times n}) \quad \text{and} \quad \mathbf{D} = \frac{2}{n} \cdot \mathbf{I}_{n\times n}.$$

Thus, referring to the eigendecomposition of $\mathbf{C}^{(p,q)}$ shown in (4), the top eigenvector of $\mathbf{D} + \mathbf{W}$ is $\mathbf{v}_1 = \mathbf{1}_{n\times 1}/\sqrt{n}$ with corresponding eigenvalue: $\lambda_1 = \frac{4}{n^2(p+q)-2np} \cdot \left(\frac{n(p+q)}{2} - p\right) + \frac{2}{n} = \frac{4}{n}$. The second eigenvector is the scaled cluster indicator vector $\mathbf{v}_2 = \boldsymbol{\chi}/\sqrt{n}$ with eigenvalue

$$\lambda_2 = \frac{4}{n^2(p+q)-2np} \cdot \left(\frac{n(p-q)}{2} - p\right) + \frac{2}{n} = \frac{4}{n} \cdot \frac{p}{p + \frac{n}{n-2} \cdot q}.$$

Finally, for all remaining eigenvalues of $\mathbf{D} + \mathbf{W}$, $\{\lambda_3, ..., \lambda_n\}$, $\lambda_i = \frac{2}{n} - \frac{4p}{n^2(p+q)-2np}$. We can bound the eigenvalue gaps:

$$\lambda_1 - \lambda_2 \geq \frac{4}{n} - \frac{4}{n} \cdot \frac{p}{p+q} = \frac{4q}{n(p+q)} \qquad \lambda_2 - \lambda_3 = \frac{2(p-q)}{n(p+q)-2p} \geq \frac{2(p-q)}{n(p+q)}$$

Let $\rho = \min\left(\frac{q}{p+q}, \frac{p-q}{(p+q)}\right)$. We bound the mixing time of $\mathbf{W} + \mathbf{D}$ by noting that $\lambda_2(\mathbf{I} - 1/2\mathbf{D} + 1/2\mathbf{W}) \leq 1 - \frac{2q}{n(p+q)}$. Then using that $\log(1/x) \geq 1 - x$ for all $x \in (0,1]$, $\log(\lambda_2^{-1}(\mathbf{I} - 1/2\mathbf{D} + 1/2\mathbf{W}) \geq \frac{2q}{n(p+q)} \geq \frac{2\rho}{n}$. We then apply Theorem 6 with $k = 2$, $\Lambda = \frac{4}{n} + \frac{4}{n}\frac{p}{p+\frac{n}{n-2}q} \leq \frac{8}{n}$, $\overline{\text{gap}} = \frac{4}{n} \cdot \min\left(\frac{q}{p+q}, \frac{p-q}{2(p+q)}\right) \geq \frac{2\rho}{n}$, and $\gamma_{mix} = \frac{2\rho}{n}$. With these parameters we set, for sufficiently small c_1 and large c_2, c_3,

$$\eta = \frac{c_1 \epsilon^2 \delta^2 \cdot \rho}{\log^3\left(\frac{n}{\epsilon\delta\rho}\right)}, \quad T = \frac{c_2 \cdot n \cdot \left(\log^3\left(\frac{n}{\epsilon\delta\rho}\right) + \frac{\log\left(\frac{n}{\epsilon\delta\rho}\right)}{\epsilon}\right)}{\epsilon^2\delta^2\rho^2}, \quad T' = \frac{c_3 \cdot n \cdot \left(\log\left(\frac{n}{\epsilon\delta\rho}\right) + \frac{1}{\epsilon}\right)}{\rho^2}$$

where to bound T' we use that $\frac{\lambda_1(\mathbf{D}+\mathbf{W})}{\text{gap}} \leq \frac{2}{\rho}$. Let $\hat{\mathbf{V}} \in \mathbb{R}^{n\times k}$ be given by $(\hat{\mathbf{V}})_{u,j} = \hat{v}_u^{(j)}$ where $\hat{v}_u^{(j)}$ are the states of $\texttt{AsynchOja}(T, T', \eta)$ and let $\hat{\mathbf{v}}_2$ be the second column of $\hat{\mathbf{V}}$. With these parameters, Theorem 6 gives with probability $\geq 1-\delta$ that $\left|\hat{\mathbf{v}}_2^T \mathbf{v}_2\right| \geq 1-\epsilon$ and $\|\hat{\mathbf{v}}_2\|_2 \leq 1+\epsilon$.

Applying Lemma 7 then gives the theorem if we adjust ϵ by a factor of $1/5$. Recall that the second eigenvector of $\mathbf{D}+\mathbf{W}$ is identical to that of $\mathbf{C}^{(p,q)}$. Additionally, in expectation, each node is involved in $L = \frac{2(T+T')}{n}$ interactions. This bound holds for all nodes within a factor 2 with probability $1-\delta$ by a Chernoff bound, since $L = \Omega(\log(n/\delta))$. We can union bound over our two failure probabilities and adjust δ by $1/2$ to obtain overall failure probability $\leq \delta$. ◀

4.2 Community Detection in the $G(n, p, q)$-communication model

In the $G(n, p, q)$-communication model, nodes communicate using a random graph which is equal to the communication graph in the (n, p, q)-weighted communication model *in expectation*. Using an approach similar to [34], which is a simplifies the perturbation method used in [24], we can prove that in the $G(n, p, q)$-communication model $\mathbf{W}$ is a small perturbation of $\mathbf{C}^{(p,q)}$ and so the second eigenvector of $\mathbf{D}+\mathbf{W}$ approximates that of $\mathbf{C}^{(p,q)}$ – i.e., the cluster indicator vector $\boldsymbol{\chi}$. We defer this analysis to the full version, stating the main result here:

▶ **Theorem 9** (ϵ-approximate community detection: $G(n, p, q)$-communication model)**.** *Consider Algorithm 5 in the $G(n, p, q)$-communication model. Let $\rho = \min\left(\frac{q}{p+q}, \frac{p-q}{p+q}\right)$. For sufficiently small constant c_1 and sufficiently large c_2 and c_3 let*

$$\eta = \frac{c_1\epsilon^2\delta^2\rho}{\log^3\left(\frac{n}{\epsilon\delta\rho}\right)}, \quad T = \frac{c_2 n\left(\log^3\left(\frac{n}{\epsilon\delta\rho}\right) + \frac{\log\left(\frac{n}{\epsilon\delta\rho}\right)}{\epsilon}\right)}{\epsilon^2\delta^2\rho^2}, \quad T' = \frac{c_3 n\left(\log\left(\frac{n}{\epsilon\delta\rho}\right) + \frac{1}{\epsilon}\right)}{\rho^2}.$$

If $\frac{\min[q, p-q]}{\sqrt{p+q}} \geq \frac{c_4\sqrt{\log(n/\delta)}}{\epsilon\sqrt{n}}$ for large enough constant c_4, then, with probability $1-\delta$, after ignoring ϵn nodes, all remaining nodes in V_1 terminate in some state $s_1 \in \{-1, 1\}$, and all nodes in V_2 terminate in state $s_2 = -s_1$. Supressing polylogarithmic factors, the total number of global rounds and local rounds required are: $T + T' = \tilde{O}\left(\frac{n}{\epsilon^3\delta^2\rho^2}\right)$ and $L = \tilde{O}\left(\frac{1}{\epsilon^3\delta^3\rho^2}\right)$.

If for example, $p, q = \Theta(1)$ and thus the $G(n, p, q)$ graph is dense, we can recover the communities with probability $1-\delta$ up to $O(1)$ error as long as $q \leq p - c\sqrt{\log(n/\delta)/n}$ for sufficiently large constant c. Alternatively, if $p, q = \Theta(\log(n/\delta)/n)$, so the $G(n, p, q)$ graph is sparse, we require $q \leq cp$ for sufficiently small c.

5 Cleanup Phase

After we apply Theorem 9 (respectively, Theorem 8) an ϵ-fraction of nodes are incorrectly clustered. The goal of this section is to provide a simple algorithm that improves this clustering so that *all nodes* are labeled correctly after a small number of rounds.

For the (n, p, q)-weighted communication model, doing so is straightforward. After running Algorithm 2 and selecting a label, each time a node communicates in the future it records the chosen label of the node it communicates with. Ultimately, it changes its label to the majority of labels encountered. If ϵ is small enough so $p(1-\epsilon) > q + \epsilon p$, this majority tends towards the node's correct label. The number of required rounds for the majority to be correct, with good probability for all nodes, is a simple a function of p, q, and ϵ.

The $G(n, p, q)$-communication model is more difficult. Theorem 9 does not guarantee how incorrectly labeled nodes are distributed: it is possible that a majority of a node's neighbors fall into the set of ϵn "bad nodes". In that case, even after infinitely many rounds of communication, the majority label encountered will not tend towards the node's correct identity.

As a remedy, we introduce a phased algorithm (Algorithm 6) where each node updates its label to the majority of labels seen during a phase. We show that in each phase the fraction of incorrectly labeled nodes decreases by a constant factor. Our analysis establishes a graph theoretic bound on the external edge density of most subsets of nodes. Specifically, for all subsets S below a certain size, we show that, with high probability, there are at most $|S|/3$ nodes which have enough connections to S so that if an adversary gave all nodes in S incorrect labels, it could cause these nodes to have an incorrect majority label. This bound guarantees that at most $|S|/3$ bad labels 'propagate' to the next phase of the algorithm.

Algorithm 6 Cleanup phase (pseudocode for node u)

Input: Number of phases k and number of rounds per phase r.
Output: Label $\hat{\chi}_u \in \{-1, 1\}$

1: **for** Phase 1 to k **do**
2:     **for** Round $i = 1$ to r **do**
3:         $S_i := \hat{\chi}_v$, where $\hat{\chi}_v$ denotes the i^{th} sample of node u.
4:     **end for**
5:     $\hat{\chi}_u := 1$ if $\sum_i^r S_i \geq 0$, $\hat{\chi}_u := -1$ otherwise.
6: **end for**

▶ **Theorem 10.** *Consider the (n, p, q)-weighted communication model. Assume that a fraction of at most $\epsilon \leq 1/64$ of the nodes are incorrectly clustered after Algorithm 2. As long as $p' = (1-\epsilon)p$ and $q' = q + \epsilon p$ satisfy $p' > q'$, Algorithm 6 ensures that all nodes are correctly labeled with high probability after $O(\frac{p \ln n}{(\sqrt{p'} - \sqrt{q'})^2})$ local rounds. In particular, for $q \leq p/2$ and $\epsilon < 1/8$, the number of local rounds required is $O(\log n)$.*

▶ **Theorem 11.** *Consider the $G(n, p, q)$-communication model. Let $\Delta = \frac{p}{2} - \frac{q}{2} - \sqrt{12 p \ln n / n} - \sqrt{12 q \ln n / n}$. Assume that $\Delta = \Omega(\ln n / n)$ and at most $\epsilon \leq \Delta / 24p$ nodes are incorrectly clustered after Algorithm 2. As long as $p'' = \frac{p}{2} - \sqrt{\frac{6 p \ln n}{n}} - \frac{\Delta}{12}$ and $q'' = \frac{q}{2} + \sqrt{\frac{6 q \ln n}{n}} + \frac{\Delta}{12}$ satisfy $p'' > q''$, Algorithm 6 ensures that all nodes are correctly labeled with high probability after $O(\frac{p \ln^2 n}{(\sqrt{p''} - \sqrt{q''})^2})$ local rounds. In particular, for $q \leq p/2$ the number of local rounds required is $O(\log^2 n)$.*

Note that if $p - q = \Omega(\sqrt{\log n / n})$, then Δ simplifies to $\Delta = \Theta(p - q)$. Incidentally, $p - q = \Omega(\sqrt{\log n / n})$ is sometimes tight because, in this regime, clustering correctly can be infeasible: some nodes will simply have more neighbors in the opposite cluster. Consider for example when $p = 1/2 + \sqrt{\ln n / (10n)}$ and $q = 1/2$.

References

1 Zeyuan Allen-Zhu and Yuanzhi Li. First efficient convergence for streaming k-pca: a global, gap-free, and near-optimal rate. pages 487–492, 2017.

2 James Aspnes and Eric Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, pages 97–120. Springer, 2009.

3 Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, Prasad Raghavendra, and Luca Trevisan. Distributed asynchronous averaging for community detection. *arXiv:1703.05045*, 2017.

4 Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Find your place: Simple distributed algorithms for community detection. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 940–959, 2017.

5 Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

6 Phillip Bonacich. Some unique properties of eigenvector centrality. *Social networks*, 29(4):555–564, 2007.

7 Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Gossip algorithms: Design, analysis and applications. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1653–1664. IEEE, 2005.

8 Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking (TON)*, 14(SI):2508–2530, 2006.

9 Shakila Bu-Pasha. Cross-border issues under eu data protection law with regards to personal data protection. 26:1–16, 05 2017.

10 Sam Cole, Shmuel Friedland, and Lev Reyzin. A simple spectral algorithm for recovering planted partitions. *Special Matrices*, 5:139–157, 2017.

11 Christopher M De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Taming the wild: A unified analysis of hogwild-style algorithms. In *Advances in neural information processing systems*, pages 2674–2682, 2015.

12 Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

13 James W Demmel. *Applied numerical linear algebra*, volume 56. Siam, 1997.

14 Alexandros G Dimakis, Soummya Kar, José MF Moura, Michael G Rabbat, and Anna Scaglione. Gossip algorithms for distributed signal processing. *Proceedings of the IEEE*, 98(11):1847–1864, 2010.

15 Chris HQ Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 107–114. IEEE, 2001.

16 Nisrine Ghadban, Paul Honeine, Farah Mourad-Chehade, Joumana Farah, and Clovis Francis. Gossip algorithms for principal component analysis in networks. In *Signal Processing Conference (EUSIPCO), 2015 23rd European*, pages 2366–2370. IEEE, 2015.

17 Torben Hagerup and Christine Rüb. A guided tour of chernoff bounds. *Information Processing Letters*, 33(6):305–308, 1990.

18 Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

19 Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

20 Prateek Jain, Chi Jin, Sham M. Kakade, Praneeth Netrapalli, and Aaron Sidford. Streaming pca: Matching matrix bernstein and near-optimal finite sample guarantees for oja's algorithm. In *Proceedings of the 29th Annual Conference on Computational Learning Theory (COLT)*, 2016.

21 David Kempe and Frank McSherry. A decentralized algorithm for spectral analysis. *Journal of Computer and System Sciences*, 74(1):70–83, 2008.

22 Satish Babu Korada, Andrea Montanari, and Sewoong Oh. Gossip pca. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 209–220. ACM, 2011.

23 Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.

24 F. McSherry. Spectral partitioning of random graphs. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 529–537, 2001.

25 Frank McSherry. Spectral partitioning of random graphs. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 529–537, 2001.

26 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

27 Gemma Morral, Pascal Bianchi, and Jérémie Jakubowicz. Asynchronous distributed principal component analysis using stochastic approximation. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 1398–1403. IEEE, 2012.

28 Elchanan Mossel, Joe Neeman, and Allan Sly. Belief propagation, robust reconstruction and optimal recovery of block models. In *Conference on Learning Theory*, pages 356–370, 2014.

29 Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

30 Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

31 Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

32 Mark Rudelson and Roman Vershynin. Smallest singular value of a random rectangular matrix. *Communications on Pure and Applied Mathematics*, 62(12):1707–1739, 2009.

33 Semih Salihoglu, Jaeho Shin, Vikesh Khanna, Ba Quan Truong, and Jennifer Widom. Graft: A debugging tool for apache giraph. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1403–1408. ACM, 2015.

34 Daniel Spielman. Spectral graph theory: Spectral partitiong in a stochastic block model. University Lecture, 2015.

35 Ji-guang Sun. Rounding-error and perturbation bounds for the cholesky and ldlt factorizations. *Linear algebra and its applications*, 173:77–97, 1992.

36 Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.

37 John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

38 Van Vu. A simple SVD algorithm for finding hidden partitions. *Combinatorics, Probability and Computing*, 27(1):124–140, 2018.