# Distributed Graph Clustering and Sparsification

HE SUN, The University of Edinburgh
LUCA ZANETTI, The University of Cambridge

Graph clustering is a fundamental computational problem with a number of applications in algorithm design, machine learning, data mining, and analysis of social networks. Over the past decades, researchers have proposed a number of algorithmic design methods for graph clustering. Most of these methods, however, are based on complicated spectral techniques or convex optimisation, and cannot be directly applied for clustering many networks that occur in practice, whose information is often collected on different sites. Designing a simple and distributed clustering algorithm is of great interest, and has comprehensive applications for processing big datasets.

In this paper we present a simple and distributed algorithm for graph clustering: for a wide class of graphs that are characterised by a strong cluster-structure, our algorithm finishes in a poly-logarithmic number of rounds, and recovers a partition of the graph close to optimal. One of the main procedures behind our algorithm is a sampling scheme that, given a dense graph as input, produces a sparse subgraph that provably preserves the cluster-structure of the input. Compared with previous sparsification algorithms that require Laplacian solvers or involve combinatorial constructions, this procedure is easy to implement in a distributed setting and runs fast in practice.

CCS Concepts: •**Theory of computation** →**Graph algorithms analysis; Distributed algorithms; Random walks and Markov chains;**

Additional Key Words and Phrases: graph clustering, graph sparsification, distributed computing

**ACM Reference format:**
He Sun and Luca Zanetti. 201x. Distributed Graph Clustering and Sparsification. *ACM Trans. Parallel Comput.* x, 4, Article 00 (March 201x), 23 pages.
DOI: 0000001.0000001

## 1 INTRODUCTION

Analysis of large-scale networks has brought significant advances to our understanding of complex systems. One of the most relevant features of networks occurring in practice is their structure of clusters, i.e., an organisation of nodes into clusters such that nodes within the same cluster enjoy a higher degree of connectivity in contrast to nodes from different clusters. Graph clustering is an important research topic in many disciplines, including computer science, biology, and sociology. For instance, graph clustering is widely used in finding communities in social networks, webpages dealing with similar topics, and proteins having the same specific function within the cell in protein-protein interaction networks (Fortunato 2010). However, despite extensive studies on efficient methods for graph clustering, many approximation algorithms for this problem require advanced algorithm design techniques, e.g., spectral methods, or convex optimisation, which make the algorithms difficult to be implemented in the distributed setting, where graphs are allocated in sites which are physically remote. Designing a simple and distributed algorithm is of important interest in practice, and has received considerable attention in recent years (Chen et al. 2016; Hui et al. 2007; Yang and Xu 2015).

## 1.1 Structure of clusters

Let $G = (V, E, w)$ be an undirected graph with $n$ nodes and weight function $w : V \times V \longrightarrow \mathbb{R}_{\geqslant 0}$. For any set $S$, let the conductance of $S$ be

$$\phi_G(S) \triangleq \frac{w(S, V \setminus S)}{\mathrm{vol}(S)},$$

where $w(S, V \setminus S) \triangleq \sum_{u \in S} \sum_{v \in V \setminus S} w(u, v)$ is the total weight of edges between $S$ and $V \setminus S$, and $\mathrm{vol}(S) \triangleq \sum_{u \in S} \sum_{u \sim v} w(u, v)$ is the volume of $S$, where $u \sim v$ stands for the fact that there is an edge between $u$ and $v$. Intuitively, nodes in $S$ form a cluster if there are fewer connections between the nodes of $S$ to the nodes in $V \setminus S$, i.e., the value of $\phi_G(S)$ is small. We call subsets of nodes (i.e. *clusters*) $A_1, \ldots, A_k$ a *$k$-way partition* of $G$ if $A_i \neq \emptyset$ for all $1 \leqslant i \leqslant k$, $A_i \cap A_j = \emptyset$ for different $i$ and $j$, and $\bigcup_{i=1}^{k} A_i = V$. Moreover, we define the *$k$-way expansion constant* by

$$\rho(k) \triangleq \min_{\text{partition } A_1, \ldots, A_k} \max_{1 \leqslant i \leqslant k} \phi_G(A_i).$$

Computing the exact value of $\rho(k)$ is coNP-hard (Blum et al. 1981), and a sequence of results show that $\rho(k)$ can be approximated by algebraic quantities relating to the matrices representing $G$. For instance, Lee et al. (Lee et al. 2014) shows the following high-order Cheeger inequality:

$$\frac{\lambda_k}{2} \leqslant \rho(k) \leqslant O\left(k^2\right) \sqrt{\lambda_k}, \tag{1}$$

where $0 = \lambda_1 \leqslant \cdots \leqslant \lambda_n \leqslant 2$ are the eigenvalues of the normalised Laplacian matrix of $G$. By (1) we know that a large gap between $\lambda_{k+1}$ and $\rho(k)$ guarantees (i) existence of a $k$-way partition $S_1, \ldots S_k$ with bounded $\phi_G(S_i) \leqslant \rho(k)$, and (ii) any $(k+1)$-way partition $A_1, \ldots, A_{k+1}$ of $G$ contains a subset $A_i$ with much higher conductance $\rho(k+1) \geqslant \lambda_{k+1}/2$ compared with $\rho(k)$. Peng et al. (Peng et al. 2015) formalises this observation by defining the parameter

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

and shows that a suitable lower bound on $\Upsilon_G(k)$ implies that $G$ has $k$ well-defined clusters.

## 1.2 Our results

In this paper we study distributed graph clustering, and our algorithm is obtained by combining the following two results. Our first result is a sampling-based algorithm to obtain a sparse graph $H$ that approximately preserves the cluster-structure of the input graph $G$. In contrast to previous algorithms for constructing a spectral sparsifier, which usually involve complicated sampling processes, our algorithm samples edges based on the degrees of their two endpoints, and can be implemented in the distributed setting. We show that our sampling approximately preserves the cluster-structure of $G$. The approximation guarantees are summarised as follows:

Theorem 1.1. *There exists an algorithm that, given a graph $G = (V, E, w)$ with $k$ clusters as input, with probability greater than 0.99, computes a sparsifier $H = (V, F \subset E, \widetilde{w})$ with $|F| = O((1/\lambda_{k+1}) \cdot n \log n)$ edges such that the following holds:*

1. *It holds for any $1 \leqslant i \leqslant k$ that $\phi_H(S_i) = O(k \cdot \phi_G(S_i))$, where $S_1, \ldots, S_k$ are the optimal clusters in $G$ that achieves $\rho(k)$;*
2. *$\Upsilon_H(k) = \Omega(\Upsilon_G(k)/k)$.*

*Moreover, this algorithm can be implemented in $O(1)$ rounds in the distributed setting, and the total information exchanged among all nodes is $O((1/\lambda_{k+1}) \cdot n \log n)$ words.*

The first property of Theorem 1.1 shows that the conductance of each optimal cluster $S_i$ in $G$ is approximately preserved in $H$ up to a factor of $k$, therefore $S_i$ is a low-conductance subset in $H$. Notice that these $k$ clusters $S_1, \ldots, S_k$ might not form an optimal clustering in $H$ anymore, however this is not an issue since every cluster with low conductance in $H$ has large overlap with its optimal correspondence. Hence, any algorithm that recovers a clustering close to the optimal one in $H$ will recover a clustering close to the optimal one in $G$. The second property $\Upsilon_H(k) = \Omega(\Upsilon_G(k)/k)$ of Theorem 1.1 further ensures that the gap in $H$ is preserved as long as $\Upsilon_G(k) \gg k$. In addition, since $\lambda_{k+1}$ represents the inner-connectivity of the clusters (Oveis Gharan and Trevisan 2014), it is usually quite high: for most interesting cases we can assume $\lambda_{k+1} = \Omega\left(1/\mathrm{poly}(\log n)\right)$, which makes the total number of sampled edges nearly-linear in the number of nodes in $G$.

Our second result is a distributed algorithm to partition a graph $G$ that possesses a cluster-structure with clusters of balanced size. At a high level, the algorithm consists of three steps: the seeding, averaging, and query steps. (1) In the seeding step, each node becomes active with a certain probability. (2) In the averaging step, which consists of $T$ rounds, nodes repeatedly update their states based on the states of their neighbours from the previous round. Alternatively, one can view the seeding step as choosing a subset of nodes to initiate parallel random walks, and the average step as simulating these parallel random walks for $T$ steps. (3) In the query step, each node uses the information computed in the averaging step, i.e., the current state of the $T$-step parallel random walks started from the active nodes, to determine the label of the cluster it belongs to. The performance of our algorithm is summarised as follows.

Theorem 1.2. *There is a distributed algorithm that, given as input a graph $G = (V, E, w)$ with $n$ nodes, $m$ edges, and $k$ optimal clusters $S_1, \ldots, S_k$ with $\mathrm{vol}(S_i) \geqslant \beta\,\mathrm{vol}(V)$ for any $1 \leqslant i \leqslant k$ and*

$$\Upsilon_G(k) = \omega\left(k^2 \log^2 \frac{1}{\beta} + \log n \cdot \log \frac{1}{\beta}\right), \tag{2}$$

*finishes in*

$$T \triangleq \Theta\left(\frac{\log n}{\lambda_{k+1}}\right)$$

*rounds, and with probability greater than $0.99$ the following statements hold:*

(1) *Each node $v$ receives a label $\ell_v$ such that the total volume of misclassified nodes is $o(\mathrm{vol}(V))$, i.e., under a possible permutation of the labels $\sigma$, it holds that*

$$\mathrm{vol}\left(\bigcup_{i=1}^{k} \{v | v \in S_i \text{ and } \ell_v \neq \sigma(i)\}\right) = o(\mathrm{vol}(V));$$

(2) *The total information exchanged among these $n$ nodes, i.e., the message complexity, is $O\left(T \cdot m \cdot \frac{1}{\beta} \log \frac{1}{\beta}\right)$ words.*

As a direct application of Theorems 1.1 and 1.2, let us look at a graph $G$ that consists of $k = O(1)$ expander graphs of almost balanced size connected by sparse cuts. By first applying the sparsification algorithm from Theorem 1.1, we obtain a sparse subgraph $H$ of $G$ that has a similar cluster-structure to $G$, and this graph $H$ is obtained with total communication cost $O(n \cdot \mathrm{poly}\log n)$ words. Then, we apply the distributed clustering algorithm (Theorem 1.2) on $H$, which has $O(n \cdot \mathrm{poly}\log n)$ edges. The distributed clustering algorithm finishes in $O(\log n)$ rounds, has total communication cost $O(n \cdot \mathrm{poly}\log n)$ words, and the volume of the misclassified nodes is $o(\mathrm{vol}(V))$. Notice that the communication cost of the two algorithms together is $O(n \cdot \mathrm{poly}\log n)$ words, which is sublinear in $m$ for a dense input graph.

### 1.3 Related work

There is a large amount of literature on graph clustering, and our work is most related to efficient algorithms for graph clustering under different formulations of clusters. Oveis Gharan and Trevisan (Oveis Gharan and Trevisan 2014) formulates the notion of clusters with respect to the inner and outer conductance: a cluster $S$ should have low outer conductance, and the conductance of the induced subgraph by $S$ should be high. Under some assumption about the gap between $\lambda_{k+1}$ and $\lambda_k$, they present a polynomial-time algorithm which finds a $k$-way partition $\{A_i\}_{i=1}^{k}$ that satisfies the inner and outer conductance condition. Allen-Zhu et al. (Allen-Zhu et al. 2013) studies graph clustering with a gap assumption similar to ours, and presents a local algorithm with better approximation guarantee under the gap assumption. However, the setup of our algorithms differs significantly from most local graph clustering algorithms (Allen-Zhu et al. 2013; Gharan and Trevisan 2012; Spielman and Teng 2013) for the following reasons: (1) One needs to run a local algorithm $k$ times in order to find $k$ clusters. However, as the output of each execution of a local algorithm only returns an *approximate* cluster, the approximation ratio of the final output cluster might not be guaranteed when the value of $k$ is large. (2) For many instances, our algorithm requires only a poly-logarithmic number of rounds, while local algorithms run in time proportional to the volume of the output set. It is unclear how these algorithms could finish in a poly-logarithmic number of rounds, even if we were able to implement them in the distributed setting.

Becchetti et al. (Becchetti et al. 2017) studies a distributed process to partition an almost-regular graph into clusters, and their analysis focuses mostly on graphs generated randomly from stochastic block models. In contrast to ours, their algorithm requires every node to exchange information with all of its neighbours in each round, which results in significantly higher communication cost. Moreover, the design and analysis of our algorithm succeeds to overcome their regularity constraint by an alternative averaging rule. More recently, Becchetti et al. (Becchetti et al. 2018) studies the same problem in the asynchronous setting, which is more general than our synchronous setting. Their algorithm requires, again, the graph to be almost regular and to have a strong community-structure (this is satisfied by graphs sampled from the stochastic block model).

We notice that the distributed algorithm presented in Kempe and McSherry (Kempe and McSherry 2004) for computing the top $k$ eigenvectors of the adjacency matrix of a graph can be applied for graph clustering. However, their algorithm is more involved than ours. Moreover, for an input graph $G$ of $n$ nodes, the number of rounds required in their algorithm is proportional to the mixing time of a random walk in $G$. For a graph consisting of multiple expanders connected by very few edges, their algorithm requires $O(\text{poly}(n))$ rounds, which is much higher than $O(\text{poly} \log n)$ rounds needed for our algorithm.

Another line of research closely related to our work is graph sparsification, including both cut sparsification (Benczúr and Karger 1996) and spectral sparsification (Batson et al. 2012; Lee and Sun 2015, 2017; Spielman and Srivastava 2011; Spielman and Teng 2011). The constructions of both cut and spectral sparsifiers, however, are quite complicated or require solving Laplacian systems, while our algorithm is simply based on sampling and easy to implement. The idea of using sparsification to reduce the communication complexity for clustering a graph in the distributed setting is first proposed by (Chen et al. 2016). They assume the graph is distributed across multiple servers, while our work considers more extreme distributed settings: each node of the graph is a computational unit. Our algorithms, however, work in their distributed model as well. We emphasise that the sparsification schemes of (Chen et al. 2016) require the computation of effective resistances, which is very expensive in practice, while our scheme is much simpler and faster.

### 1.4 Organisation

The remaining part of the paper is organised as follows: Section 2 lists the notations used in the paper. We present and analyse the sparsification algorithm in Section 3, and prove Theorem 1.1. Section 4 is to present the distributed algorithm for graph clustering, which corresponds to Theorem 1.2. We report the experimental results of our sparsification algorithm in Section 5.

## 2 PRELIMINARIES

Let $G = (V, E, w)$ be an undirected weighted graph with $n$ nodes and weight function $w : E \longrightarrow \mathbb{R}_{\geqslant 0}$. For any node $u$, the degree $d_u$ of $u$ is defined as $d_u \triangleq \sum_{u \sim v} w(u, v)$, where we write $u \sim v$ if $\{u, v\} \in E[G]$. For any set $S \subseteq V$, the volume of $S$ is defined by $\mathrm{vol}_G(S) \triangleq \sum_{v \in S} d_v$. The (normalised) indicator vector of a set $S \subset V$ is defined by $\chi_S \in \mathbb{R}^n$, where $\chi_S(v) = \sqrt{d_v/\mathrm{vol}(S)}$ if $v \in S$, and $\chi_S(v) = 0$ otherwise.

We work with algebraic objects related to $G$. Let $\mathbf{A}_G$ be the adjacency matrix of $G$ defined by $(\mathbf{A}_G)_{u,v} = w(u, v)$ if $\{u, v\} \in E(G)$, and $(\mathbf{A}_G)_{u,v} = 0$ otherwise. The degree matrix $\mathbf{D}_G$ of $G$ is a diagonal matrix defined by $(\mathbf{D}_G)_{u,u} = d_u$, and the normalised Laplacian of $G$ is defined by $\mathcal{L}_G \triangleq \mathbf{I} - \mathbf{D}_G^{-1/2} \mathbf{A}_G \mathbf{D}_G^{-1/2}$. Alternatively, we can write the normalised Laplacian with respect to the indicator vectors of nodes: for each node $v$, we define an indicator vector $\chi_v \in \mathbb{R}^n$ by $\chi_v(u) = 1/\sqrt{d_v}$ if $u = v$, and $\chi_v(u) = 0$ otherwise. We further define $b_e \triangleq \chi_u - \chi_v$ for each edge $e = \{u, v\}$, where the orientation of $e$ is chosen arbitrarily. Then, we can write $\mathcal{L}_G = \sum_{e=\{u,v\} \in E} w(u, v) \cdot b_e b_e^{\intercal}$. We always use $0 = \lambda_1 \leqslant \cdots \leqslant \lambda_n \leqslant 2$ to express the eigenvalues of $\mathcal{L}_G$, with the corresponding orthonormal eigenvectors $f_1, \ldots, f_n$. With a slight abuse of notation, we use $\mathcal{L}_G^{-1}$ for the pseudoinverse of $\mathcal{L}_G$, i.e., $\mathcal{L}_G^{-1} \triangleq \sum_{i=2}^{n} \frac{1}{\lambda_i} f_i f_i^{\intercal}$. When $G$ is connected, it holds that $\lambda_2 > 0$ and the matrix $\mathcal{L}_G^{-1}$ is well-defined. Sometimes we drop the subscript $G$ when it is clear from the context.

Remember that the Euclidean norm of any vector $x \in \mathbb{R}^n$ is defined as $\|x\| \triangleq \sqrt{\sum_{i=1}^{n} x_i^2}$, and the spectral norm of any matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ is defined as

$$\|\mathbf{M}\| \triangleq \max_{x \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{\|\mathbf{M}x\|}{\|x\|}.$$

## 3 CLUSTER-PRESERVING SPARSIFIERS

In this section we present an algorithm for constructing a cluster-preserving sparsifier that can be easily implemented in the distributed setting. Our algorithm is based on sampling edges with respect to the degrees of their endpoints, which was originally introduced in (Spielman and Teng 2011) as a way to construct spectral sparsifiers for graphs with *high* spectral expansion. To sketch the intuition behind our algorithm, let us look at the following toy example illustrated in Figure 1, i.e., the graph $G$ consisting of two complete graphs of $n$ nodes connected by a single edge. It is easy to see that, when we sample $O(n \log n)$ edges uniformly at random from $G$ to form a graph $H$, with high probability the middle edge will not be sampled and $H$ will consist of two isolated expander graphs, each of which has constant spectral expansion. Although our sampled graph $H$ does not preserve the spectral and cut structure of $G$, it does preserve its cluster-structure: every reasonable clustering algorithm will recover these two disjoint components of $H$, which correspond exactly to the two clusters in $G$. We will show that this sampling scheme can be generalised, and sampling every edge $e = \{u, v\}$ with probability depending only on $d_u$ and $d_v$ suffices to construct a sparse subgraph that preserves the cluster-structure of the original graph.

### 3.1 Algorithm description

In our algorithm every node $u$ checks every edge $e = \{u, v\}$ adjacent to $u$ itself, and samples edge $e$ with probability

$$p_u(v) \triangleq \min\left\{ w(u, v) \cdot \frac{C \cdot \log n}{d_u \cdot \lambda_{k+1}}, 1 \right\} \tag{3}$$

for a large enough constant $C \in \mathbb{R}_{\geqslant 0}$. The algorithm uses a set $F$ to maintain all the sampled edges, where $F$ is initially set to be empty. Finally, the algorithm returns a weighted graph $H = (V, F, w_H)$, where the weight

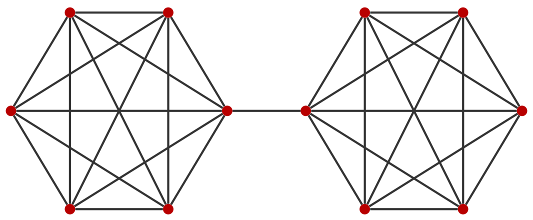


Fig. 1. The graph $G$ consists of two complete subgraphs of $n$ nodes connected by an edge. It is easy to see that sampling $O(n \log n)$ edges uniformly at random suffices to construct a subgraph having the same cluster-structure of $G$.

$w_H(u,v)$ of every sampled edge $e = \{u,v\} \in F$ is defined as

$$w_H(u,v) \triangleq \frac{w(u,v)}{p_e},$$

and

$$p_e \triangleq p_u(v) + p_v(u) - p_u(v) \cdot p_v(u)$$

is the probability that $e = \{u,v\}$ is sampled by at least one of its endpoints. Notice that our algorithm can be easily implemented in a distributed setting: any node $u$ chooses to retain (or not) an edge $u \sim v$ independently from any other node, and communication between $u$ and $v$ is needed only if $u \sim v$ is sampled by one of its two endpoints. Therefore, the total communication cost of the algorithm is linear in the number of edges in $H$.

## 3.2 Analysis of the algorithm

Now we analyse the algorithm, and prove Theorem 1.1. At a high level, our proof consists of the following two steps:

(1) We show that the conductance of $S_1, \ldots, S_k$ are approximated preserved in $H$, i.e.,

$$\phi_H(S_i) = O\left(k \cdot \phi_G(S_i)\right) \text{ for any } i = 1, \ldots, k. \tag{4}$$

(2) We analyse the intra-connectivity of the clusters in the returned graph $H$, and prove that the top $n-k$ eigenspaces of $\mathcal{L}_G$ are preserved in $\mathcal{L}_H$. This implies that $\lambda_{k+1}(\mathcal{L}_H) = \Omega(\lambda_{k+1}(\mathcal{L}_G))$.

Combining these two steps, we will prove that $\Upsilon_H(k) = \Omega(\Upsilon_G(k)/k)$, which proves the approximation guarantees of Theorem 1.1. The total number of edges in $H$ follows by the sampling scheme of our algorithm. We first recall the following concentration inequalities that will be used in our proof.

Lemma 3.1 (Bernstein's inequality (Chung and Lu 2006)). *Let $X_1, \ldots, X_n$ be independent random variables such that $|X_i| \leqslant M$ for any $i \in \{1, \ldots, n\}$. Let $X = \sum_{i=1}^n X_i$ and let $R = \sum_{i=1}^n \mathbf{E}[X_i^2]$. Then, it holds that*

$$\mathbf{P}\left[\, |X - \mathbf{E}[X]| \geqslant t \,\right] \leqslant 2\exp\left(-\frac{t^2}{2(R + Mt/3)}\right).$$

Lemma 3.2 (Matrix Chernoff Bound, (Tropp 2012)). *Consider a finite sequence $\{X_i\}$ of independent, random, PSD matrices of dimension $d$ that satisfy $\|X_i\| \leqslant R$. Let $\mu_{\min} \triangleq \lambda_{\min}\left(\mathbf{E}\left[\sum_i X_i\right]\right)$ and $\mu_{\max} \triangleq \lambda_{\max}\left(\mathbf{E}\left[\sum_i X_i\right]\right)$. Then* *it holds that*

$$\mathbf{P}\left[\lambda_{\min}\left(\sum_i X_i\right) \leqslant (1-\delta)\mu_{\min}\right] \leqslant d \cdot \left(\frac{\mathrm{e}^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu_{\min}/R} \quad \textit{for } \delta \in [0,1], \ \textit{and}$$

$$\mathbf{P}\left[\lambda_{\max}\left(\sum_i X_i\right) \geqslant (1+\delta)\mu_{\max}\right] \leqslant d \cdot \left(\frac{\mathrm{e}^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu_{\max}/R} \quad \textit{for } \delta \geqslant 0.$$

Proof of Theorem 1.1. We first analyse the size of $F$. Since

$$\sum_{u \in V} \sum_{e=\{u,v\}} w(u,v) \cdot \frac{C \log n}{d_u \cdot \lambda_{k+1}} = O\left(\frac{n \log n}{\lambda_{k+1}}\right),$$

it holds by Markov inequality that the number of edges $e = \{u,v\}$ with $p_u(v) \geqslant 1$ is $O((1/\lambda_{k+1}) \cdot n \log n)$. Without loss of generality, we assume that these edges are in $F$, and in the remaining part of the proof we assume it holds for any edge $u \sim v$ that

$$w(u,v) \cdot \frac{C \cdot \log n}{d_u \cdot \lambda_{k+1}} < 1.$$

Then, the expected number of edges in $H$ equals to

$$\sum_{e=\{u,v\}} p_e \leqslant \sum_{e=\{u,v\}} p_u(v) + p_v(u) = \sum_{e=\{u,v\}} \left(w(u,v) \cdot \frac{C \cdot \log n}{d_u \cdot \lambda_{k+1}} + w(u,v) \cdot \frac{C \cdot \log n}{d_v \cdot \lambda_{k+1}}\right) = O\left(\frac{n \log n}{\lambda_{k+1}}\right),$$

and by Markov inequality it holds with constant probability that $|F| = O\left((1/\lambda_{k+1}) \cdot n \log n\right)$. This implies that the total information exchanged among all nodes is $|F| = O\left((1/\lambda_{k+1}) \cdot n \log n\right)$. The $O(1)$ rounds needed for the algorithm is simply by the algorithm description.

Now we will show that the degrees of nodes in $H$ are approximately preserved with high probability. Let $u$ be an arbitrary node of $G$. For any edge $e = \{u,v\}$, we define random variable $Y_e$ by

$$Y_e = \begin{cases} \frac{w(u,v)}{p_e} & \text{with probability } p_e, \\ 0 & \text{otherwise.} \end{cases}$$

We further define $Z_u = \sum_{e=\{u,v\}} Y_e$. Hence, it holds that

$$\mathbf{E}[Z_u] = \sum_{e=\{u,v\}} \mathbf{E}[Y_e] = \sum_{e=\{u,v\}} p_e \cdot \frac{w(u,v)}{p_e} = \sum_{e=\{u,v\}} w(u,v) = d_u.$$

For the second moment, we have that

$$R_u \triangleq \sum_{e=\{u,v\}} \mathbf{E}\left[Y_e^2\right] = \sum_{e=\{u,v\}} p_e \cdot \left(\frac{w(u,v)}{p_e}\right)^2 = \sum_{e=\{u,v\}} \frac{(w(u,v))^2}{p_e}. \tag{5}$$

Since $p_e = p_u(v) + p_v(u) - p_u(v)p_v(u) \geqslant p_u(v)$, we can rewrite (5) as

$$R_u = \sum_{e=\{u,v\}} \mathbf{E}[Y_e^2] \leqslant \sum_{e=\{u,v\}} \frac{(w(u,v))^2}{p_u(v)} = \sum_{e=\{u,v\}} \frac{(w(u,v))^2 \cdot d_u \cdot \lambda_{k+1}}{w(u,v) \cdot C \cdot \log n} = \frac{d_u^2 \cdot \lambda_{k+1}}{C \cdot \log n}.$$

On the other hand, we have for any $e = \{u,v\}$ that

$$0 \leqslant \frac{w(u,v)}{p_e} \leqslant \frac{w(u,v)}{p_u(v)} = \frac{d_u \cdot \lambda_{k+1}}{C \cdot \log n},$$

by applying Bernstein's inequality (Lemma 3.1) we have that

$$\begin{aligned}\mathbf{P}\left[|d_H(u)-d_u| \geqslant \frac{1}{2}\cdot d_u\right] &= \mathbf{P}\left[|Z_u - \mathbf{E}[Z_u]| \geqslant \frac{1}{2}\cdot \mathbf{E}[Z_u]\right]\\ &\leqslant 2\cdot\exp\left(-\frac{d_u^2/8}{\frac{d_u^2\cdot\lambda_{k+1}}{C\log n}+\frac{1}{6}\cdot\frac{d_u^2\lambda_{k+1}}{C\log n}}\right)\\ &= 2\cdot\exp\left(-\frac{C\log n/8}{\lambda_{k+1}+\lambda_{k+1}/6}\right)\\ &= o(1/n).\end{aligned}$$

Hence, it holds by the union bound that, with high probability, the degree of all the nodes in $H$ are approximately preserved up to a constant factor. We assume that this event occurs in the rest of the proof. This implies that $\mathrm{vol}_H(S) = \Theta(\mathrm{vol}_G(S))$ for any subset $S \subseteq V$.

We continue to show that $\phi_H(S_i) = O(k\cdot\phi_G(S_i))$ for any $1 \leqslant i \leqslant k$, where $S_1,\ldots,S_k$ form an optimal clustering in $G$. By the definition of $Y_e$, it holds for any $1 \leqslant i \leqslant k$ that

$$\mathbf{E}\left[\, w_H(S_i, V\setminus S_i)\,\right] = \mathbf{E}\left[\sum_{\substack{e=\{u,v\},\\ u\in S_i, v\notin S_i}} Y_e\right] = \sum_{\substack{e=\{u,v\},\\ u\in S_i, v\notin S_i}} p_e\cdot\frac{w(u,v)}{p_e} = w(S_i, V\setminus S_i).$$

Hence, by Markov's inequality and the union bound, with constant probability it holds for all $i = 1,\ldots,k$ that

$$w_H(S_i, V\setminus S_i) = O\left(k\cdot w(S_i, V\setminus S_i)\right), \tag{6}$$

i.e., the cut value between $S_i$ and $V\setminus S_i$ is approximately preserved for any $1\leqslant i\leqslant k$. Therefore, it holds with constant probability that

$$\rho_H(k) \leqslant \max_{1\leqslant i\leqslant k}\phi_H(S_i) = \max_{1\leqslant i\leqslant k} O\left(k\cdot\phi_G(S_i)\right) = O(k\cdot\rho_G(k)).$$

Secondly, we show that the cluster-structure of $G$ is approximately preserved in $H$. Let $\overline{\mathcal{L}}_G$ be the projection of $\mathcal{L}_G$ on its top $n-k$ eigenspaces, and $\overline{\mathcal{L}}_G$ can be written as

$$\overline{\mathcal{L}}_G = \sum_{i=k+1}^{n}\lambda_i f_i f_i^{\intercal}.$$

With a slight abuse of notation we call $\overline{\mathcal{L}}_G^{-1/2}$ the square root of the pseudoinverse of $\overline{\mathcal{L}}_G$, i.e.,

$$\overline{\mathcal{L}}_G^{-1/2} = \sum_{i=k+1}^{n}(\lambda_i)^{-1/2} f_i f_i^{\intercal}.$$

Analogously, we call $\overline{\mathcal{I}}$ the projection on $\mathrm{span}\{f_{k+1},\ldots,f_n\}$, i.e.,

$$\overline{\mathcal{I}} = \sum_{i=k+1}^{n} f_i f_i^{\intercal}.$$

We will prove that the top $n-k$ eigenspaces of $\mathcal{L}_G$ are preserved, which implies that $\lambda_{k+1}(\mathcal{L}_G) = \Theta(\lambda_{k+1}(\mathcal{L}_H))$. To prove this, we recall that the probability that every edge $e=\{u,v\}$ is sampled in $H$ is

$$p_e = p_u(v) + p_v(u) - p_u(v)\cdot p_v(u),$$

and it holds that $\frac{1}{2}(p_u(v) + p_v(u)) \leqslant p_e \leqslant p_u(v) + p_v(u)$. Now for each edge $e = \{u, v\}$ of $G$ we define a random matrix $X_e \in \mathbb{R}^{n\times n}$ by

$$X_e = \begin{cases} w_H(u,v) \cdot \overline{\mathcal{L}_G}^{-1/2} b_e b_e^\intercal \overline{\mathcal{L}_G}^{-1/2} & \text{if } e = \{u,v\} \text{ is sampled by the algorithm,} \\ \mathbf{0} & \text{otherwise,} \end{cases}$$

where we recall that $b_e = \chi_u - \chi_v$ for edge $e = \{u, v\}$. Notice that

$$\sum_{e\in E[G]} X_e = \sum_{\text{sampled edges } e=\{u,v\}} w_H(u,v) \cdot \overline{\mathcal{L}_G}^{-1/2} b_e b_e^\intercal \overline{\mathcal{L}_G}^{-1/2} = \overline{\mathcal{L}_G}^{-1/2} \mathcal{L}'_H \overline{\mathcal{L}_G}^{-1/2},$$

where

$$\mathcal{L}'_H = \sum_{\text{sampled edges } e=\{u,v\}} w_H(u,v) \cdot b_e b_e^\intercal$$

is essentially the Laplacian matrix of $H$ but is normalised with respect to the degrees of the nodes in the original graph $G$, i.e., $\mathcal{L}'_H = D_G^{-1} D_H - D_G^{-1/2} A_H D_G^{-1/2}$. We will prove that, with high probability, the top $n-k$ eigenspaces of $\mathcal{L}'_H$ and $\mathcal{L}_G$ are approximately the same. Later we will show the same holds for $\mathcal{L}_H$ and $\mathcal{L}'_H$, which implies that $\lambda_{k+1}(\mathcal{L}'_H) = \Omega(\lambda_{k+1}(\mathcal{L}_G))$.

We start looking at the first moment of the expression above:

$$\begin{aligned}\mathbf{E}\left[\sum_{e\in E} X_e\right] &= \sum_{e=\{u,v\}\in E[G]} p_e \cdot w_H(u,v) \cdot \overline{\mathcal{L}_G}^{-1/2} b_e b_e^\intercal \overline{\mathcal{L}_G}^{-1/2} \\ &= \sum_{e=\{u,v\}\in E[G]} p_e \cdot \frac{w(u,v)}{p_e} \cdot \overline{\mathcal{L}_G}^{-1/2} b_e b_e^\intercal \overline{\mathcal{L}_G}^{-1/2} \\ &= \overline{\mathcal{L}_G}^{-1/2} \mathcal{L}_G \overline{\mathcal{L}_G}^{-1/2} = \overline{\mathcal{I}}.\end{aligned}$$

Moreover, for any sampled $e = \{u, v\} \in E$ we have that

$$\begin{aligned}\|X_e\| &\leqslant w_H(u,v) \cdot b_e^\intercal \overline{\mathcal{L}_G}^{-1/2} \overline{\mathcal{L}_G}^{-1/2} b_e = \frac{w(u,v)}{p_e} \cdot b_e^\intercal \overline{\mathcal{L}_G}^{-1} b_e \leqslant \frac{w(u,v)}{p_e} \cdot \frac{1}{\lambda_{k+1}} \cdot \|b_e\|^2 \\ &\leqslant \frac{2\lambda_{k+1}}{C \cdot \log n \cdot \left(\frac{1}{d_u} + \frac{1}{d_v}\right)} \cdot \frac{1}{\lambda_{k+1}} \left(\frac{1}{d_u} + \frac{1}{d_v}\right) \leqslant \frac{2}{C \log n},\end{aligned}$$

where the second inequality follows by the min-max theorem of eigenvalues. Now we apply the matrix Chernoff bound (Lemma 3.2) to analyse the eigenvalues of $\sum_{e\in E} X_e$, and build a connection between $\lambda_{k+1}(\mathcal{L}'_H)$ and $\lambda_{k+1}(\mathcal{L}_G)$. By setting the parameters of Lemma 3.2 by $\mu_{\max} = \lambda_{\max}\left(\mathbf{E}\left[\sum_{e\in E[G]} X_e\right]\right) = \lambda_{\max}\left(\overline{\mathcal{I}}\right) = 1$, $R = 2/(C \cdot \log n)$ and $\delta = 1/2$, we have that

$$\mathbf{P}\left[\lambda_{\max}\left(\sum_{e\in E[G]} X_e\right) \geqslant 3/2\right] \leqslant n \cdot \left(\frac{e^{1/2}}{(1+1/2)^{3/2}}\right)^{C\log n/2} = O\left(1/n^c\right)$$

for some constant $c$. This gives us that

$$\mathbf{P}\left[\lambda_{\max}\left(\sum_{e\in E[G]} X_e\right) \leqslant 3/2\right] = 1 - O(1/n^c). \tag{7}$$

On the other side, since our goal is to analyse $\lambda_{k+1}(\mathcal{L}'_H)$ with respect to $\lambda_{k+1}(\mathcal{L}_G)$, it suffices to work with the top $(n-k)$ eigenspace of $\mathcal{L}_G$. Since $\mathbf{E}\left[\sum_{e\in E} X_e\right] = \overline{\mathcal{I}}$, we can assume without loss of generality that $\mu_{\min} = 1$. [1] Hence, by setting $R = 2/(C\cdot\log n)$ and $\delta = 1/2$, we have that

$$\mathbf{P}\left[\lambda_{\min}\left(\sum_{e\in E[G]} X_e\right) \leqslant 1/2\right] = n\cdot\left(\frac{\mathrm{e}^{-1/2}}{(1/2)^{1/2}}\right)^{C\log n/2} = O\left(1/n^c\right)$$

for some constant $c$. This gives us that

$$\mathbf{P}\left[\lambda_{\min}\left(\sum_{e\in E[G]} X_e\right) > 1/2\right] = 1 - O(1/n^c). \tag{8}$$

Combining (7), (8), and the fact of $\sum_{e\in E[G]} X_e = \overline{\mathcal{L}_G}^{-1/2}\mathcal{L}'_H\overline{\mathcal{L}_G}^{-1/2}$, with probability $1 - O\left(1/n^c\right)$ it holds for any non-zero $x\in\mathbb{R}^n$ in the space spanned by $f_{k+1},\ldots,f_n$ that

$$\frac{x^\intercal\overline{\mathcal{L}_G}^{-1/2}\mathcal{L}'_H\overline{\mathcal{L}_G}^{-1/2}x}{x^\intercal x} \in (1/2, 3/2)\,. \tag{9}$$

By setting $y = \overline{\mathcal{L}_G}^{-1/2}x$, we can rewrite (9) as

$$\frac{y^\intercal\mathcal{L}'_H y}{y^\intercal\overline{\mathcal{L}_G}^{1/2}\overline{\mathcal{L}_G}^{1/2}y} = \frac{y^\intercal\mathcal{L}'_H y}{y^\intercal\overline{\mathcal{L}_G}y} = \frac{y^\intercal\mathcal{L}'_H y}{y^\intercal y}\frac{y^\intercal y}{y^\intercal\overline{\mathcal{L}_G}y} \in (1/2, 3/2).$$

Since $\dim(\mathrm{span}\{f_{k+1},\ldots,f_n\}) = n-k$, we have just proved there exist $n-k$ orthogonal vectors whose Rayleigh quotient with respect to $\mathcal{L}'_H$ is $\Omega(\lambda_{k+1}(\mathcal{L}_G))$. By the Courant-Fischer Theorem, we have

$$\lambda_{k+1}(\mathcal{L}'_H) \geqslant \frac{1}{2}\lambda_{k+1}(\mathcal{L}_G). \tag{10}$$

It remains to show that $\lambda_{k+1}(\mathcal{L}_H) = \Omega\left(\lambda_{k+1}(\mathcal{L}'_H)\right)$, which implies that $\lambda_{k+1}(\mathcal{L}_H) = \Omega\left(\lambda_{k+1}(\mathcal{L}_G)\right)$ by (10). By the definition of $\mathcal{L}'_H$, we have that $\mathcal{L}_H = D_H^{-1/2}D_G^{1/2}\mathcal{L}'_H D_G^{1/2}D_H^{-1/2}$. Therefore, for any $x\in\mathbb{R}^n$ and $y = D_G^{1/2}D_H^{-1/2}x$, it holds that

$$\frac{x^\intercal\mathcal{L}_H x}{x^\intercal x} = \frac{y^\intercal\mathcal{L}'_H y}{x^\intercal x} \geqslant \frac{1}{2}\cdot\frac{y^\intercal\mathcal{L}'_H y}{y^\intercal y}, \tag{11}$$

where the last equality follows from the fact that the degrees in $H$ and $G$ differ just by a constant multiplicative factor, and therefore,

$$y^\intercal y = \left(D_G^{1/2}D_H^{-1/2}x\right)^\intercal\left(D_G^{1/2}D_H^{-1/2}x\right) = x^\intercal D_G D_H^{-1}x \geqslant \frac{1}{2}\cdot x^\intercal x.$$

Finally, we show that (11) implies that $\lambda_{k+1}(\mathcal{L}_H) \geqslant (1/2)\cdot\lambda_{k+1}(\mathcal{L}'_H)$. To see this, let $S_1 \subseteq \mathbb{R}^n$ be a $(k+1)$-th dimensional subspace of $\mathbb{R}^n$ such that

$$\lambda_{k+1}(\mathcal{L}_H) = \max_{x\in S_1}\frac{x^\intercal\mathcal{L}_H x}{x^\intercal x}.$$

Let $S_2 = \left\{D_G^{1/2}D_H^{-1/2}x \colon x\in S_1\right\}$. Notice that since $D_G^{1/2}D^{-1/2}$ is full rank, $S_2$ has dimension $k+1$. Therefore,

$$\lambda_{k+1}(\mathcal{L}'_H) = \min_{S\colon\,\dim(S)=k+1}\max_{y\in S}\frac{y^\intercal\mathcal{L}'_H y}{y^\intercal y} \leqslant \max_{y\in S_2}\frac{y^\intercal\mathcal{L}'_H y}{y^\intercal y} \leqslant 2\max_{x\in S_1}\frac{x^\intercal\mathcal{L}_H x}{x^\intercal x} = 2\lambda_{k+1}(\mathcal{L}_H), \tag{12}$$

[1]To understand why this is the case, notice we could have defined $X_e$ as an $(n-k)$-dimensional operator. Then, $\overline{\mathcal{I}}$ would be just the identity in this $(n-k)$-dimensional space.

where the last inequality follows by (11). Combining (10) with (12) gives us that $\lambda_{k+1}(\mathcal{L}_H) = \Omega(\lambda_{k+1}(G))$, which implies $\Upsilon_H(k) = \Omega(\Upsilon_G(k)/k)$. □

## 4 DISTRIBUTED GRAPH CLUSTERING

In this section we present and analyse a distributed algorithm to partition a graph $G$ that possesses a cluster-structure with clusters of balanced size, and prove Theorem 1.2. Throughout the section, we assume that $S_1, \ldots, S_k$ are the optimal clusters satisfying $\mathrm{vol}(S_i) \geqslant \beta\,\mathrm{vol}(V)$ for any $1 \leqslant i \leqslant k$.

### 4.1 Algorithm description

Our algorithm consists of Seeding, Averaging, and Query steps, which are described as follows.

**The Seeding step:** The algorithm sets

$$\bar{s} = \Theta\left(\frac{1}{\beta} \cdot \log \frac{1}{\beta}\right),$$

and each node $v$ chooses to be *active* with probability $\bar{s} \cdot d_v/\mathrm{vol}(V)$. For simplicity, we assume that $v_1, \cdots, v_s$ are the active nodes, for some $s \in \mathbb{N}$. The algorithm associates each active node with a vector $x^{(0,i)} = \chi_{v_i}$, and these vectors $x^{(0,1)}, \ldots, x^{(0,s)}$ represent the initial state (round 0) of the graph, where each node $v$ only maintains the values $x^{(0,1)}(v), \ldots, x^{(0,s)}(v)$. Notice that the information about which nodes are active doesn't need to be broadcasted during the seeding step of the algorithm.

**The Averaging step:** This step consists of $T$ rounds, and in each round every node $v$ updates its state based on the states of its neighbours from the previous round. Namely, for any $1 \leqslant i \leqslant s$, the values $x^{(t,i)}(v)$ maintained by node $v$ in round $t$ are computed according to

$$x^{(t,i)}(v) = \frac{1}{2}x^{(t-1,i)}(v) + \frac{1}{2}\sum_{\{u,v\}\in E} \frac{w(u,v)}{\sqrt{d_u d_v}} x^{(t-1,i)}(u). \tag{13}$$

**The Query step:** Every node $v$ computes the label $\ell_v$ of the cluster that it belongs to by the formula

$$\ell_v = \min\left\{ i \;\middle|\; x^{(T,i)}(v) \geqslant \frac{\sqrt{d_v}}{2\beta\,\mathrm{vol}(V)} \right\}. \tag{14}$$

Notice that the execution of the algorithm requires each node to know certain parameters about the graph, including the number of nodes $n$, the volume of the graph $\mathrm{vol}(V)$, a bound $\beta$ on the size of the clusters, and the value of $T$. However, nodes do not need to know the exact values of these parameters, but only a reasonable approximation. Moreover, although the value of $T$ is application-dependent, for graphs with clusters that have strong intra-connectivity properties, we can set $T \approx \log n$ in practice.

### 4.2 Analysis of the algorithm

In this subsection we analyse the distributed clustering algorithm, and prove Theorem 1.2. Recall that we assume $G$ has an optimal clustering $S_1, \ldots, S_k$ with $\mathrm{vol}(S_i) \geqslant \beta\,\mathrm{vol}(V)$ for any $1 \leqslant i \leqslant k$, and $G$ satisfies the following gap assumption:

$$\Upsilon_G(k) = \omega\left(k^2 \log^2 \frac{1}{\beta} + \log n\right). \tag{15}$$

Before analysing the algorithm, we first describe some intuitions behind the proof. We use $x^{(t,1)}, \ldots, x^{(t,s)}$ to denote the configuration of the algorithm's execution in round $t$, and these vectors are updated according to (13). For the sake of intuition, we assume that $G$ is regular, and the vector $x^{(t,i)}$ corresponds to the probability distribution of a $t$-step lazy random walk in $G$. It is well-known that, after a sufficient number of rounds, the vectors $x^{(t,i)}$s are close to the uniform distribution, which would provide no information about the structure of clusters in a graph. However, the time $T = \Theta(\log n/\lambda_{k+1})$ can be informally viewed as the *local mixing time* of some cluster $S_i$: when a random walk starts with some $v \in S_i$ and the random walk does not leave $S_i$, the resulting distribution of this $T$-step random walk will be close to the uniform one supported on the node set of $S_i$. We will formalise the fact by showing that, when picking a starting vertex uniformly at random from $S_i$, with high probability the probability mass of a $T$-step random walk will be concentrated on $S_i$. In other words, after $T$ rounds, each vector $x^{(T,1)}, \ldots, x^{(T,s)}$ is almost uniform on one of the clusters, and close to zero everywhere else. This further implies that, once the algorithm ensures that there is at least one active node from every cluster, the query step will assign the same label to two nodes if and only if they belong to the same cluster. When $G$ is not regular, the averaging step defined in (13) can be viewed as a *power iteration method* to approximate ($k$ linearly independent combination of) the bottom eigenvectors of $\mathcal{L}_G$. We will show that these eigenvectors contain all the information needed to obtain a good partitioning of the graph.

We first analyse the properties of a $T$-step *single* random walk, and let $x^{(i)}$ be the distribution that the random walk reaches a specific vertex after $i$ steps. Let

$$\underline{\mathcal{I}} = \sum_{i=1}^{k} f_i f_i^{\intercal}$$

be the projection on the bottom $k$ eigenspaces. The following lemma shows that, when $x^{(0)}$ is the initial distribution of a random walk, after $T$ steps the distribution $x^{(T)}$ that the random walk reaches a specific node is close to $\underline{\mathcal{I}} x^{(0)}$.

Lemma 4.1. *For a large constant $c > 0$, it holds*

$$\left\| x^{(T)} - \underline{\mathcal{I}} x^{(0)} \right\| = O\left( \frac{\log n}{\Upsilon_G(k)} \cdot \left\| \underline{\mathcal{I}} x^{(0)} \right\| + n^{-c} \right).$$

Proof. We introduce the matrix $\mathbf{P}$ defined by

$$\mathbf{P} = \frac{1}{2} \cdot \mathbf{I} + \frac{1}{2} \cdot \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{I} - \frac{1}{2}\mathcal{L}_G.$$

Hence, by the description of the averaging step, it holds that $x^{(t)} = \mathbf{P}x^{(t-1)}$, which implies that $x^{(T)} = \mathbf{P}^T x^{(0)}$. To analyse the property of $x^{(T)}$, we study the spectral property of $\mathbf{P}^T$, which can be written as

$$\begin{aligned}\mathbf{P}^T = \left(\mathbf{I} - \frac{1}{2}\mathcal{L}_G\right)^T &= \sum_{i=1}^{n} \left(1 - \frac{\lambda_i}{2}\right)^T f_i f_i^{\intercal} \\ &= \sum_{i=1}^{k} \left(1 - \frac{\lambda_i}{2}\right)^T f_i f_i^{\intercal} + \sum_{i=k+1}^{n} \left(1 - \frac{\lambda_i}{2}\right)^T f_i f_i^{\intercal},\end{aligned} \tag{16}$$

where we recall that $\lambda_1 \leqslant \ldots \leqslant \lambda_n$ are the eigenvalues of $\mathcal{L}_G$ with the corresponding eigenvectors $f_1, \ldots, f_n$. We look at the second term in (16) and, by setting $T = c \log n/\lambda_{k+1}$, it holds for $i \geqslant k+1$ that

$$\left(1 - \frac{\lambda_i}{2}\right)^T \leqslant \left(1 - \frac{\lambda_{k+1}}{2}\right)^{2c \cdot \log n/\lambda_{k+1}} \leqslant \mathrm{e}^{-c \log n} \leqslant n^{-c}, \tag{17}$$

where the second inequality holds by the fact that $\lambda_{k+1}/2 \leqslant 1$ and $1 - x \leqslant \mathrm{e}^{-x}$ holds for $x \in [-1, 1]$. Similarly, we look at the first term in (16) and it holds for any $1 \leqslant i \leqslant k$ that

$$\left(1 - \frac{\lambda_i}{2}\right)^T \geqslant 1 - \frac{T \cdot \lambda_i}{2} = 1 - O\left(\frac{\lambda_i \cdot \log n}{\lambda_{k+1}}\right) = 1 - O\left(\frac{\log n \cdot \lambda_k}{\lambda_{k+1}}\right), \tag{18}$$

where the first inequality holds by the fact that $(1+x)^r \geqslant 1 + rx$ for any $x \geqslant -1$ and $r \geqslant 1$. Combining (16), (17) and (18), we upper bound the distance between $x^{(T)}$ and $\underline{I}x^{(0)}$ by

$$\begin{aligned}\left\|x^{(T)} - \underline{I}x^{(0)}\right\|^2 &= \left\|\left(\mathbf{P}^T - \underline{I}\right)x^{(0)}\right\|^2\\ &= \left\|\sum_{i=1}^{k}\left(1-\left(1-\frac{\lambda_i}{2}\right)^T\right) f_i f_i^{\intercal} x^{(0)} + \sum_{i=k+1}^{n}\left(1-\frac{\lambda_i}{2}\right)^T f_i f_i^{\intercal} x^{(0)}\right\|^2\\ &= \sum_{i=1}^{k}\left(1-\left(1-\frac{\lambda_i}{2}\right)^T\right)^2 \left\langle f_i, x^{(0)}\right\rangle^2 + \sum_{i=k+1}^{n}\left(1-\frac{\lambda_i}{2}\right)^{2T}\left\langle f_i, x^{(0)}\right\rangle^2\\ &= O\left(\left(\frac{\log n\cdot\lambda_k}{\lambda_{k+1}}\right)^2\sum_{i=1}^{k}\left\langle f_i, x^{(0)}\right\rangle^2 + n^{-2c}\right)\\ &= O\left(\left(\frac{\log n\cdot\lambda_k}{\lambda_{k+1}}\cdot\left\|\underline{I}x^{(0)}\right\|\right)^2 + n^{-2c}\right)\\ &= O\left(\left(\frac{\log n}{\Upsilon_G(k)}\cdot\left\|\underline{I}x^{(0)}\right\|\right)^2 + n^{-2c}\right),\end{aligned}$$

where the last inequality uses the fact that

$$\frac{\lambda_k}{\lambda_{k+1}} \leqslant \frac{\rho(k)}{2\lambda_{k+1}} = \frac{1}{2\Upsilon_G(k)}$$

by the higher-order Cheeger inequality (Lee et al. 2014). Then, taking the square root on both sides of the equality above proves the lemma. □

Our algorithm is to recover the structure of clusters based on $x^{(T)}$, and the previous lemma builds a connection between $x^{(T)}$ and $\underline{I}$. Next, we will build a connection between $\underline{I}$ and the indicator vectors of $S_1, \ldots, S_k$. Based on Lemma 4.2, we will prove in Lemma 4.3 that the bottom $k$ eigenvectors of $\mathcal{L}_G$ are close to a linear combination of the indicator vectors of $S_1, \ldots, S_k$, which implies the structure of clusters can be approximately recovered from $x^{(T)}$.

Lemma 4.2 ((Peng et al. 2015)). *Let $\{S_i\}_{i=1}^k$ be a $k$-way partition of $G$ achieving $\rho(k)$, and let $\Upsilon_G(k) = \Omega\left(k^2\right)$. Assume that $\widetilde{\chi}_i$ is the projection of $f_i$ in the span of $\{\chi_{S_1}, \ldots, \chi_{S_k}\}$. Then, it holds for any $1 \leqslant i \leqslant k$ that*

$$\|\widetilde{\chi}_i - f_i\| = O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right).$$

Lemma 4.3. *Let $\Upsilon_G(k) = \Omega\left(k^2\right)$. For any $1 \leqslant i \leqslant k$ there exists $\widehat{\chi}_i \in \mathrm{span}\{\chi_{S_1}, \ldots, \chi_{S_k}\}$ such that*

$$\|\widehat{\chi}_i - f_i\| = O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right).$$

*Moreover, $\{\widehat{\chi}_i\}_{i=1}^k$ form an orthonormal set.*

Proof. Since $\{f_i\}_{i=1}^k$ is an orthonormal set, it holds by Lemma 4.2 that $\{\widetilde{\chi}_i\}_{i=1}^k$ are *almost* orthonormal. Hence, our task is to construct an orthonormal set $\{\widehat{\chi}_i\}_{i=1}^k$ based on $\{\widetilde{\chi}_i\}_{i=1}^k$, which can be achieved by applying the

Gram-Schmidt orthonormalisation procedure. We start by setting

$$\widehat{\chi}_1 \triangleq \frac{\widetilde{\chi}_1}{\|\widetilde{\chi}_1\|}.$$

By Lemma 4.2 and the triangle inequality, it holds that

$$\left|\|\widetilde{\chi}_1\| - 1\right| = \left|\|\widetilde{\chi}_1\| - \|f_1\|\right| \leqslant \|\widetilde{\chi}_1 - f_1\| = O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right),$$

which implies that

$$\|\widehat{\chi}_1 - \widetilde{\chi}_1\| = \left\|\frac{\widetilde{\chi}_1}{\|\widetilde{\chi}_1\|} - \widetilde{\chi}_1\right\| = \left|\frac{1}{\|\widetilde{\chi}_1\|} - 1\right| \cdot \|\widetilde{\chi}_1\| = \left|\|\widetilde{\chi}_1\| - 1\right| = O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right).$$

By another application of the triangle inequality, this implies that

$$\|\widehat{\chi}_1 - f_1\| \leqslant \|\widehat{\chi}_1 - \widetilde{\chi}_1\| + \|\widetilde{\chi}_1 - f_1\| = O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right).$$

For any $2 \leqslant i \leqslant k$, we inductively construct $\widehat{\chi}_i$ in the following way:

$$\widehat{\chi}_i \triangleq \frac{\widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle\widetilde{\chi}_i, \widehat{\chi}_j\rangle\widehat{\chi}_j}{\left\|\widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle\widetilde{\chi}_i, \widehat{\chi}_j\rangle\widehat{\chi}_j\right\|}.$$

By construction, $\{\widehat{\chi}_i\}_{i=1}^{k}$ are orthonormal. It remains to bound $\|\widehat{\chi}_i - f_i\|$. Notice that each $\widehat{\chi}_i$ is a linear combination of $\widetilde{\chi}_1, \ldots, \widetilde{\chi}_i$. Indeed, $\{\widehat{\chi}_1, \ldots, \widehat{\chi}_{i-1}\}$ is a basis of the space spanned by $\widetilde{\chi}_1, \ldots, \widetilde{\chi}_{i-1}$. Let $Q_{i-1}$ be the orthogonal projection of the space spanned by $\widetilde{\chi}_1, \ldots, \widetilde{\chi}_{i-1}$. We have that

$$\left\|\widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle\widetilde{\chi}_i, \widehat{\chi}_j\rangle\widehat{\chi}_j - f_i\right\| = \|\widetilde{\chi}_i - Q_{i-1}\widetilde{\chi}_i - f_i\| \leqslant \|\widetilde{\chi}_i - f_i\| + \|Q_{i-1}\widetilde{\chi}_i\|. \tag{19}$$

By Lemma 4.2 we know that $\|\widetilde{\chi}_i - f_i\| = O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right)$. Moreover, it holds that

$$\|Q_{i-1}\widetilde{\chi}_i\| = \|\widetilde{\chi}_i\| - \|(I - Q_{i-1})\widetilde{\chi}_i\| \leqslant 1 - \|\widetilde{\chi}_i - f_i\| - |\langle\widetilde{\chi}_i, f_i\rangle| = O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right), \tag{20}$$

where the last equality follows again by Lemma 4.2. Combining with (19) and (20), we have that

$$\left\|\widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle\widetilde{\chi}_i, \widehat{\chi}_j\rangle\widehat{\chi}_j - f_i\right\| = O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right). \tag{21}$$

This implies that

$$\begin{aligned}
\|\widehat{\chi}_i - f_i\| &= \left\| \frac{\widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle \widetilde{\chi}_i, \widehat{\chi}_j\rangle \widehat{\chi}_j}{\left\|\widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle \widetilde{\chi}_i, \widehat{\chi}_j\rangle \widehat{\chi}_j\right\|} - f_i \right\| \\
&\leqslant \left\| \frac{\widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle \widetilde{\chi}_i, \widehat{\chi}_j\rangle \widehat{\chi}_j}{\left\|\widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle \widetilde{\chi}_i, \widehat{\chi}_j\rangle \widehat{\chi}_j\right\|} - \left(\widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle \widetilde{\chi}_i, \widehat{\chi}_j\rangle \widehat{\chi}_j\right)\right\| + \left\| \left(\widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle \widetilde{\chi}_i, \widehat{\chi}_j\rangle \widehat{\chi}_j\right) - f_i \right\| \\
&= \left| \frac{1}{\left\|\widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle \widetilde{\chi}_i, \widehat{\chi}_j\rangle \widehat{\chi}_j\right\|} - 1 \right| \cdot \left\| \widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle \widetilde{\chi}_i, \widehat{\chi}_j\rangle \widehat{\chi}_j \right\| + O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right) \\
&= \left| \left\| \widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle \widetilde{\chi}_i, \widehat{\chi}_j\rangle \widehat{\chi}_j \right\| - 1 \right| + O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right) \\
&= \left| \left\| \widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle \widetilde{\chi}_i, \widehat{\chi}_j\rangle \widehat{\chi}_j \right\| - \|f_i\| \right| + O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right) \\
&\leqslant \left\| \widetilde{\chi}_i - \sum_{j=1}^{i-1}\langle \widetilde{\chi}_i, \widehat{\chi}_j\rangle \widehat{\chi}_j - f_i \right\| + O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right) \\
&= O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right),
\end{aligned}$$

where the third line follows from (21), the sixth line follows from the triangle inequality, and the last line follows from (21). □

Based on Lemma 4.3, we will prove that there is a set $A$ consisting of a sufficient number of "good" nodes such that, for any random walk starting from $v \in A \cap S_j$ for some $1 \leqslant j \leqslant k$, the resulting distribution of a $T$-step random walk from $v$ is close to the indicator vector of $S_j$. This fact is formalised as follows:

Lemma 4.4. *There is a set $A \subseteq V$ with*

$$\mathrm{vol}(A) \geqslant \mathrm{vol}(V)\left(1 - \frac{\beta}{C\log(1/\beta)}\right)$$

*for some constant $C$ such that, for any $j = 1, \ldots, k$ and any $v \in A \cap S_j$, setting $x^{(0)} = \chi_v$ we have that*

$$\left\| x^{(T)} - \frac{1}{\sqrt{\mathrm{vol}(S_j)}}\chi_{S_j} \right\| = O\left( \frac{\log n}{\Upsilon_G(k) \cdot \sqrt{\beta\, \mathrm{vol}(V)}} + \sqrt{\frac{k^2 \log(1/\beta)}{\Upsilon_G(k)\beta\, \mathrm{vol}(V)}} \right).$$

Proof. Without loss of generality we assume $v \in S_j$, and let $\{\widehat{\chi}_i\}_{i=1}^k$ be the set of vectors defined in Lemma 4.3. We show that the projection of $\chi_v$ on $\mathrm{span}\{\widehat{\chi}_1, \ldots, \widehat{\chi}_k\}$ is exactly equal to $\frac{1}{\sqrt{\mathrm{vol}(S_j)}}\chi_{S_j}$. To this end, first notice that $\mathrm{span}\{\widehat{\chi}_1, \ldots, \widehat{\chi}_k\} = \mathrm{span}\{\chi_{S_1}, \ldots, \chi_{S_k}\}$, since each $\widehat{\chi}_i$ is by definition a linear combination of vectors in $\{\chi_{S_i}\}_{i=1}^k$ and

$$\dim\left(\mathrm{span}\{\widehat{\chi}_1, \ldots, \widehat{\chi}_k\}\right) = \dim\left(\mathrm{span}\{\chi_{S_1}, \ldots, \chi_{S_k}\}\right) = k.$$

Then,

$$\sum_{i=1}^k \langle \chi_v, \widehat{\chi}_i\rangle\, \widehat{\chi}_i = \sum_{i=1}^k \left\langle \chi_v, \chi_{S_i}\right\rangle \chi_{S_i} = \left\langle \chi_v, \chi_{S_j}\right\rangle \chi_{S_j} = \frac{1}{\sqrt{\mathrm{vol}(S_j)}}\chi_{S_j}. \tag{22}$$

where the first equality holds by the fact that span $\{\widehat{\chi}_1, \ldots, \widehat{\chi}_k\}$ = span $\{\chi_{S_1}, \ldots, \chi_{S_k}\}$ and the orthonormality of the two sets of vectors, and the second holds because $\chi_v$ is orthogonal to every $\chi_{S_\ell}$ with $\ell \neq j$.

Now we start with analysing the distance between the resulting distribution of a $T$-step random walk starting with $v \in S_j$ and $\frac{1}{\sqrt{\mathrm{vol}(S_j)}}\chi_{S_j}$. First of all, we apply the triangle inequality and have that

$$\left\| x^{(T)} - \frac{1}{\sqrt{\mathrm{vol}(S_j)}}\chi_{S_j} \right\| \leqslant \left\| x^{(T)} - \underline{\mathcal{I}}\chi_v \right\| + \left\| \underline{\mathcal{I}}\chi_v - \frac{1}{\sqrt{\mathrm{vol}(S_j)}}\chi_{S_j} \right\|. \tag{23}$$

We will analyse the two terms in the right hand size of (23) separately. By Lemma 4.1 we have

$$\left\| x^{(T)} - \underline{\mathcal{I}}\chi_v \right\| = O\left( \frac{\log n \cdot \left\| \underline{\mathcal{I}}\chi_v \right\|}{\Upsilon_G(k)} + n^{-c} \right). \tag{24}$$

To show that the term above is small, we introduce for any $v \in V$ a parameter $\alpha_v$ which represents how far the $\widehat{\chi}_i$'s are from the bottom eigenvectors in the entries corresponding to $v$, i.e.,

$$\alpha_v \triangleq \sqrt{\frac{1}{d_v}\sum_{i=1}^{k} \left(f_i(v) - \widehat{\chi}_i(v)\right)^2},$$

and we will prove that

$$\left\| \underline{\mathcal{I}}\chi_v \right\| = O\left( \frac{1}{\sqrt{\mathrm{vol}(S_j)}} + \alpha_v \right). \tag{25}$$

By expanding $\underline{\mathcal{I}}\chi_v$ as a linear combination of $f_1, \ldots, f_k$ we have that

$$\begin{aligned}
\left\| \underline{\mathcal{I}}\chi_v \right\|^2 &= \sum_{i=1}^{k} \langle \chi_v, f_i \rangle^2 = \sum_{i=1}^{k} \langle \chi_v, \widehat{\chi}_i - (\widehat{\chi}_i - f_i) \rangle^2 \\
&= \sum_{i=1}^{k} \left( \langle \chi_v, \widehat{\chi}_i \rangle - \langle \chi_v, \widehat{\chi}_i - f_i \rangle \right)^2 \\
&\leqslant \sum_{i=1}^{k} 2\left( \langle \chi_v, \widehat{\chi}_i \rangle^2 + \langle \chi_v, \widehat{\chi}_i - f_i \rangle^2 \right) && (26) \\
&= \frac{2}{\mathrm{vol}(S_j)} + 2\sum_{i=1}^{k} \langle \chi_v, \widehat{\chi}_i - f_i \rangle^2 && (27) \\
&\leqslant \frac{2}{\mathrm{vol}(S_j)} + 2\alpha_v^2 && (28)
\end{aligned}$$

where (26) follows from the inequality $(a-b)^2 \leqslant 2(a^2+b^2)$, (27) follows from (22), and (28) follows from the definition of $\alpha_v$. Hence, (25) holds and the first term $\left\| x^{(T)} - \underline{\mathcal{I}}\chi_v \right\|$ in the right hand side of (23) can be upper bounded by

$$\left\| x^{(T)} - \underline{\mathcal{I}}\chi_v \right\| = O\left( \frac{\log n}{\Upsilon_G(k)} \cdot \left( \frac{1}{\sqrt{\mathrm{vol}(S_j)}} + \alpha_v \right) + n^{-c} \right). \tag{29}$$

Now we look at the second term in the right hand side of (23): by (22) and the triangle inequality we have that

$$\begin{aligned}\left\|\underline{\mathcal{I}}\chi_v - \frac{1}{\sqrt{\mathrm{vol}(S_j)}}\chi_{S_j}\right\| &= \left\|\sum_{i=1}^{k}\langle\chi_v, f_i\rangle f_i - \sum_{i=1}^{k}\langle\chi_v, f_i\rangle\widehat{\chi}_i + \sum_{i=1}^{k}\langle\chi_v, f_i\rangle\widehat{\chi}_i - \sum_{i=1}^{k}\langle\chi_v, \widehat{\chi}_i\rangle\widehat{\chi}_i\right\| \\ &\leqslant \left\|\sum_{i=1}^{k}\langle\chi_v, f_i\rangle f_i - \sum_{i=1}^{k}\langle\chi_v, f_i\rangle\widehat{\chi}_i\right\| + \left\|\sum_{i=1}^{k}\langle\chi_v, f_i\rangle\widehat{\chi}_i - \sum_{i=1}^{k}\langle\chi_v, \widehat{\chi}_i\rangle\widehat{\chi}_i\right\|. \end{aligned} \tag{30}$$

Let's now analyse the two terms in (30) separately. For the first term, it holds that

$$\begin{aligned}\left\|\sum_{i=1}^{k}\langle\chi_v, f_i\rangle f_i - \sum_{i=1}^{k}\langle\chi_v, f_i\rangle\widehat{\chi}_i\right\| &\leqslant \sum_{i=1}^{k}|\langle\chi_v, f_i\rangle|\,\|f_i - \widehat{\chi}_i\| \\ &\leqslant O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right)\cdot\sum_{i=1}^{k}|\langle\chi_v, f_i\rangle| \\ &\leqslant O\left(\sqrt{\frac{k}{\Upsilon_G(k)}}\right)\cdot\sqrt{\sum_{i=1}^{k}|\langle\chi_v, f_i\rangle|^2}\cdot\sqrt{k} \\ &= O\left(\sqrt{\frac{k^2}{\Upsilon_G(k)}}\right)\cdot\|\underline{\mathcal{I}}\chi_v\| \\ &= O\left(\sqrt{\frac{k^2}{\Upsilon_G(k)\beta\,\mathrm{vol}(V)}} + \alpha_v\right) \end{aligned} \tag{31}$$

where the first inequality follows from the triangle inequality, the second by Lemma 4.3, the third one follows by the Cauchy-Schwarz inequality, and the last inequality follows by (25). For the second term of (30), we have

$$\left\|\sum_{i=1}^{k}\left(\langle\chi_v, f_i\rangle - \langle\chi_v, \widehat{\chi}_i\rangle\right)\widehat{\chi}_i\right\| = \left\|\sum_{i=1}^{k}\frac{1}{\sqrt{d_v}}\left(f_i(v) - \widehat{\chi}_i(v)\right)\widehat{\chi}_i\right\| = \sqrt{\sum_{i=1}^{k}\frac{1}{d_v}\left(f_i(v) - \widehat{\chi}_i(v)\right)^2} = \alpha_v, \tag{32}$$

where the second equality follows from the orthonormality of $\{\widehat{\chi}_i\}_i$. Combining (23), (29), (30), (31) and (32), we have

$$\begin{aligned}\left\|x^{(T)} - \frac{1}{\sqrt{\mathrm{vol}(S_j)}}\chi_{S_j}\right\| &= O\left(\frac{\log n}{\Upsilon_G(k)}\cdot\left(\frac{1}{\sqrt{\mathrm{vol}(S_j)}} + \alpha_v\right) + n^{-c}\right) + O\left(\sqrt{\frac{k^2}{\Upsilon_G(k)\beta\,\mathrm{vol}(V)}} + \alpha_v\right) + \alpha_v \\ &= O\left(\frac{\log n}{\Upsilon_G(k)\cdot\sqrt{\beta\,\mathrm{vol}(V)}} + n^{-c} + \sqrt{\frac{k^2}{\Upsilon_G(k)\beta\,\mathrm{vol}(V)}} + \alpha_v\right). \end{aligned} \tag{33}$$

Let's define the set

$$B \triangleq \left\{v \middle| \alpha_v > \sqrt{\frac{Ck^2\log(1/\beta)}{\Upsilon_G(k)\beta\,\mathrm{vol}(V)}}\right\},$$

for some constant $C$. Let us look at $\mathrm{vol}(B)$ now. By Lemma 4.3 we know that

$$\sum_{v\in V} d_v\alpha_v^2 = \sum_{i=1}^{k}\sum_{v\in V}\left(f_i(v) - \widehat{\chi}_i(v)\right)^2 = \sum_{i=1}^{k}\|f_i - \widehat{\chi}_i\|^2 \leqslant \frac{k^2}{\Upsilon_G(k)}.$$

By the definition of $B$ and the averaging argument, it holds that

$$\mathrm{vol}(B) \leqslant \frac{k^2}{\Upsilon_G(k)} \cdot \frac{\Upsilon_G(k)\beta\,\mathrm{vol}(V)}{Ck^2\log(1/\beta)} = \frac{\beta\,\mathrm{vol}(V)}{C\log(1/\beta)}. \tag{34}$$

By (33) and the definition of $B$, we have that for all $v \in S_j \setminus B$, by setting $x^{(0)} = \chi_v$ we have that

$$\begin{aligned}\left\| x^{(T)} - \frac{1}{\sqrt{\mathrm{vol}(S_j)}}\chi_{S_j} \right\| &= O\left( \frac{\log n}{\Upsilon_G(k)\cdot\sqrt{\beta\,\mathrm{vol}(V)}} + \sqrt{\frac{k^2}{\Upsilon_G(k)\beta\,\mathrm{vol}(V)}} + \sqrt{\frac{Ck^2\log(1/\beta)}{\Upsilon_G(k)\beta\,\mathrm{vol}(V)}} \right) \\ &= O\left( \frac{\log n}{\Upsilon_G(k)\cdot\sqrt{\beta\,\mathrm{vol}(V)}} + \sqrt{\frac{k^2\log(1/\beta)}{\Upsilon_G(k)\beta\,\mathrm{vol}(V)}} \right).\end{aligned}$$

By defining $A = V \setminus B$, we proved the statement. □

So far we analysed the case of a $T$-step single random walk: we showed that there is a sufficient number of "good" nodes and, for any $T$-step random walk starting from a good node $v \in S_j$, in $T$ steps the resulting distribution $x^{(T)}$ is close to $\chi_{S_j}$. To identify all the $k$ clusters simultaneously, our designed algorithm runs this process multiple times in parallel with carefully chosen initial nodes. In particular, as the resulting distribution $x^{(T)}$ is close to $\chi_{S_j}$ for most random walks starting from $v \in \chi_{S_j}$, we need to ensure that there is a random walk starting with a good node $v \in S_j$ for any $1 \leqslant j \leqslant k$. This is why we introduce the Seeding step: by setting the probability $\bar{s} \cdot d_v/\mathrm{vol}(V)$ for every node $v$ to be active, it is easy to see that with constant probability there is at least an active node in each cluster.

Our analysis for the Query step is based on the relation between our averaging procedure and lazy random walks: since any single random walk gets well mixed inside a cluster after $T$ steps, we expect that the states of the nodes inside a cluster are similar. In particular, if the $i$th random walk starts with a good node in $S_j$, we expect that $x^{(T,i)}(v) \approx 1/\mathrm{vol}(S_j)$ for most $v \in S_j$ and $x^{(T,i)}(v) \approx 0$ otherwise. Hence, nodes from the same cluster will choose the same label based on (14), while nodes from different clusters will have different labels.

Proof of Theorem 1.2. For each node $v$, the probability that we start an averaging process with initial vector $\chi_v$ is equal to $\bar{s} \cdot d_v/\mathrm{vol}(V)$, where $\bar{s} = \Theta\left(\frac{1}{\beta}\log\frac{1}{\beta}\right)$. Hence, the probability that there exists a $j$ such that no node from $S_j$ is chosen as an initial node is at most

$$\prod_{v\in S_j}\left(1 - \frac{\bar{s}\cdot d_v}{\mathrm{vol}(V)}\right) \leqslant \prod_{v\in S_j} \mathrm{e}^{-\bar{s}\cdot d_v/\mathrm{vol}(V)} = \mathrm{e}^{-\bar{s}\sum_{v\in S_j} d_v/\mathrm{vol}(V)} \leqslant \frac{1}{200k},$$

where we used the inequality $1 - x \leqslant \mathrm{e}^{-x}$ for $x \leqslant 1$, the assumption on the size of the clusters, i.e., $\mathrm{vol}(S_j) \geqslant \beta\,\mathrm{vol}(V)$, and the trivial fact that $\beta \leqslant 1/k$. As a consequence, with probability greater than 199/200, for each cluster $S_j$, at least one node $v \in S_j$ is chosen as a starting node of the averaging process.

Next, we bound the probability that all the starting nodes belong to the set $A$ defined in Lemma 4.4. By the algorithm description, the actual number of active nodes $s$ satisfies $\mathbf{E}[\,s\,] = \bar{s}$. Therefore, it holds with probability $1 - O(1)$ that $s = O\left(\frac{1}{\beta}\log\frac{1}{\beta}\right)$. We assume that this event occurs in the rest of the proof. Let $v_1, \ldots, v_s$ be the starting nodes. By Lemma 4.4, the probability that there exists a starting node $v_i$ not belonging to $A$ is at most

$$\mathbf{P}[\,\text{there is some starting node } v_i \notin A\,] \leqslant \frac{O(\bar{s})\cdot(\mathrm{vol}(V) - \mathrm{vol}(A))}{\mathrm{vol}(V)} \leqslant \frac{O(\bar{s})\cdot\beta}{C\log(1/\beta)} \leqslant \frac{1}{200}.$$

Hence, with probability $1 - O(1)$ every starting node belongs to $A$. For the rest of the proof we assume this is the case. For any node $v$, let $\mathcal{S}(v)$ be the cluster $v$ belongs to. Then, by the definition of the set $A$, it holds for any starting node $v_i$ that

$$\left\| x^{(T,i)} - \frac{\chi_{\mathcal{S}(v_i)}}{\sqrt{\mathrm{vol}(\mathcal{S}(v_i))}} \right\| = O\left( \sqrt{\frac{k^2 \log(1/\beta)}{\Upsilon_G(k)\beta\,\mathrm{vol}(V)}} \right). \tag{35}$$

On the other side, by the algorithm description we know that node $v$ could be misclassified only if

$$x^{(T,i)}(v) = \left| x^{(T,i)}(v) - \frac{\chi_{\mathcal{S}(v_i)}(v)}{\sqrt{\mathrm{vol}(\mathcal{S}(v_i))}} \right| \geqslant \frac{\sqrt{d_v}}{2\beta\,\mathrm{vol}(V)}$$

for some $i$ satisfying $\mathcal{S}(v_i) \neq \mathcal{S}(v)$, since $\chi_{\mathcal{S}(v_i)}(v) = 0$ in this case. By studying a weaker condition we know that node $v$ could be misclassified by the query step (14) only if there is some $i$ such that

$$\left| x^{(T,i)}(v) - \frac{\chi_{\mathcal{S}(v_i)}(v)}{\sqrt{\mathrm{vol}(\mathcal{S}(v_i))}} \right| \geqslant \frac{\sqrt{d_v}}{2\beta\,\mathrm{vol}(V)},$$

which is equivalent to

$$\left| \frac{x^{(T,i)}(v)}{\sqrt{d_v}} - \frac{\chi_{\mathcal{S}(v_i)}(v)}{\sqrt{d_v \cdot \mathrm{vol}(\mathcal{S}(v_i))}} \right|^2 \geqslant \frac{1}{4\beta^2\,\mathrm{vol}(V)^2}. \tag{36}$$

By (35) and the averaging argument the total volume of misclassified nodes is at most

$$\begin{aligned} & O(\bar{s}) \cdot \left\| x^{(T,i)} - \frac{1}{\sqrt{\mathrm{vol}(\mathcal{S}(v_i))}} \chi_{\mathcal{S}(v_i)} \right\|^2 \cdot 4\beta^2\,\mathrm{vol}(V)^2 \\ & = O\left( \frac{1}{\beta} \log\frac{1}{\beta} \cdot \left( \frac{k^2}{\Upsilon_G(k)} \cdot \frac{\log(1/\beta)}{\beta\,\mathrm{vol}(V)} + \frac{\log^2 n}{\Upsilon_G^2(k) \cdot \beta\,\mathrm{vol}(V)} \right) \cdot \beta^2\,\mathrm{vol}(V)^2 \right) \\ & = O\left( \frac{k^2}{\Upsilon_G(k)} \cdot \log^2\frac{1}{\beta} + \frac{\log^2 n}{\Upsilon_G^2(k)} \cdot \log\frac{1}{\beta} \right) \mathrm{vol}(V). \end{aligned}$$

Combining this with the assumption (15) proves the first statement. The second statement follows by the fact that the total communication among all nodes in each round is $O(m \cdot (1/\beta)\log(1/\beta))$ words. □

## 5 EXPERIMENTS

Now we present experimental results for our proposed algorithms. Since our clustering algorithm can be viewed as a distributed variant of the power method, we replace the distributed clustering algorithm by spectral clustering in order to evaluate our sparsification algorithm more accurately.

We test our sparsification algorithm on both synthetic and real-world datasets. To report a detailed and quantitative result, we will compare the clustering results of the following two approaches: (1) apply spectral clustering on the original input dataset; (2) apply spectral clustering on the graph returned by our sparsification algorithm. Besides giving the visualised results of our algorithm on various datasets, we use two functions to measure the quality of the above-mentioned two approaches: (1) For the synthetic datasets for which the underlying ground-truth clustering is known, the quality of a clustering algorithm is measured by the ratio of misclassified points, i.e.,

$$\mathrm{err}(A_1, \ldots, A_k) \triangleq \frac{1}{n} \cdot \sum_{i=1}^{k} |\{v \in A_i : v \notin S_i\}|,$$

where $\{S_1, \cdots, S_k\}$ is the underlying ground-truth clustering and $\{A_1, \ldots, A_k\}$ is the one returned by the clustering algorithm. (2) For datasets for which a ground-truth clustering is not well-defined, the quality of a clustering is measured by the *normalised cut value* defined by

$$\mathrm{ncut}(A_1, \ldots, A_k) \triangleq \sum_{i=1}^{k} \frac{w(A_i, V \setminus A_i)}{\mathrm{vol}(A_i)},$$

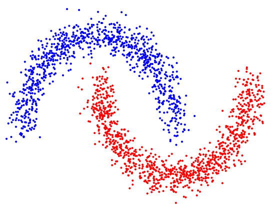
which is a standard objective function for spectral clustering algorithms (Shi and Malik 2000; Von Luxburg 2007). All the experiments are conducted with Matlab and we use an implementation of the classical spectral clustering algorithm described in (Ng et al. 2001).

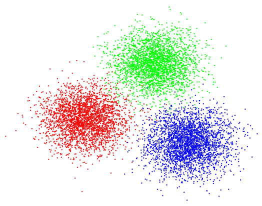
## 5.1 Datasets

We test the algorithms in the following three synthetic and real-world datasets, which are visualised in Figure 2.

- `Twomoons`: this dataset consists of $n$ points in $\mathbb{R}^2$, where $n$ is chosen between 1,000 and 15,000. We consider each point to be a node. For any two nodes $u, v$, we add an edge with weight $w(u, v) = \exp(-\|u-v\|^2/2\sigma^2)$, where $\sigma = 0.1$.
- `Gaussians`: this dataset consists of $n$ points in $\mathbb{R}^2$, where $n$ is chosen between 1,000 and 15,000. Each point is sampled from a uniform mixture of 3 isotropic Gaussians of variance 0.04. The similarity graph is constructed in the same way as `Twomoons`, and we set $\sigma = 1$ here.
- `Sculpture`: we use a $73 \times 160$ version of a photo of *The Greek Slave*[2] where each pixel is viewed as a node. To construct a similarity graph, we map each pixel to a point in $\mathbb{R}^5$ with the form $(x, y, r, g, b)$, where the last three coordinates are the RGB values. For any two nodes $u, v$, we put an edge between $u$ and $v$ with weight $w(u, v) = \exp(-\|u - v\|^2/2\sigma^2)$, where $\sigma = 20$. This results in a graph with about 11,000 nodes and $k = 3$ clusters.

These datasets are essentially the ones used in (Chen et al. 2016), which studies the effects of spectral sparsification on clustering. This makes it possible to easily compare our results with the state-of-the-art. The choice of $\sigma$ varies for different datasets, since they have in general different intra-cluster variance. There are several heuristics to choose the "correct" value of $\sigma$ (see, e.g., the classical reference (Ng et al. 2001)). In our case the value of $\sigma$ is chosen so that the spectral gap of the original similarity graph is large. This ensures that the clusters in the graph are well-defined, and spectral clustering outputs a meaningful clustering.


(a) `Twomoons`


(b) `Gaussians`

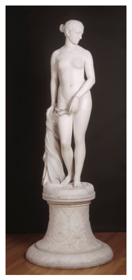

(c) `Sculpture`

Fig. 2. Visualisation of the datasets used in our experiments.

[2] http://artgallery.yale.edu/collections/objects/14794

Table 1. Experimental results for the Twomoons dataset, where $\tau = C/\lambda_{k+1}$ is set to be 0.8. Here err1 and err2 stand for the error ratios of spectral clustering on the original datasets and our sparsified graphs.

| $n$ | # edges (%) | err1 (%) | err2 (%) |
|---|---|---|---|
| 1,000 | 1.56 | 0.900 | 0.600 |
| 2,000 | 0.86 | 0.150 | 0.100 |
| 4,000 | 0.48 | 0.150 | 0.175 |
| 8,000 | 0.26 | 0.086 | 0.088 |
| 10,000 | 0.22 | 0.120 | 0.150 |
| 15,000 | 0.14 | 0.080 | 0.100 |

Table 2. Experimental results for the Gaussians dataset, where $\tau = C/\lambda_{k+1}$ is set to be 1.6. Here err1 and err2 stand for the error ratios of spectral clustering on the original datasets and our sparsified graphs.

| $n$ | # edges (%) | err1 (%) | err2 (%) |
|---|---|---|---|
| 1,000 | 3.13 | 1.700 | 1.900 |
| 2,000 | 1.75 | 1.350 | 1.650 |
| 4,000 | 0.96 | 0.400 | 0.417 |
| 8,000 | 0.66 | 0.128 | 0.140 |
| 10,000 | 0.42 | 0.113 | 0.119 |
| 15,000 | 0.29 | 0.125 | 0.148 |

## 5.2 Results on clustering quality

We test the performance of our algorithm on the three datasets. Notice that the sampling probability of the edges in our sparsification algorithm involves the factor $C/\lambda_{k+1}$. To find a desired value of $C/\lambda_{k+1}$, denoted by $\tau$, we use the following doubling method: starting with $\tau = 0.1$, we double the value of $\tau$ each time, until the spectral gap $|\lambda_{k+1} - \lambda_k|$ of the resulting matrices doesn't change significantly. Remarkably, for all the datasets considered in the paper, $\tau = 1.6$ always suffices for our purposes. Notice that this method will only increase the time complexity of our algorithm by at most a poly-logarithmic factor of $n$.

For the Twomoons and Gaussians datasets, for all the tested graphs with size ranging from 1,000 to 15,000 points, our sparsified graphs require only about $(1.63 \pm 1.5)\%$ of the total edges. The error ratios of spectral clustering on the original datasets and our sparsified graphs are listed respectively as err1 and err2, and are always very close. See Table 1 and Table 2 for details.

The Sculpture dataset corresponds to a similarity graph of $n = 11,680$ nodes and 68 million edges. We run spectral clustering on both the input graph and our sparsified one, and compute the normalised cut values of each clustering in the original input graph. By setting $\tau = 1.6$, our algorithm samples only 0.37% of the edges (320,000) from the input graph. The normalised cut value of spectral clustering on the original dataset is 0.0938, while the normalised cut value of spectral clustering on our sparsified graph is 0.0935. The visualisations of the two clustering results are almost identical, as shown in Figure 3.

## ACKNOWLEDGMENTS

We would like to thank Dr. Emanuele Natale and Prof. Luca Trevisan who found a mistake in our paper that studies the same problem and appeared at SPAA'17. To fix that mistake, in the current paper we designed and

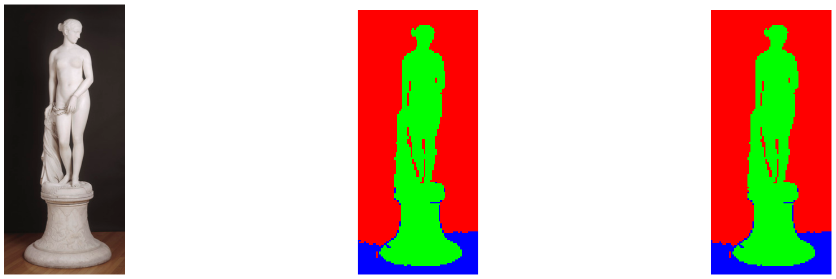


Fig. 3. Visualisation of the results on Sculpture. The left-side picture is the original input dataset, the middle one is the output of spectral clustering on the original input dataset, while the right-side picture is the output of spectral clustering on our sparsified graph.

analysed a different algorithm that works for a more general family of graphs. In particular, our algorithm works for non-regular graphs, while our claimed statement in the flawed SPAA'17 paper requires the underlying graph to be regular.

The second-named author wishes to acknowledge financial support received from the ERC Starting Grant (DYNAMIC MARCH).

## REFERENCES

Zeyuan Allen-Zhu, Silvio Lattanzi, and Vahab S. Mirrokni. 2013. A Local Algorithm for Finding Well-Connected Clusters. In *30th International Conference on Machine Learning (ICML'13)*. 396–404.

Joshua Batson, Daniel A. Spielman, and Nikhil Srivastava. 2012. Twice-Ramanujan sparsifiers. *SIAM J. Comput.* 41, 6 (2012), 1704–1721.

Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. 2017. Find your place: Simple distributed algorithms for community detection. In *28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*. 940–959.

Luca Becchetti, Andrea E. F. Clementi, Pasin Manurangsi, Emanuele Natale, Francesco Pasquale, Prasad Raghavendra, and Luca Trevisan. 2018. Average Whenever You Meet: Opportunistic Protocols for Community Detection. In *26th Annual European Symposium on Algorithms (ESA'18)*. 7:1–7:13.

András A. Benczúr and David R. Karger. 1996. Approximating $s$-$t$ minimum cuts in $\widetilde{O}(n^2)$ time. In *28 Annual ACM Symposium on Theory of Computing (STOC'96)*. 47–55.

Manuel Blum, Richard M. Karp, Oliver Vornberger, Christos H. Papadimitriou, and Mihalis Yannakakis. 1981. The Complexity of Testing Whether a Graph is a Superconcentrator. *Inf. Process. Lett.* 13, 4/5 (1981), 164–167.

Jiecao Chen, He Sun, David P. Woodruff, and Qin Zhang. 2016. Communication-Optimal Distributed Clustering. In *29th Advances in Neural Information Processing Systems (NIPS'16)*. 3720–3728.

Fan Chung and Linyuan Lu. 2006. Concentration inequalities and martingale inequalities: a survey. *Internet Math.* 3, 1 (2006), 79–127.

Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3 (2010), 75–174.

Shayan Oveis Gharan and Luca Trevisan. 2012. Approximating the Expansion Profile and Almost Optimal Local Graph Clustering. In *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'12)*. 187–196.

Pan Hui, Eiko Yoneki, Shu Yan Chan, and Jon Crowcroft. 2007. Distributed community detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture.*

David Kempe and Frank McSherry. 2004. A decentralized algorithm for spectral analysis. In *36th Annual ACM Symposium on Theory of Computing (STOC'04)*. 561–568.

James R. Lee, Shayan Oveis Gharan, and Luca Trevisan. 2014. Multiway Spectral Partitioning and Higher-Order Cheeger Inequalities. *Journal of the ACM* 61, 6 (2014), 37:1–37:30.

Yin Tat Lee and He Sun. 2015. Constructing Linear-Sized Spectral Sparsification in Almost-Linear Time. In *56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*. 250–269.

Yin Tat Lee and He Sun. 2017. An SDP-based algorithm for linear-sized spectral sparsification. In *49th Annual ACM Symposium on Theory of Computing (STOC'17)*.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *14th Advances in Neural Information Processing Systems (NIPS'01)*. 849–856.

Shayan Oveis Gharan and Luca Trevisan. 2014. Partitioning into Expanders. In *25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*. 1256–1266.

Richard Peng, He Sun, and Luca Zanetti. 2015. Partitioning Well-Clustered Graphs: Spectral Clustering Works!. In *28th Conference on Learning Theory (COLT'15)*. 1423–1455.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence* 22, 8 (2000), 888–905.

Daniel A. Spielman and Nikhil Srivastava. 2011. Graph Sparsification by Effective Resistances. *SIAM J. Comput.* 40, 6 (2011), 1913–1926.

Daniel A. Spielman and Shang-Hua Teng. 2013. A Local Clustering Algorithm for Massive Graphs and Its Application to Nearly Linear Time Graph Partitioning. *SIAM J. Comput.* 42, 1 (2013), 1–26.

Daniel A Spielman and Shang-Hua Teng. 2011. Spectral sparsification of graphs. *SIAM J. Comput.* 40, 4 (2011), 981–1025.

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

Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

Wenzhuo Yang and Huan Xu. 2015. A Divide and Conquer Framework for Distributed Graph Clustering. In *32nd International Conference on Machine Learning (ICML'15)*. 504–513.