Distributed Detection of Clusters of Arbitrary Size

Bogdan-Adrian Manghiuc(✉)

School of Informatics, The University of Edinburgh, Edinburgh, UK
b.a.manghiuc@sms.ed.ac.uk

Abstract. Graph clustering is a fundamental technique in data analysis with a vast number of applications in computer science and statistics. In theoretical computer science, the problem of graph clustering has received significant research attention over the past two decades, which has led to pivotal algorithmic breakthroughs. However, the design of most graph clustering algorithms is based on complicated techniques from computational optimisation, which are not applicable for processing massive data sets stored in physically remote locations.

In this work we present a novel distributed algorithm for graph clustering. Most of the previous algorithms only work for graphs with balanced-sized clusters, which restrict their applications in many practical settings. Our proposed algorithm works for graphs with clusters of *arbitrary size* and its performance is analysed with respect to every *individual* cluster. In addition, our algorithm is easy to implement, and only requires a poly-logarithmic number of rounds for many graphs occurring in practice.

Keywords: Distributed computing · Graph clustering · Randomised algorithms

1 Introduction

Graph clustering, also known as community detection, is one of the most fundamental problems in algorithms with applications in distributed computing, machine learning, network analysis, and statistics. Over the past four decades, graph clustering algorithms have been extensively studied from both the theoretical and applied perspectives [10,21]. On the theoretical side, the problem is known as graph partitioning and is one of the most fundamental NP-hard problems. Among the many reasons, we mention its connection to several important topics in theoretical computer science including the Unique Games Conjecture and the Small Set Expansion Conjecture. Because of this, most graph clustering algorithms with better approximation guarantee are based on complicated spectral and convex optimisation techniques [17,22], whose runtime is slow even in the centralised setting. From the practical point of view, graph clustering is a key component in unsupervised learning, and has been widely applied in data mining, analysis of social networks, and statistics. In particular, since many graphs

T. Jurdziński and S. Schmid (Eds.): SIROCCO 2021, LNCS 12810, pp. 370–387, 2021.
https://doi.org/10.1007/978-3-030-79527-6_21

occurring in practice (e.g. social networks) are stored in physically distributed servers (sites), designing simple and more practical distributed algorithms, with better performance, has received a lot attention in recent years [2–4,8,12,23,24].

We study graph clustering algorithms in the distributed setting. We assume that the input $G = (V, E)$ is an unweighted distributed network over a set of $|V| = n$ nodes and $|E| = m$ edges. The set of nodes is always fixed and there are no node failures. Each node is a computational unit communicating only to its neighbours. We consider the synchronous timing model when, in each round, a node can either send the same message to all its neighbours or choose not to communicate. We assume that every node knows the rough size of $|V|$, which is not difficult to approximate, however the global structure of G is unknown to each node. Every node v has a unique[1] identifier $\mathsf{ID}(v)$ of size $O(\log(n))$. We will assume that any message sent by a node v will also contain $\mathsf{ID}(v)$.

In this work we study the clustering problem when the input G consists of k well-defined clusters $S_1, \cdots, S_k$ that form a partition of V, i.e., it holds that $S_i \cap S_j = \emptyset$ for $i \neq j$ and $\bigcup_{1 \leq i \leq k} S_i = V$. We allow the nodes in the network to exchange information with their neighbours over a number of rounds T. At the end of the T rounds, every node v determines a label indicating the cluster in which it belongs. Our objective is to design a distributed algorithm which guarantees that: (1) most nodes within the same cluster would receive the same label, and (2) every cluster would have its own unique label. The performance of our algorithm is measured by (1) the total number of proceeded rounds T, (2) the approximation guarantee, i.e., how many nodes in each cluster receive the correct label, and (3) the total message complexity, i.e., the total number of words exchanged among the nodes.

Structure of Clusters. The performance of a clustering algorithm always depends on the inherent cluster structure of the network: the more significant the cluster structure is, the easier the algorithm could approximate it. To quantify the significance of the cluster structure associated with the underlying graph, we follow the previous reference [18,19] and introduce the gap assumption. For any set $S \subset V$, let the conductance of S be

$$\phi_G(S) \triangleq \frac{|\partial(S)|}{\mathrm{vol}(S)},$$

where $\partial(S) = E(S, V \setminus S)$ is the set of edges crossing S and $V \setminus S$, and $\mathrm{vol}(S)$ is the sum of degrees of nodes in S. We define the *k-way* expansion of G by

$$\rho(k) \triangleq \min_{\text{partitions } S_1, \ldots, S_k} \max_{1 \leq i \leq k} \phi_G(S_i),$$

and we call a partition $\{S_i\}_{i=1}^k$ that achieves $\rho(k)$ an optimal partitioning.

One of the basic facts in spectral graph theory is a tight connection between $\rho(k)$ and the eigenvalues of the normalised Laplacian matrix of G. In particular, Lee et al. [14] proved the so-called higher-order Cheeger inequality:

[1] Every node v can randomly select a number between $[1, \mathrm{poly}(n)]$, such that, with high probability, those numbers are distinct.

$$\frac{\lambda_k}{2} \leq \rho(k) \leq O(k^3)\sqrt{\lambda_k}, \tag{1}$$

where $0 = \lambda_1 \leq \cdots \leq \lambda_n \leq 2$ are the eigenvalues of the normalised Laplacian of G. By definition, it is easy to see that a small value of $\rho(k)$ ensures the existence of disjoint $S_1, \ldots, S_k$ of low conductance. On the other hand, by (1) we know that a large value of λ_{k+1} implies that no matter how we partition G into $k+1$ subsets $S_1, \ldots, S_{k+1}$, there will be at least one subset S_i for which $\phi_G(S_i) \geq \frac{\lambda_{k+1}}{2}$. To formalise this intuition, we follow the previous reference (e.g., [19,23]) and define

$$\Upsilon_G(k) \triangleq \frac{\lambda_{k+1}}{\rho(k)}.$$

By definition, a large value of $\Upsilon_G(k)$ would ensure that G has k well-defined clusters.

Our Result. Our main result is an improved distributed graph clustering algorithm for inputs with k-well defined clusters. For the ease of presentation, we assume that G is d-regular, however our algorithm works and the analysis follows as long as the maximum degree $d_{\max}$ of G and the minimum degree $d_{\min}$ satisfy $d_{\max}/d_{\min} = O(1)$. Our result is as follows:

Theorem 1 (Main Result). *Let $G = (V, E)$ be a d-regular network with $|V| = n$ nodes, $|E| = m$ edges and k optimal clusters $S_1, \ldots, S_k$. If $\Upsilon_G(k) = \omega\left(k^4 \log^3(n)\right)$, there is a distributed algorithm that finishes in $T = O\left(\frac{\log(n)}{\lambda_{k+1}}\right)$ rounds, such that the following three statements hold:*

1. *For any cluster S_j of size $|S_j| \leq \log(n)$, every node $u \in S_j$ will determine the same label. Moreover, this label is $\mathsf{ID}(v)$ for some $v \in S_j$.*
2. *With probability at least 0.9, for any cluster S_j of size $|S_j| > \log(n)$, all but $o(|S_j|)$ nodes $u \in S_j$ will determine the same label. Moreover, this label is $\mathsf{ID}(v)$ for some $v \in S_j$.*
3. *With probability at least 0.9, the total information exchanged among the n nodes, i.e. the message complexity is $\widetilde{O}\left(\frac{n^2}{\lambda_{k+1}}\right)$, where $\widetilde{O}(\cdot)$ hides $\operatorname{poly}\log(n)$ factors.*

Now we discuss the significance of our result. First of all, notice that $\lambda_{k+1} = \Omega(1)$ in many practical settings [16,18], and in this case our algorithm finishes in $T = O(\log n)$ rounds. Secondly, our result significantly improves the previous work with respect to the approximation ratio. As far as we know, the vast majority of the previous algorithms for distributed clustering are analysed with respect to the *total* volume (or number) of misclassified nodes *over all clusters* (e.g., [2–4,23]). However, this form of approximation is unsatisfactory when the underlying graph contains clusters with very unbalanced sizes, since an upper bound on the total volume (or number) of misclassified nodes could still imply that nodes from a smaller cluster are completely misclassified. Our current work successfully overcomes this downside by analysing the approximation guarantee with respect to *every* approximated cluster and its optimal correspondent.

To the best of our knowledge, such a strong approximation guarantee with respect to every optimal cluster is only known in the centralised setting [13,19]. We show that such result can be obtained for distributed algorithms as well.

It is not difficult to image that obtaining this strong approximation guarantee would require a more refined analysis on the *smaller* clusters, since clusters with different size might have different orders of mixing time if random walk based processes are needed when performing the algorithm. Surprisingly, we are able to show that our algorithm is able to *perfectly* recover every small cluster. To the best of our knowledge, such a result of perfectly recovering small clusters is unknown even for centralised algorithms, and our developed subroutine of small cluster recovery might have other applications.

Finally, as a key component of our algorithm, we present a distributed subroutine which allows most nodes to estimate the size of the cluster they belong. This subroutine is based on the power method applied to a number of initial vectors. We show that the information retrieved by each node after this process is sufficient for most nodes to obtain good estimates for the size of their cluster. We believe that our present algorithm and the developed techniques would inspire further applications for many different problems concerning multiple and parallel random walks [11,20] or testing clusters of communities in networks [9,15].

Related Work. There is a large amount of work on graph clustering over the past decades, and here we discuss the ones closely related to ours. First of all, there have been several studies on graph clustering where the presence of the cluster structure is guaranteed by some spectral properties of the Laplacian matrix of an input graph. Von Luxburg [16] studies spectral clustering, and discusses the influence of the eigen-gap on the quality of spectral clustering algorithms. Peng et al. [19] analyse spectral clustering on well-clustered graphs and show that, when there is a gap between λ_{k+1} and $\rho(k)$, the approximation guarantee of spectral clustering can be theoretically analysed. Gharan and Trevisan [18] designed an approximation algorithm that, under some condition on the relationship between λ_k and λ_{k+1}, returns k clusters $S_1, \ldots, S_k$ such that both the inner and outer conductance of each S_i can be theoretically analysed. Allen-Zhu et al. [1] present a local algorithm for finding a cluster with improved approximation guarantee under some gap assumption similar with ours.

For distributed graph clustering, the work most related to ours is the distributed algorithm developed by Sun and Zanetti [23]. In comparison to our algorithm, the algorithm in [23] only holds for graphs that consist of clusters of balanced sizes, and the approximation guarantee (i.e., the number of misclassified nodes) of their algorithm is with respect to the volume of the input graph, while the approximation guarantee of our algorithm is with respect to *every* individual cluster. Becchetti et al. [3] presented a distributed graph clustering algorithm for the case $k = 2$, based on an Averaging dynamics process. However, their analysis holds only for a restricted class of graphs exhibiting sparse cuts. Becchetti et al. [4] extended the results for a more general class of volume regular networks with k clusters. Nonetheless, their results apply to reasonably balanced

networks, which is a setting more restricted than ours. Finally, we would like to mention a related sequence of work on decomposing graphs into expanders [6,7]. However, we highlight that these algorithms cannot be applied in our setting for the following reasons: the number of partitioning sets could be much larger than the initial number of clusters k and the decomposition also allows some fraction of nodes not to be in any cluster [7].

Notation. We consider the input network $G = (V, E)$ to be an unweighted d-regular network on $|V| = n$ nodes and $|E| = m$ edges. The $n \times n$ adjacency matrix is denoted A_G and is defined as $A_G(u, v) = 1$ if $\{u, v\} \in E$ and $A_G(u, v) = 0$ otherwise. The normalised Laplacian of G is the $n \times n$ matrix defined as

$$\mathcal{L}_G \triangleq I - \frac{1}{d} \cdot A_G.$$

We denote the eigenvalues of $\mathcal{L}_G$ by $0 = \lambda_1 \leq \cdots \leq \lambda_n \leq 2$. For any subset $S \subseteq V$, we denote the characteristic vector $\mathbf{1}_S \in \mathbb{R}^n$ by $\mathbf{1}_S(v) = 1$ if $v \in S$ and $\mathbf{1}_S(v) = 0$ otherwise. For brevity, we will write $\mathbf{1}_v$ whenever $S = \{v\}$.

We will consider the setting when the input network G contains k disjoint clusters $S_1, \ldots, S_k$, that form a partition of V. For a given node $v \in V$, we will denote by $\mathcal{S}(v)$ the cluster that contains v. We will write $\text{Broadcast}_u(\mathsf{Message})$ whenever a node u sends a $\mathsf{Message}$ to its neighbours and we will drop the subscript u whenever that is clear from the context. We will denote the label of a node v by $L(v)$ and we will assume that initially $L(v) = \perp$, for all nodes v. Throughout our algorithm, some nodes $v \in V$ will become *active*. We will use the notation v^* whenever referring to an active node v.

2 Algorithm Description

Our algorithm consists in three major phases: Averaging, Small Detection and Large Detection, which we will describe individually.

Averaging Phase: The Averaging phase (Algorithm 1) consists in the execution of n different diffusion processes, one for every node. To each diffusion process, say corresponding to a node v, we associate a set of vectors $\{x_i^v\}_i$ such that, after every round i, every node u in the network will store the value $x_i^v(u)$. The value $x_i^v(u)$ is the mass value that node u received from node v after i rounds. Initially (round 0) the diffusion process starts from $x_0^v = \frac{1}{\sqrt{d}}\mathbf{1}_v$, i.e. all mass value $\frac{1}{\sqrt{d}}$ is concentrated at node v with 0 mass to the other nodes. For a general round i, the vector x_i^v is constructed iteratively, via the recursive formula

$$x_i^v(u) = \frac{1}{2}x_{i-1}^v(u) + \frac{1}{2d} \sum_{\{u,w\} \in E} x_{i-1}^v(w), \tag{2}$$

for any u. We remark that, the only information a node u needs in round i are the values $x_{i-1}^v(w)$ for all its neighbours w. We note that at any round i, node u

does not need to update the values x_i^v for all nodes v in the network. Instead, u focuses on the diffusion processes started at the nodes which u has already seen throughout the process. To keep track of the already seen nodes, u maintains the set $\mathsf{Seen}(u)$ (See Line 2 of Algorithm 1).

Now let us discuss the intuition behind this phase. The goal of the diffusion process started at node v is that, after $T = \Theta\left(\frac{\log(n)}{\lambda_{k+1}}\right)$ rounds, the entire mass $\frac{1}{\sqrt{d}}$ is split roughly equally among all nodes inside the cluster $\mathcal{S}(v)$, with very few of this mass exiting the cluster. A closer look at Eq. (2) tells us that the diffusion process started at v is nothing but a T-step, 1/2-lazy random walk process starting from the vector $\frac{1}{\sqrt{d}}\mathbf{1}_v$. It is well known [5] that, assuming G is connected, the vectors x_i^v will converge to the uniform distribution as i goes to infinity. However, if the process runs for merely $T = \Theta\left(\frac{\log(n)}{\lambda_{k+1}}\right)$ iterations, one should expect the vector x_T^v to be close to the (normalised) indicator vector of the cluster $\mathcal{S}(v)$. This is because $\frac{\log(n)}{\lambda_{k+1}}$ corresponds to the local mixing time inside the cluster $\mathcal{S}(v)$. Therefore, after T rounds, we expect for every $u \in \mathcal{S}(v)$, the values $x_T^v(u)$ to be similar and significantly greater than 0, while for nodes $w \notin \mathcal{S}(v)$, we expect the values $x_T^v(w)$ to be close to 0.

At the end of the Averaging phase, based on the values $\{x_T^v(u)\}_v$ each node u computes an estimate ℓ_u for the size of its cluster. We define the estimates as

$$\ell_u \triangleq \frac{3}{d \cdot \sum_v \left[x_T^v(u)\right]^2},$$

and we will show that, for most nodes u, the estimates satisfy $\ell_u \in [|\mathcal{S}(u)|, 4|\mathcal{S}(u)|]$ (see Lemma 6).

Algorithm 1. $\mathtt{Average}\,(u, T)$

Require: A node u, a number of rounds $T = \Theta\left(\frac{\log(n)}{\lambda_{k+1}}\right)$

1: Set $x_0^u(u) = \frac{1}{\sqrt{d}}$ and $x_0^v(u) = 0$ for all $v \neq u$ ▷ Initialisation step
2: $\mathsf{Seen}(u) = \{u\}$.
3: **for** $i = 1 \dots T$ **do**
4: Broadcast$\left(\left\{(x_{i-1}^v(u), \mathsf{ID}(v)) \,\middle|\, v \in \mathsf{Seen}(u)\right\}\right)$
5: **for all** $(x_{i-1}^v(w), \mathsf{ID}(v))$ that u receives **do**
6: Add v to $\mathsf{Seen}(u)$
7: **for all** $v \in \mathsf{Seen}(u)$ **do**
8: $x_i^v(u) = \frac{1}{2}x_{i-1}^v(u) + \frac{1}{2d}\sum_{w \sim u} x_{i-1}^v(w)$. ▷ Update the current status
9: **return** $\{x_T^v(u)\}$.

Small Detection Phase: The purpose of this phase is for every node u in a cluster of small size $|\mathcal{S}(u)| \leq \log(n)$ to determine its label. Again, we focus on the intuition behind this process and we refer the reader to Algorithm 2 for a formal description. From the perspective of a node u, we would like to use

the values $\{x_T^v(u)\}_v$ to decide which nodes are in its own cluster. Informally, the values $\{x_T^v(u)\}_{v\in\mathcal{S}(u)}$ should be similar and close to $\frac{1}{\sqrt{d}|\mathcal{S}(u)|}$ since, for every diffusion process started at $v \in \mathcal{S}(u)$ we expect the $\frac{1}{\sqrt{d}}$ mass to be equally distributed among all nodes in the cluster. At the same time, we expect the values $\{x_T^w(u)\}_{w\notin\mathcal{S}(u)}$ not to be very large, because they correspond to random walks started in a different clusters. Therefore, if $|\mathcal{S}(u)|$ is not too big, we expect to see a clear separation between $x_T^v(u)$ and $x_T^w(u)$, for any $v \in \mathcal{S}(u)$ and $w \notin \mathcal{S}(u)$.

Since u knows that it is a member of its own cluster, it can use $x_T^u(u)$ as a reference point. Namely u computes the pairwise differences

$$y_v \triangleq |x_T^u(u) - x_T^v(u)|,$$

for all v and sorts them in increasing order. Let us call $y_{v_1} \leq \cdots \leq y_{v_n}$ to be those values. Based on the previous remarks, it should be expected that the first $|\mathcal{S}(u)|$ values are small and correspond to nodes $v \in \mathcal{S}(u)$. Then, u performs a binary search to find the exact size of its cluster $\mathcal{S}(u)$ (lines 4-10 of Algorithm 2). Finally, u sets its label to be the minimum[2] ID among nodes corresponding to the smallest $|\mathcal{S}(u)|$ values $\{y_{v_i}\}$.

Algorithm 2. `SmallDetection` $(u, \ell_u, \mathrm{RW}(u))$

Require: A node u, an estimated cluster size ℓ_u, a list of values $\mathrm{RW}(u) = \{x_T^v(u)\}$
1: **for all** v **do**
2: Compute $y_v = |x_T^u(u) - x_T^v(u)|$.
3: Sort the values $\{y_v\}$ and call the sorted ones $y_{v_1} \leq \cdots \leq y_{v_n}$
4: Let $i_{\text{low}} = 1$, $i_{\text{high}} = \ell_u$. ▷ The binary search step
5: **while** $i_{\text{low}} + 1 < i_{\text{high}}$ **do** ▷ At the end of the loop, we have $i_{\text{low}} = |\mathcal{S}(u)|$
6: $i = \lfloor \frac{i_{\text{low}} + i_{\text{high}}}{2} \rfloor$
7: **if** $y_{v_i} < \frac{9}{10\sqrt{d}\cdot i}$ **then**
8: $i_{\text{low}} = i$
9: **else**
10: $i_{\text{high}} = i$.
11: $L(u) = \min_{v_i} \{\mathsf{ID}(v_i) | i \leq i_{\text{low}}\}$ ▷ Determining the label
12: **return** $L(u)$.

At the end of this phase we would like to stress two important facts. First of all, it is really crucial that the cluster $\mathcal{S}(u)$ has size $|\mathcal{S}(u)| \leq \log(n)$. Otherwise, the values $\{x_T^v(u)\}_{v\in\mathcal{S}(u)}$ would be too small for u to distinguish them. Therefore, we cannot use this approach to determine the label of all nodes in the network. Secondly, we remark that for the algorithm to work, every node u should be in possession of the value $x_T^u(u)$. This can be ensured only if all nodes start their own diffusion process.

[2] The minimum does not play any special role here, it is only used to guarantee consensus among all nodes in the same cluster. The maximum ID works just as fine.

Large Detection Phase: In this phase, the algorithm will detect the remaining clusters of large and medium size, that are clusters S with $|S| > \log(n)$. The formal description of this phase can be found in Algorithm 3. Again, a node u in such a cluster would like to use the values $\{x_T^v(u)\}_v$ to determine the composition of its cluster. Unfortunately, node u cannot trust all values $x_T^v(u)$ because of the error in the diffusion processes caused by some mass exiting the cluster.

To overcome this difficulty, we use a different approach. We want each cluster S_j to select some representatives, which we will refer to as active nodes. The label of the cluster S_j will be the minimum ID among the active nodes in S_j. The purpose of this selection is to avoid the (bad) nodes for which the diffusion process does not behave as expected. To that extent, we let each node u activate itself independently (Line 3 of Algorithm 3), with probability

$$p(u) \triangleq \frac{5 \cdot \log(100k)}{\ell_u}.$$

Since for most nodes $\ell_u \approx |\mathcal{S}(u)|$, the probability $p(u)$ is large enough to ensure that: every cluster will have at least one active node and, in expectation, there will not be too many active nodes overall. If a node u becomes active, it will announce this in the network, along with its estimated cluster size ℓ_u. This happens throughout T rounds of communication (Lines 6–14 of Algorithm 3). In such a round j, every node v in the network keeps a list $\mathsf{Act}(v)$ of the active nodes v has seen up to round j. Then, v checks which of those active nodes he has not yet communicated, broadcasts them in a $\mathsf{Message}$ (Line 12 of Algorithm 3) and marks them as Sent. This process ensures that every node v announces each active node at most once, which significantly reduces the communication cost.

Coming back to node u, at the end of the T rounds u has seen all active nodes $v^* \in \mathsf{Act}(u)$. Node u considers an active node v^* to be in $\mathcal{S}(u)$, if two conditions are satisfied: 1) its estimated cluster size ℓ_u and ℓ_{v^*} are similar and 2) the value $x_T^{v^*}$ is similar to what u expects to see. More precisely, u sets the threshold

$$t_u \triangleq \frac{1}{2\sqrt{d}\ell_u},$$

and considers its set of candidates

$$\mathsf{Cand}(u) \triangleq \left\{ v^* \middle| v^* \in \mathsf{Act}(u), \frac{\ell_u}{4} \leq \ell_{v^*} \leq 4\ell_u \text{ and } x_T^{v^*}(u) \geq t_u \right\}.$$

The set of candidates $\mathsf{Cand}(u)$ represents the set of active nodes that u believes are in its own cluster. If $\mathsf{Cand}(u) \neq \emptyset$, then u sets its label to be the $\min_{v^* \in \mathsf{Cand}(u)} \mathsf{ID}(v^*)$. Otherwise, if u is unlucky so that $\mathsf{Cand}(u) = \emptyset$, then u randomly chooses an active node $v^* \in \mathsf{Act}(u)$ and selects its label as $L(u) = \mathsf{ID}(v^*)$ (Lines 20−23 of Algorithm 3).

Algorithm 3. `LargeDetection` $(u, \ell_u, \mathrm{RW}(u), T)$

Require: A node u, an estimated cluster size ℓ_u, a list of values $\mathrm{RW}(u) = \{x_T^v(u)\}$, the number of rounds T

1: Set $\mathsf{Act}(u) = \emptyset$, $\mathsf{Cand}(u) = \emptyset$, $\mathsf{Sent}(v) = \text{false}, \forall v$ ▷ Initialisation step
2: **if** $L(u) = \perp$ **then**
3: Activate u with probability $p(u) = \frac{5\log(100k)}{\ell_u}$. ▷ Activation step
4: **if** u becomes active **then**
5: Set $\mathsf{Act}(u) = \{(u, \ell_u)\}$
6: **for** $j = 1 \dots T$ **do** ▷ Propagation step
7: $\mathsf{Message} = \emptyset$
8: **for all** $(v^*, \ell_{v^*}) \in \mathsf{Act}(u)$ and $\mathsf{Sent}(v^*) = \text{false}$ **do**
9: Add (v^*, ℓ_{v^*}) to $\mathsf{Message}$
10: Set $\mathsf{Sent}(v^*) = \text{true}$
11: **if** $\mathsf{Message} \neq \emptyset$ **then**
12: Broadcast $(\mathsf{Message})$.
13: **for all** v^* such that u received (v^*, ℓ_{v^*}) in round j **do**
14: Add (v^*, ℓ_{v^*}) to $\mathsf{Act}(u)$.
15: Set $t_u = \frac{1}{2\sqrt{d}\cdot\ell_u}$
16: **for all** $(v^*, \ell_{v^*}) \in \mathsf{Act}(u)$ **do**
17: **if** $\frac{\ell_{v^*}}{4} \leq \ell_u \leq 4\ell_{v^*}$ and $x_T^{v^*}(u) \geq t_u$ **then**
18: Add v^* to $\mathsf{Cand}(u)$
19: **if** $\mathsf{Cand}(u) \neq \emptyset$ **then** ▷ Labeling step
20: Set $L(u) = \min_{v^* \in \mathsf{Act}(u)} \{\mathsf{ID}(v^*)\}$
21: **else**
22: Choose a random $(v^*, \ell_v^*) \in \mathsf{Act}(u)$
23: Set $L(u) = \mathsf{ID}(v^*)$
24: **return** $L(u)$.

The Main Algorithm: Now we bring together all three subroutines and present our main Algorithm 4. We note that once a node u has determined their label, that will not change in the future. This is because Algorithm 3 can only change the label if $L(u) = \perp$ initially. Moreover, even if a node has determined their label in the Small Detection phase, they still participate in the Large Detection phase since they are active parts in the Propagation step (Lines 6–14) of Algorithm 3.

3 Analysis of the Algorithm

In this section we analyse our distributed algorithm and prove Theorem 1. Remember that we assume G is a d-regular network with optimal k clusters $S_1, \dots, S_k$. Moreover, we work in the regime when G satisfies the assumption that

$$\Upsilon_G(k) = \omega\left(k^4 \log^3(n)\right). \tag{3}$$

Algorithm 4. `Cluster` (u, n)

Require: A node u, the number of nodes in the network n.

1: Set $T = \Theta\left(\frac{\log(n)}{\lambda_{k+1}}\right)$. ▷ Choose $\lambda_{k+1} = \frac{1}{\text{poly}\log(n)}$ in practice
2: Set $L(u) = \perp$.
3: Let $\text{RW}(u) =$ `Average`(u, T). ▷ Perform the Averaging phase
4: Let $\ell_u = 3/\left(d\sum_v [x_T^v(u)]^2\right)$ be the estimate for $|\mathcal{S}(u)|$.
5: **if** $\ell_u \leq 4\log(n)$ **then**
6: $\quad L(u) =$ `SmallDetection` $(u, \ell_u, \text{RW}(u))$ ▷ Perform the Small Detection phase
7: $L(u) =$ `LargeDetection` $(u, \ell_u, \text{RW}(u), T)$ ▷ Perform the Large Detection phase
8: **return** $L(u)$.

For brevity, we will use Υ instead of $\Upsilon_G(k)$. We will structure the analysis of our algorithm in five subsections. In Subsect. 3.1 we recap some of the results in [23] and present the guarantees achieved after the Averaging phase of the algorithm. In Subsect. 3.2 we show that most nodes in the network can obtain a good estimate for the size of their cluster. In Subsect. 3.3 we deal with the analysis of the Small Detection phase of the algorithm. We will ultimately show that for all clusters S_j of size $|S_j| \leq \log(n)$, all nodes $u \in S_j$ will determine the same label, unique for the cluster S_j. In Subsect. 3.4 we analyse the Large Detection phase of the algorithm. In this phase, most nodes in clusters of size at least $\log(n)$ will decide on a common label that is unique for the cluster. We also show that the number of misclassified nodes for each cluster is small. Finally, we conclude with the proof of Theorem 1 in Subsect. 3.5.

3.1 Analysis of the Averaging Phase

Recall that the Averaging phase consists in performing n diffusion processes for $T = \Theta\left(\frac{\log(n)}{\lambda_{k+1}}\right)$ rounds. For a node v, its own diffusion process can be viewed as a lazy random walk starting at $x_0^v = \frac{1}{\sqrt{d}}\mathbf{1}_v$ and following the recursion

$$x_{i+1}^v = Px_i^v,$$

where

$$P \triangleq \frac{1}{2}\cdot I + \frac{1}{2d}\cdot A_G = I - \frac{1}{2}\mathcal{L}_G$$

is the transition matrix of the process. It is known that, assuming G is connected, the vectors $\{x_i^v\}$ will converge to the stationary distribution as i goes to infinity. However, if the power method runs for $T = \Theta\left(\frac{\log(n)}{\lambda_{k+1}}\right)$ phases, we expect $x_T^v \approx \frac{1}{\sqrt{d}\cdot|\mathcal{S}(v)|}\cdot\mathbf{1}_{\mathcal{S}(v)}$. Sun and Zanetti [23] formalise this intuition and give a concrete version of the above observation (See Lemma 2). The intuition behind their result lies in the fact that, for graphs G with k good clusters, there is a strong connection between the bottom k eigenvectors of $\mathcal{L}_G$ and the normalised indicator vectors of the clusters [19].

We will now introduce the notation required to formalise the above discussion. Let $f_1, \ldots, f_k$ be the bottom k eigenvectors of $\mathcal{L}_G$ and let $\{\chi_{S_1}, \ldots, \chi_{S_k}\}$ be the normalised indicator vectors of the clusters S_i, that is $\chi_{S_i} \triangleq \frac{1}{\sqrt{|S_i|}} \mathbf{1}_{S_i}$, for all i. Let $\widetilde{\chi}_i$ be the projection of f_i onto $\text{span}\{\chi_{S_1}, \ldots \chi_{S_k}\}$ and let $\{\widehat{\chi}_i\}$ be the vectors obtained from $\{\widetilde{\chi}_i\}$ by applying the Gram-Schmidt orthonormalisation[3]. For any node v, we define the discrepancy parameter

$$\alpha_v \triangleq \sqrt{\frac{1}{d} \sum_{i=1}^{k} (f_i(v) - \widehat{\chi}_i(v))^2}.$$

We are now ready to state the result relating x_T^v to the indicator vector of the cluster $\mathbf{1}_{\mathcal{S}(v)}$:

Lemma 2 (Adaptation of Lemma 4.4 in [23]). *For any $v \in V$, if we run the lazy random walk for $T = \Theta\left(\frac{\log n}{\lambda_{k+1}}\right)$ rounds, starting at $x_0^v = \frac{1}{\sqrt{d}} \mathbf{1}_v$, we obtain a vector x_T^v such that*

$$\left\| x_T^v - \frac{1}{\sqrt{d} \cdot |\mathcal{S}(v)|} \cdot \mathbf{1}_{\mathcal{S}(v)} \right\|^2 = O\left(\frac{k^2}{d \cdot \Upsilon \cdot |\mathcal{S}(v)|} + \alpha_v^2 \right). \tag{4}$$

One can view the RHS of (4) as an upper bound for the total error of the diffusion process that started at v. To that extent, let us define the set of vectors

$$\boldsymbol{\varepsilon}_v \triangleq x_T^v - \frac{1}{\sqrt{d} \cdot |\mathcal{S}(v)|} \cdot \mathbf{1}_{\mathcal{S}(v)} \tag{5}$$

and we will use for shortend $\varepsilon_{(v,u)} = \boldsymbol{\varepsilon}_v(u)$. It is important to note that the order of the pair matters, since $\varepsilon_{(v,u)}$ corresponds to a diffusion process started at v, while $\varepsilon_{(u,v)}$ corresponds to a diffusion process started at u.

Under this notation, Eq. (4) becomes

$$\sum_{u \in V} \varepsilon_{(v,u)}^2 \leq C_\varepsilon \left(\frac{k^2}{d \cdot \Upsilon \cdot |\mathcal{S}(v)|} + \alpha_v^2 \right), \tag{6}$$

for some absolute constant C_ε. While one should expect each individual error $\varepsilon_{(v,u)}$ to be relatively small, i.e. $O\left(\frac{1}{|\mathcal{S}(v)|}\right)$, it is not immediately clear why this should be the case. Indeed, the presence of α_v in Eq. (6) can cause significant perturbation. Given the relatively complicated definition of this parameter, the only upper bound we are aware of is the following:

Lemma 3. *It holds that*

$$\sum_{v \in V} \alpha_v^2 = O\left(\frac{k^2}{d \cdot \Upsilon} \right) \leq \frac{C_\alpha \cdot k^2}{d \cdot \Upsilon}, \tag{7}$$

for some absolute constant $C_\alpha > 0$.

[3] For a more detailed discussion of the connection between the sets $\{f_i\}$, $\{\chi_{S_i}\}$, $\{\widetilde{\chi}_i\}$ we refer the reader to [19].

3.2 Estimating the Cluster Size

In this section we will show that most nodes in every cluster are able to estimate approximately the size of their cluster. Recall that the estimate that each node u computes is

$$\ell_u = \frac{3}{d\sum_v \left[x_T^v(u)\right]^2},$$

and we want to show that for most nodes u we have that $|\mathcal{S}(u)| \leq \ell_u \leq 4 \cdot |\mathcal{S}(u)|$. To that extent, we split the set of (bad) nodes, that do not obey the above condition, into two categories:

$$\mathcal{B}_{\text{big}} \triangleq \left\{ u \middle| \ell_u > 4\,|\mathcal{S}(u)| \right\} \qquad \text{and} \qquad \mathcal{B}_{\text{small}} \triangleq \left\{ u \middle| \ell_u < |\mathcal{S}(u)| \right\}.$$

Moreover we let

$$\mathcal{B}_\ell \triangleq \mathcal{B}_{\text{big}} \cup \mathcal{B}_{\text{small}}$$

and we will show that in each cluster, only a small fraction of nodes can be in $\mathcal{B}_\ell$. We start with set $\mathcal{B}_{\text{big}}$ and here we show that only a small fraction of nodes in each cluster can be in $\mathcal{B}_{\text{big}}$.

Lemma 4. *For every cluster S_j it holds that*

$$|\mathcal{B}_{\text{big}} \cap S_j| \leq \frac{|S_j|}{2 \cdot 500k \cdot \log(nk)}.$$

Now we focus on the set $\mathcal{B}_{\text{small}}$. In this case, we will prove something stronger, namely that in each cluster S_j the fraction of nodes estimating some value ℓ is directly proportional to the value of ℓ. In other words, the smaller the value of ℓ, the fewer the number of nodes will estimate it. This result is crucial for the analysis of the Large Detection phase. To that extent, we define the level sets

$$\mathcal{B}^i_{\text{small}} \triangleq \left\{ u \middle| \ell_u < \frac{|\mathcal{S}(u)|}{2^{i-1}} \right\},$$

for $i = 1, \ldots, \log(n)$. The following result formalises our discussion.

Lemma 5. *For every cluster S_j and every $i = 1, \ldots, \log(n)$ it holds that*

$$\left|\mathcal{B}^i_{\text{small}} \cap S_j\right| \leq \frac{|S_j|}{2^i \cdot 500k \cdot \log(nk)}.$$

Now we are ready to state and prove the main result of this subsection.

Lemma 6. *Almost all nodes $u \in V$ have a good approximation $\ell_u \approx |\mathcal{S}(u)|$. That is, for every cluster S_j the following conditions hold:*

1. $\frac{|\mathcal{B}_\ell \cap S_j|}{|S_j|} \leq \frac{1}{500k \cdot \log nk}$;
2. $\forall u \notin \mathcal{B}_\ell$, *it holds that* $|\mathcal{S}(u)| \leq \ell_u \leq 4|\mathcal{S}(u)|$.

Proof. Applying Lemmas 4 and 5 we have that

$$\frac{|\mathcal{B}_\ell \cap S_j|}{|S_j|} \leq \frac{|\mathcal{B}_{\text{small}} \cap S_j|}{|S_j|} + \frac{|\mathcal{B}_{\text{big}} \cap S_j|}{|S_j|} \leq \frac{1}{500k \cdot \log(nk)}.$$

□

3.3 Analysis of the Small Detection Phase

This subsection is dedicated to the analysis of the Small Detection phase of our clustering algorithm and thus to the analysis of Algorithm 2. In this section we will show that our algorithm will perfectly recover all clusters of small size. We first introduce some notation. Up to a permutation of the indices, without loss of generality we consider the clusters $S_1, \ldots, S_p$ such that

$$|S_i| \leq \log(n),$$

for each $1 \leq i \leq p \leq k$. Moreover, we will denote by

$$\mathcal{A} = S_1 \cup \cdots \cup S_p$$

to be the union of these clusters. The proof of our claim lies on the key observation that, for nodes $u \in \mathcal{A}$, there is a large enough gap between the mass values of diffusion processes started in the same cluster and processes started indifferent clusters. This observation is formalised below.

Lemma 7. *For any $u \in \mathcal{A}$ and $v \in V$ the following statements hold.*

1. *If $v \in \mathcal{S}(u)$, then $|x_T^u(u) - x_T^v(u)| \leq \frac{1}{100\sqrt{d} \cdot \log(n)}$;*
2. *If $v \notin \mathcal{S}(u)$, then $|x_T^u(u) - x_T^v(u)| \geq \frac{9}{10\sqrt{d} \cdot |\mathcal{S}(u)|}$*

Now we state and prove the main result of this subsection.

Lemma 8. *Let S_j be a cluster such that $|S_j| \leq \log(n)$. At the end of the Small Detection phase of the algorithm, all nodes $u \in S_j$ will agree on a unique label. Moreover, this label is $\mathsf{ID}(v)$, for some $v \in S_j$.*

Proof. Let S_j be some cluster of size $|S_j| \leq \log(n)$ and let $u \in S_j$ be some node. Applying Lemma 6, we see that all nodes $u \in S_j$ have an approximation $|S_j| \leq \ell_u \leq 4|S_j| \leq 4\log(n)$. Therefore every node $u \in S_j$ will certainly perform the Small Detection phase (Line 6 of Algorithm 4).

Firstly, u sorts the values $y_v = |x_T^u(u) - x_T^v(u)|$, for all $v \in V$. Say the sorted values are $y_{v_1} \leq \cdots \leq y_{v_n}$. Notice that if $w_1 \in S_j$ and $w_2 \notin S_j$, by Lemma 7 it must be that

$$|x_T^u(u) - x_T^{w_1}(u)| \leq \frac{1}{100\sqrt{d} \cdot \log(n)} < \frac{9}{10\sqrt{d} \cdot |S_j|} \leq |x_T^u(u) - x_T^{w_2}(u)|.$$

Therefore, u knows that the first $|S_j|$ values, namely $y_{v_1}, \ldots, y_{v_{|S_j|}}$ correspond to nodes in its own cluster and the other values correspond to nodes in different clusters. Thus, u needs to find a pair of consecutive values $y_{v_i} \leq y_{v_{i+1}}$ such that $v_i \in S_j$ and $v_{i+1} \notin S_j$.

To do this, u performs a binary search to find the size of its cluster. At any intermediate phase, say u considers the value y_{v_i}, for some i, and compares this with $\frac{9}{10\sqrt{d}\cdot i}$. If $y_{v_i} < \frac{9}{10\sqrt{d}\cdot i}$ we claim that $i \leq |S_j|$. If not, then $v_i \notin S_j$ and by Lemma 7 we have that

$$y_{v_i} \geq \frac{9}{10\sqrt{d}\cdot|S_j|} \geq \frac{9}{10\sqrt{d}\cdot i},$$

which gives the contradiction. Similarly, we can show that if $y_{v_i} \geq \frac{9}{10\sqrt{d}\cdot i}$ then $i > |S_j|$.

Once the node u finds the exact size of its cluster $|\mathcal{S}(u)|$, he also knows which nodes are in the same cluster, i.e. $v_1, \ldots, v_{|\mathcal{S}(u)|}$. Thus u can set its label to be the smallest ID among nodes in its cluster. This holds for all nodes $u \in S_j$ and all clusters S_j. □

3.4 Analysis of the Large Detection Phase

At this point, we will assume all n random walks have been completed, i.e. the T rounds have been executed and each node u has a list of values $\{x_T^v(u)\}_v$. From the perspective of u, we would like to use this information to decide which nodes are in the same cluster as u and which are not. Unfortunately, node u cannot trust all values $x_T^v(u)$ because of the error term $\varepsilon_{(v,u)}$. Going back to Eq. (6), we see that these errors are dependent on the parameters α_v. To overcome this issue, we define the notion of a γ-bad node, that is a node v for which the value of α_v is large relative to its cluster size:

Definition 9. *We say that a node u is γ-bad if*

$$\alpha_u \geq \frac{\gamma \cdot C_\alpha \cdot k^2}{d \cdot \Upsilon \cdot |\mathcal{S}(u)|}.$$

The set of γ-bad nodes is denoted by

$$\mathcal{B}_\gamma \triangleq \left\{ u \middle| \alpha_u \geq \frac{\gamma \cdot C_\alpha \cdot k^2}{d \cdot \Upsilon \cdot |\mathcal{S}(u)|} \right\}.$$

One should think about the γ-bad nodes as nodes for which the diffusion process does not necessarily behave as expected. Hence we want to avoid activating them since they are not good representatives for their own clusters. To put it differently, combining Eq. (6) with the above definition, we have the following remark:

Remark 10. For every node $v \notin \mathcal{B}_\gamma$ it holds that

$$\|\boldsymbol{\varepsilon}_v\|^2 \leq C_\varepsilon \left(1 + C_\alpha \cdot \gamma\right) \cdot \left(\frac{k^2}{d \cdot \Upsilon \cdot |\mathcal{S}(v)|}\right) \leq \frac{2 \cdot C_\varepsilon \cdot C_\alpha \cdot \gamma \cdot k^2}{d \cdot \Upsilon \cdot |\mathcal{S}(v)|}.$$

For the rest of the analysis, we will consider the value

$$\gamma \triangleq 500k \cdot \log(100k). \tag{8}$$

As for the question of how many γ-bad nodes are inside each cluster, the answer is not too many and is formalised in the Lemma bellow:

Lemma 11. *Let S_j be some cluster. It holds that*

$$|\mathcal{B}_\gamma \cap S_j| \leq \frac{|S_j|}{\gamma}.$$

The ultimate goal of the activation process is to select representatives for each cluster in such a way that the following conditions hold: (1) Every cluster has at least one active node, (2) The total number of active nodes is small and (3) No γ-bad node becomes active. Recall that each node activates independently with probability

$$p(u) = \frac{5 \log(100k)}{\ell_u}.$$

For most nodes, i.e. $u \in V \setminus \mathcal{B}_\ell$, the probabilities are good enough to ensure the three conditions hold. The tricky part is to deal with nodes $u \in \mathcal{B}_\ell$. More precisely, for nodes u such that $\ell_u \ll |\mathcal{S}(u)|$ and $u \in \mathcal{B}_\gamma$ the activation probability is simply too large to reason directly that no such node becomes active. We overcome this by first showing that, with high constant probability, no node $u \in \mathcal{B}_\ell$ becomes active, and based on this no node in $\mathcal{B}_\gamma$ becomes active as well. We formalise our discussion in Lemma 12, which is the main technical result of this subsection.

Lemma 12. *With probability at least* 0.9*, the following statements hold:*

A1. No node from $\mathcal{B}_\ell \cup \mathcal{B}_\gamma$ becomes active.
A2. Every cluster S_j contains at least one active node $v_j^* \in S_j \setminus \mathcal{B}_\ell$;
A3. The total number of active nodes is $n_a \leq 500k \cdot \log(100k)$;

Now we are ready to state the main result of this subsection.

Lemma 13. *At the end of the Large Detection phase, with probability at least* 0.9*, for any cluster S_j of size $|S_j| > \log(n)$, all but $o(|S_j|)$ nodes $u \in S_j$ will determine the same label. Moreover, this label is $\mathsf{ID}(v)$ for some $v \in S_j$.*

Proof (Sketch). We assume the conclusions of Lemma 12 hold. Fix some cluster S_j. We focus on the nodes $u \in S_j' = S_j \setminus \mathcal{B}_\ell$ and assume the other nodes are misclassified. This is sufficient since, by Lemma 6, $|S_j \cap \mathcal{B}_\ell| = o\left(|S_j|\right)$. Let s_j^* be the active node in S_j of smallest ID. By (A2) and (A1) we know s_j^* exists and $s_j^* \notin \mathcal{B}_\ell \cup \mathcal{B}_\gamma$. Let $u \in S_j'$ be a misclassified node. By the Algorithm's description we know one of the two conditions must happen:

1. $s_j^* \notin \mathsf{Cand}(u)$
2. $\exists v^* \notin S_j$, but $v^* \in \mathsf{Cand}(u)$

We look at each condition separately. For the first one, since $s_j^*, u \in S'_j$, it must be that $x_T^{s_j^*}(u) < t_u$. This means that the error $\varepsilon_{(s_j^*,u)}$ is large in absolute value: $\varepsilon_{(s_j^*,u)} < -\frac{1}{2\sqrt{d}|S_j|}$. But since $s_j^* \notin \mathcal{B}_\gamma$, by Remark 10 the total error $\left\|\boldsymbol{\varepsilon}_{s_j^*}\right\|$ cannot be too large. This means that the first condition can happen only for a small number $o(|S_j|)$ of nodes. For the second condition the argument is similar. Let v^* be an active node such that $v^* \notin S_j$, but $v^* \in \mathsf{Cand}(u)$. Since $v^* \in \mathsf{Cand}(u)$, we know that $\ell_u \approx \ell_{v^*}$ and that the error $\varepsilon_{(v^*,u)}$ is quite large: $\varepsilon_{(v^*,u)} \geq t_u$. However, by (A1) $v^* \notin \mathcal{B}_\gamma$, so $\|\boldsymbol{\varepsilon}_{v^*}\|$ cannot be too large. Therefore v^* can be a candidate for a limited number of $u \in S'_j$. Summing over all active nodes and using the upper bound (A3) is sufficient to show that condition 2 can happen only for a small number $o(|S_j|)$ of nodes. □

3.5 Proof of the Main Result

In this section we bring everything together and prove Theorem 1:

Proof (Proof of Theorem 1)
Number of Rounds. Firstly, let us look at the number of rounds of our Algorithm 4. We know that the Averaging phase of the algorithm takes $T = \Theta\left(\frac{\log(n)}{\lambda_{k+1}}\right)$ rounds. The Small Detection phase does not require any extra rounds of communication. For the Large Detection phase, we have again T rounds. This brings the total number of rounds to

$$2 \cdot T = \Theta\left(\frac{\log(n)}{\lambda_{k+1}}\right).$$

Clustering Guarantee. Secondly, we will look at the clustering guarantees of Algorithm 4. Let S_j be some cluster of G. If $|S_j| \leq \log(n)$, then by Lemma 8 we know that all nodes of S_j will choose the same label, that is the minimum ID among nodes in S_j. If $|S_j| > \log(n)$, by Lemma 13 it follows that with probability at least 0.9 all but $o(|S_j|)$ nodes will determine the same label that is the $\mathsf{ID}(v)$ for some $v \in S_j$.

Communication Cost. Let us first look at the cost for the Averaging phase. In any round $i \leq T$, every node u has to send to all its neighbours the values $\{x_{i-1}^v(u)\}$. This results in a total communication cost of $\text{cost}_{\text{Avg}} = O(T \cdot n \cdot m)$. Again, for the Small Detection phase there is no cost attached. While for the Large Detection phase, by Lemma 13 we know that, with probability at least 0.9, the total number of active nodes is $n_a = O(k \log(n) \cdot \log(k \log(n)))$. By the design of Algorithm 3, every node u in the network will broadcast each active node at most once. Therefore the total communication cost in the Large Detection phase is $\text{cost}_{\text{LD}} = O(m \cdot n_a) = O(mk \cdot \log(k))$. This gives in total a communication

cost of $O(\text{cost}_{\text{Avg}} + \text{cost}_{\text{LD}}) = O(T \cdot n \cdot m)$. We remark that, while in general the number of edges could be $m = \Theta(n^2)$, we can first apply the sampling scheme in [23] to sparsify our network and then run our algorithm. The sparsification ensures that the structure of the clusters, the degree sequence and the parameter $\Upsilon_G(k)$ are preserved up to a small constant factor and the resulting number of edges becomes $m = \widetilde{O}(n)$. Thus the final communication cost can be expressed as $\widetilde{O}(T \cdot n^2)$. □

Acknowledgements. I would like to thank my supervisor Dr. He Sun for helpful discussion and comments on improving the presentation of this paper.

References

1. Allen-Zhu, Z., Lattanzi, S., Mirrokni, V.S.: A local algorithm for finding well-connected clusters. In: 30th International Conference on Machine Learning (ICML 2013), pp. 396–404 (2013)
2. Becchetti, L., Clementi, A.E.F., Manurangsi, P., Natale, E., Pasquale, F., Raghavendra, P., Trevisan, L.: Average whenever you meet: Opportunistic protocols for community detection. In: 26th European Symposium on Algorithms (ESA'18). LIPIcs, vol. 112, pp. 7:1–7:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.ESA.2018.7
3. Becchetti, L., Clementi, A.E.F., Natale, E., Pasquale, F., Trevisan, L.: Find your place: simple distributed algorithms for community detection. SIAM J. Comput. **49**(4), 821–864 (2020). https://doi.org/10.1137/19M1243026
4. Becchetti, L., Cruciani, E., Pasquale, F., Rizzo, S.: Step-by-step community detection in volume-regular graphs. Theoret. Comput. Sci. **847**, 49–67 (2020). https://doi.org/10.1016/j.tcs.2020.09.036
5. Boyd, S.P., Ghosh, A., Prabhakar, B., Shah, D.: Randomized gossip algorithms. IEEE Trans. Inf. Theory **52**(6), 2508–2530 (2006). https://doi.org/10.1109/TIT.2006.874516
6. Chang, Y., Saranurak, T.: Improved distributed expander decomposition and nearly optimal triangle enumeration. In: Robinson, P., Ellen, F. (eds.) Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, 29 July–2 August 2019, pp. 66–73. ACM (2019). https://doi.org/10.1145/3293611.3331618
7. Chang, Y., Saranurak, T.: Deterministic distributed expander decomposition and routing with applications in distributed derandomization. In: 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2020), pp. 377–388. IEEE (2020). https://doi.org/10.1109/FOCS46700.2020.00043
8. Chen, J., Sun, H., Woodruff, D.P., Zhang, Q.: Communication-optimal distributed clustering. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) 29th Advances in Neural Information Processing Systems (NeurIPS 2016), pp. 3720–3728 (2016)
9. Czumaj, A., Peng, P., Sohler, C.: Testing cluster structure of graphs. In: 47th Annual ACM Symposium on Theory of Computing (STOC 2015), pp. 723–732. ACM (2015). https://doi.org/10.1145/2746539.2746618
10. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)
11. Georgakopoulos, A., Haslegrave, J., Sauerwald, T., Sylvester, J.: The power of two choices for random walks. arXiv preprint arXiv:1911.05170 (2019)

12. Hui, P., Yoneki, E., Chan, S.Y., Crowcroft, J.: Distributed community detection in delay tolerant networks. In: Proceedings of 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture. Association for Computing Machinery (2007). https://doi.org/10.1145/1366919.1366929
13. Laenen, S., Sun, H.: Higher-order spectral clustering of directed graphs. In: 33rd Advances in Neural Information Processing Systems (NeurIPS 2020) (2020)
14. Lee, J.R., Gharan, S.O., Trevisan, L.: Multiway spectral partitioning and higher-order Cheeger inequalities. J. ACM **61**(6), 37:1–37:30 (2014). https://doi.org/10.1145/2665063
15. Li, A., Peng, P.: Community structures in classical network models. Internet Math. **7**(2), 81–106 (2011). https://doi.org/10.1080/15427951.2011.566458
16. von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007). https://doi.org/10.1007/s11222-007-9033-z
17. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: analysis and an algorithm. In: 14th Advances in Neural Information Processing Systems (NeurIPS 2021), pp. 849–856. MIT Press (2001)
18. Oveis Gharan, S., Trevisan, L.: Partitioning into expanders. In: 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014), pp. 1256–1266. SIAM (2014). https://doi.org/10.1137/1.9781611973402.93
19. Peng, R., Sun, H., Zanetti, L.: Partitioning well-clustered graphs: spectral clustering works!. SIAM J. Comput. **46**(2), 710–743 (2017). https://doi.org/10.1137/15M1047209
20. Sauerwald, T., Zanetti, L.: Random walks on dynamic graphs: mixing times, hitting times, and return probabilities. In: 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019). LIPIcs, vol. 132, pp. 93:1–93:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.93
21. Schaeffer, S.E.: Graph clustering. Comput. Sci. Rev. **1**(1), 27–64 (2007). https://doi.org/10.1016/j.cosrev.2007.05.001
22. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000). https://doi.org/10.1109/34.868688
23. Sun, H., Zanetti, L.: Distributed graph clustering and sparsification. ACM Trans. Parallel Comput. **6**(3), 17:1–17:23 (2019). https://doi.org/10.1145/3364208
24. Yang, W., Xu, H.: A divide and conquer framework for distributed graph clustering. In: 32nd International Conference on Machine Learning (ICML 2015). JMLR Workshop and Conference Proceedings, vol. 37, pp. 504–513 (2015). JMLR.org