An Effective Approach to Semi-supervised Cluster Extraction

Ming-Jun Lai MJLAI@UGA.EDU
Department of Mathematics
University of Georgia
Athens, GA 30602, USA

Zhaiming Shen ZHAIMING.SHEN@UGA.EDU
Department of Mathematics
University of Georgia
Athens, GA 30602, USA

Editor: Maya Gupta, Samory K. Kpotufe, Sanjiv Kumar, Ingo Steinwart

Abstract

A least square based semi-supervised local clustering algorithm and its variants are proposed to extract clusters from a graph with known adjacency matrix. The algorithms are based on a two stage approaches similar to the ones proposed by Lai and Mckenzie (2020). However, under a weaker assumption and with less computational complexity than the one in Lai and Mckenzie (2020), the algorithm is shown to be able to find a desired cluster with high probability. Several numerical experiments including the synthetic data and real data such as MNIST, AT&T and YaleB human faces data sets are conducted to demonstrate the performance of this approach. A comparison with several known algorithms are also be given to show that our algorithm is very effective.

1. Introduction

Informally speaking, graph clustering is a problem of dividing the set of vertices of a graph into subsets in a way which makes more edges within each subset, and fewer edges between different subsets. When analyzing a graph, one of people's primary interest is to find the underlying clustered structure of the graph, as the vertices in the same cluster can reasonably be assumed to have some latent similarity. Even though for data set which are not presented as graphs, it can be done by first creating a suitable auxiliary graph based on the data, for example, the K-nearest-neighbors (K-NN) graph, and then apply graph clustering techiques on this auxiliary graph.

Graph clustering problem has become prevalent recently in areas of social network study, such as Fortunato (2010), Hric et al. (2014), Kossinets and Watts (2016), image classification such as Camps-Valls et al. (2007), Chen et al. (2005), Shi and Malik (2000), natural language processing such as Dhillon (2001), Mihalcea and Radev (2001). For example, suppose a social network graph has vertices which represents users of a social network (e.g. Facebook, Linkedln), then the edges could represent users which are connected to each other. The sets of nodes with high inter-connectivity, which we call them communities or

clusters, could represent friendship groups or co-workers. By identifying those communities we can suggest new connections to users. Note that some networks are directed (e.g. Twitter, Citation Networks), which could make community detection more subtle. For the scope of this paper, we will only focus on weighted undirect graphs. We leave the directed graph case for future work.

The classical graph based clustering problem is a global clustering problem which assigns every vertice a unique cluster, assuming there are no multi-class vertices. It is usually considered as an unsupervised learning problem which can be done by using method such as spectral clustering, see Luxburg (2007), Ng et al. (2002), Zelnik-Manor and Perona (2004), or ways of finding an optimal cut of the graph, see Dhillon et al. (2004), Ding et al. (2001). These approaches are generally computational expensive and hard to implement for large data sets. It can also be done semi-supervisely, such as Kulis et al. (2015), Jacobs et al. (2018), Yin and Tai (2018). However, sometimes it is only of people's interests in finding one certain cluster which contains the target vertices, given some prior knowledge of a small portion of labels for the entire true cluster, which is usually attainable for real data. This type of problem is called local clustering, or local cluster extraction, which loosely speaking, is defined to be the problem which takes a set of vertices Γ with known labels, called seed vertices, as input, and returns a cluster $C^{\#}$ such that $\Gamma \subset C^{\#}$. In this paper, we proposed a new approach using the ideas of compressed sensing and method of least square together to solve it effectively.

The local clustering problem haven't been studied exhaustively, and many aspects of the local clustering problem still remain open. Some recent related work are by Ha et al. (2020), Yan et al. (2019), Yin et al. (2017), Veldt et al. (2019), Lai and Mckenzie (2020). Especially for Lai and Mckenzie (2020), which is one of the recent works for the same setting as ours. However, as we will see in the numerical experiments section, our approach outperforms them both in terms of the accuracy and efficiency.

The main contribution of this paper is that it proposes a local cluster extraction algorithm which improves the state-of-the-art results by Lai and Mckenzie (2020). The subsequent sections in this paper are structured as follows. In Section 2, we give brief introductions to the concept of in spectral clustering such as graph Laplacian and Theorem 2, we also make the assumptions for the graph model which we will use later for theoretical analysis. In Section 3, we introduce the main algorithms for solving the local cluster extraction problem in two stages, namely *RandomWalkThreshold* and *LeastSquareClusterPursuit*, and we show the correctness of our algorithms asymptotically. In Section 4, we analyze the asymptotic complexity for our algorithms. In Section 5, several synthetic and real data sets are used to evaluate the performance of the algorithms and we also compared the performance with the state-of-the-art results.

2. Preliminaries and Models

In this section, we will give some preliminary definitions and our graph model assumptions which will be used and analyzed in later sections.

2.1 Notations and Definitions

We use standard notation $G = (V, E)$ to denote the graph G with the set of vertices V and set of edges E. For the case $|V| = n$, we identify V with the set of integers $[n] := \{1, 2, \cdots, n\}$. We use A to denote the adjacency matrix (possibly weighted but weights are non-negative) of G, so in the undirected case, A is a symmetric matrix. Let D be the diagonal matrix $D = diag(d_1, d_2, \cdots, d_n)$, where each d_i is the degree of vertex i. We have the following definition.

Definition 1. *The unnormalized* **Graph Laplacian** *is defined as $L = D - A$. There are also two other graph Laplacians which are symmetric graph Laplacian $L_{sym} := I - D^{-1/2}AD^{-1/2}$, and the random walk graph Laplacian $L_{rw} := I - D^{-1}A$.*

The following theorem serves as the fundamental theorem for solving graph clustering in our approach, we omit the proof here by directly referring to Chung (1997) and Luxburg (2007) .

Theorem 2. *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L (L_{rw}) equals to the number of connected components $C_1, C_2, \cdots, C_k$ in G, and the indicator vectors $1_{C_1}, \cdots, 1_{C_k}$ on these components span the kernel of L (L_{rw}).*

Let us further introduce some notations which we will use later. Suppose for the moment we have information about structure of the underlying clusters for each vertex, then it is useful to write G as a union of two edge-disjoint subgraphs $G = G^{in} \cup G^{out}$ where $G^{in} = (V, E^{in})$ consists of only intra-connection edges, and $G^{out} = (V, E^{out})$ consists of only inter-connection edges. We will use d_i^{in} to denote the degree of vertex i in the subgraph G^{in}, and d_i^{out} to denote the degree of vertex i in the subgraph G^{out}. We will also use A^{in} and L^{in} to denote the adjacency matrix and graph Laplacian associated with G^{in}, and A^{out} and L^{out} to denote the adjacency matrix and graph Laplacian associated with G^{out}. Note that these notations are just for convenience for the analysis in the next section, in reality we will have no assurance about which cluster each individual vertex belongs to, so we will have no access to A^{in} and L^{in}. It is also worthwhile to point out that $A = A^{in} + A^{out}$ but $L \neq L^{in} + L^{out}$ in general. In addition, we will use $|L|$ to denote the same matrix L with the entries in absolute value instead of the original entries in L.

2.2 Graph Model Assumption

We make the following assumption for our analysis of graph model in the asymptotic perspective.

Assumption 1. *Suppose $G = (V, E)$ can be partitioned into $k = O(1)$ connected components such that $V = C_1 \cup \cdots \cup C_k$, where each C_i is the underlying vertex set for each connected component of G.*

(I): The degree of each vertex is asymptotically the same for vertices belong to the same cluster C_i.

(II): The degree d_i^{out} is small relative to degree d_i^{in} asymptotically for each vertex $i \in V$.

The random graphs which satisfies assumption (I) is not uncommon, for example, the Erdös-Rényi (ER) model $G(n,p)$ with $p \sim \frac{\omega(n)\log(n)}{n}$ for any $\omega(n) \to \infty$, see Erdős and Rényi (1959) and Chung and Lu (2006). A natural generalization of the ER model is the stochastic block model (SBM), see Holland et al. (1983), which is a generative model for random graphs with certain edge densities within and between underlying clusters, such that the edges within clusters are more dense than the edges between clusters. In the case of each cluster has the same size and the intra- and inter-connection probability are the same among all the vertices, we have the symmetric stochastic block model (SSBM). It is worthwhile to note that the information theoretical bound for exact cluster recovery in SBM are given by Abbe and Sandon (2015). It was also shown by Lai and Mckenzie (2020) that a general SBM with certain choice of parameters can be clustered by using a compressed sensing approach. Our model requires a weaker assumption than the one in Lai and Mckenzie (2020), indeed, we remove the assumption about the eigenvalues of graph Laplacian L which is imposed on in Lai and Mckenzie (2020). Therefore, our model will be applicable to a broader range of random graphs.

3. Cluster Extraction Algorithms

For simplicity, we will use L and L^{in} to denote L_{rw} and L^{in}_{rw} respectively. Our analysis is based on the following key observation. Suppose for the moment graph G has k connected components $C_1, \cdots, C_k$, i.e., $L = L^{in}$. Suppose further that we have access to the information about the structure of L^{in}, then we can write the graph Laplacian L^{in} into a block diagonal form. Therefore to find all the clusters, it is equivalent to find the first k eigenvectors of L^{in}.

$$L = L^{in} = \begin{pmatrix} L_1^{in} & & & \\ & L_2^{in} & & \\ & & \ddots & \\ & & & L_k^{in} \end{pmatrix}$$

Suppose now we are interested in finding the cluster with the smallest number of vertices, say C_1, which corresponds to L_1^{in}. By Theorem 2, $\{\mathbf{1}_{C_1}, \cdots, \mathbf{1}_{C_k}\}$ forms a basis of the kernel W_0 of L. Note that all the $\mathbf{1}_{C_i}$ have disjoint supports, so for $\mathbf{w} \in W_0$ and $\mathbf{w} \neq 0$, we can write

$$\mathbf{w} = \sum_{i=1}^{k} \alpha_i \mathbf{1}_{C_i}$$

with some $\alpha_i \neq 0$. Therefore, if $\mathbf{1}_{C_1}$ has the fewest non-zero entries among all elements of $W_0 \setminus \{\mathbf{0}\}$, then we can find it by solving the following minimization problem:

$$\min ||\mathbf{w}||_0 \quad \text{subject to} \quad L^{in}\mathbf{w} = 0 \quad \text{and} \quad \mathbf{w} \neq 0. \tag{1}$$

This problem can be solved using methods such as greedy algorithms in compressed sensing as explained in Lai and Mckenzie (2020). However, we will propose a different approach to tackle it in this paper and demonstrate that the new approach is more effective numerically and require a fewer number of assumptions.

3.1 Least Square Cluster Pursuit

We will use L_C and L_C^{in} to denote the submatrices of L and L^{in} with column indices subset $C \subset V = [n]$ respectively. Now consider Problem (1) again, instead of finding C_1 directly, let us try to find what are not in C_1. Suppose there is a superset $\Omega \subset V$ such that $C_1 \subset \Omega$, and $C_i \not\subset \Omega$ for all $i = 2, \cdots, n$. Then we know

$$L^{in}\mathbf{1}_\Omega = L^{in}(\mathbf{1}_{\Omega\setminus C_1} + \mathbf{1}_{C_1}) = L^{in}\mathbf{1}_{\Omega\setminus C_1} + L^{in}\mathbf{1}_{C_1} = L^{in}\mathbf{1}_{\Omega\setminus C_1}.$$

Letting $\mathbf{y} := L^{in}\mathbf{1}_\Omega$, we see that solving Problem (1) will be equivalent to solve the following problem (2), but with some condition imposed.

$$\arg\min_{\mathbf{w}\in\mathbb{R}^n} \|L^{in}\mathbf{x} - \mathbf{y}\|_2 \tag{2}$$

Note that solving Problem (2) directly will give $\mathbf{x}^* = \mathbf{1}_\Omega \in \mathbb{R}^n$ and $\mathbf{x}^* = \mathbf{1}_{\Omega\setminus C_1} \in \mathbb{R}^n$ both as solutions. By setting the columns $L^{in}_{V\setminus\Omega} = 0$, solving Problem (2) is equivalent to solving

$$\arg\min_{\mathbf{x}\in\mathbb{R}^{|\Omega|}} \|L^{in}_\Omega\mathbf{x} - \mathbf{y}\|_2. \tag{3}$$

Directly solving Problem (3) gives at least two solutions $\mathbf{x}^* = \mathbf{1} \in \mathbb{R}^{|\Omega|}$ and $\mathbf{x}^* = \mathbf{1}_{C_1^c} \in \mathbb{R}^{|\Omega|}$, where the latter one is much more informative for us to extract C_1 from Ω than the former. So we need to find a way to avoid the non-informative solution $\mathbf{x}^* = \mathbf{1}$ but keep the informative one $\mathbf{x}^* = \mathbf{1}_{C_1^c}$.

We can achieve this by removing a proportion of column indices set T from Ω. Let us consider

$$\arg\min_{\mathbf{x}\in\mathbb{R}^{|\Omega|-|T|}} \|L^{in}_{\Omega\setminus T}\mathbf{x} - \mathbf{y}\|_2, \tag{4}$$

where now $\mathbf{1} \in \mathbb{R}^{|\Omega|-|T|}$ is not a solution to (4) any more. However, if we can assure a suitable subset T such that $T \subset C_1$, then $\mathbf{1}_{C_1^c} \in \mathbb{R}^{|\Omega|-|T|}$ is still a solution to it, since $L^{in}_{\Omega\setminus T}\mathbf{1}_{C_1^c} = L^{in}\mathbf{1}_{\Omega\setminus C_1} = 0$. Note that Problem (4) has a unique solution because it is a least square problem with matrix $L^{in}_{\Omega\setminus T}$ of full column rank. This idea leads to Algorithm 1.

Remark 3. *Note that there are certainly some other heuristic ways to choose the indices set T, based on different measures on L_Ω and $\mathbf{y}$. For example, we can choose a set of seeds for T if we are given some seeds of the concerned cluster. But the way we choose T in this paper is based on the following observation. Suppose $L = L^{in}$, $\Omega \supset C_1$ and $\Omega \not\supset C_i$ for $i = 2, \cdots k$. Then $|L_a| \cdot |\mathbf{y}| = 0$ for all $a \in C_1$, and $|L_a| \cdot |\mathbf{y}| > 0$ for all $a \in \Omega \setminus C_1$. We impose the absolute value rather than direct dot product in order to have fewer cancellation between vector components when summing over the entrywised products. In practice, the size of γ will not matter too much as long as it is not being pushed too extreme.*

Remark 4. *As indicated by Lai and Mckenzie (2020), we can reformulate problem (1) as solving*

$$\arg\min_{\mathbf{x}\in\mathbb{R}^n}\{\|L\mathbf{x} - \mathbf{y}\|_2 : \quad \|x\|_0 \le s\} \tag{6}$$

Algorithm 1 *LeastSquareClusterPursuit*

Require: Adjacency matrix A, vertex subset $\Omega \subset V$, least square threshold parameter $\gamma \in (0,1)$, and rejection parameter $R \in [0,1)$.

- Compute $L = I - D^{-1}A$ and $\mathbf{y} = L\mathbf{1}_\Omega$.
- Let T be the set of column indices of $\gamma \cdot |\Omega|$ smallest components of the vector $|L_\Omega|^\top \cdot |\mathbf{y}|$ (*Here the absolute value operation is entrywised*).
- Let $\mathbf{x}^\#$ be the solution to

$$\arg\min_{\mathbf{x}\in\mathbb{R}^{|\Omega|-|T|}} \|L_{\Omega\setminus T}\mathbf{x} - \mathbf{y}\|_2 \tag{5}$$

  obtained by using an iterative least square solver.
- Let $W^\# = \{i : \mathbf{x}_i^\# > R\}$.

Ensure: $C_1^\# = \Omega \setminus W^\#$.

by applying the greedy algorithms such as subspace pursuit in Dai and Milenkovic (2009) and compressed sensing matching pursuit (CoSaMP) in Needell and Tropp (2009). Or alternatively, we can consider the LASSO (Santosa and Symes, 1986) (Tibshirani, 1996) form of the problem

$$\arg\min_{\mathbf{x}\in\mathbb{R}^n}\{\|L\mathbf{x}-\mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1\} = \arg\min_{\mathbf{x}\in\mathbb{R}^n}\{\|L\mathbf{x}-\mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_0\}. \tag{7}$$

The reason that Lasso is a good way to interpret this problem is that the solution $\mathbf{x}^*$ we are trying to solve for is the sparse indicator vector which satisfies $\|\mathbf{x}^*\|_1 = \|\mathbf{x}^*\|_0$. We will implement these two ideas in the numerical results section for the comparison.

However, in reality we have no access to L^{in}, what we know only is L, and in general $L \neq L^{in}$. We hope the solution to the above problem associated with L will not be perturbed too much from the solution $\mathbf{1}_{C_1^\varsigma}$ associated with L^{in} if the difference between L and L^{in} is relative small and the subset $T \subset \Omega$ is chosen to be appropriate. Let us make this precise by first quoting the following standard result in numerical analysis.

Lemma 5. *Let $\|\cdot\|$ be an operator norm, $A \in \mathbb{R}^{n\times n}$ be a non-singular square matrix, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^n$. Let $\tilde{A}$, $\tilde{\mathbf{x}}$, $\tilde{\mathbf{y}}$ be perturbed measurements of A, $\mathbf{x}$, $\mathbf{y}$ respectively. Suppose $A\mathbf{x} = \mathbf{y}$, $\tilde{A}\tilde{\mathbf{x}} = \tilde{\mathbf{y}}$, and suppose further cond$(A) < \frac{\|A\|}{\|\tilde{A}-A\|}$, then*

$$\frac{\|\tilde{\mathbf{x}}-\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{cond(A)}{1 - cond(A)\frac{\|\tilde{A}-A\|}{\|A\|}}\Big(\frac{\|\tilde{A}-A\|}{\|A\|} + \frac{\|\tilde{\mathbf{y}}-\mathbf{y}\|}{\|\mathbf{y}\|}\Big).$$

The above lemma tells us that the size of cond(A) is significant in determining the stability of the solution $\mathbf{x}$ is with respect to small perturbations on A and $\mathbf{y}$. For the discussion from now on, we will use $\|\cdot\|$ to denote the standard vector or matrix induced two-norm $\|\cdot\|_2$ unless state otherwise. The next lemma asserts the invertibility of $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$ and gives an estimation bound of its condition number.

Lemma 6. *Let $V = \cup_{i=1}^{k} C_i$ be the disjoint union of $k = O(1)$ underlying clusters with size n_i and assume (I). Let d_j be the degree for vertex $j \in V = [n]$, $n_1 = \min_{i\in[k]} n_i$, and suppose $\Omega \subset V$ be such that $\Omega \supset C_1$ and $\Omega \not\supset C_i$ for $i = 2, \cdots k$. Then*

(i). If $T \subset C_1$, then $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$ is invertible.

(ii). Suppose further $\lceil \frac{3n_1}{4} \rceil \le |T| < n_1$ and $|\Omega| \le \lceil \frac{5n_1}{4} \rceil$. Then $cond\big((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big) \le 4$ almost surely as $n_1 \to \infty$, e.g. when $n \to \infty$.

Proof. The invertibility is straightforward since $L^{in}_{\Omega\setminus T}$ is of full column rank, therefore $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$ is invertible. Without loss of generality, let us assume that the column indices of $L^{in}_{\Omega\setminus T}$ are already permuted such that the indices number is in the same order relative to their underlying clusters. Now since $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$ is in a block form, to estimate the condition number, we only need to estimate the largest and smallest eigenvalues for each block. Writing $L^{in}_{\Omega\setminus T} = [l_{ij}]$ and $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T} = [s_{ij}]$, for each $i \in C_1 \setminus T$, $s_{ii} = \sum_{k=1}^{n} l_{ki}l_{ki} = \sum_{k=1}^{n} l_{ki}^2 = \sum_{k=1}^{n_1} l_{ki}^2 = 1 + \frac{1}{d_i^{in}}$, and for $i, j \in C_1 \setminus T$ with $i \ne j$, $s_{ij} = \sum_{k=1}^{n} l_{ki}l_{kj} = \sum_{k=1}^{n_1} l_{ki}l_{kj}$. Note that the probability of having an edge between i and j given degree sequences $d_1, \cdots d_{n_1}$ equals to $\frac{d_i d_j}{\sum_{i\in C_1} d_i}$, assuming that the existence of an edge between two vertices is proportional to their degrees. So l_{ij} equals to $-\frac{1}{d_i}$ with probability $\frac{d_i d_j}{\sum_{i\in C_1} d_i}$, which implies $\mathbb{E}(l_{ij}) = -\frac{d_j}{\sum_{i\in C_1} d_i}$; l_{ji} equals to $-\frac{1}{d_j}$ with probability $\frac{d_i d_j}{\sum_{i\in C_1} d_i}$, which implies $\mathbb{E}(l_{ji}) = -\frac{d_i}{\sum_{i\in C_1} d_i}$. Hence the expectation

$$\begin{aligned}
\mathbb{E}(s_{ij}) &= \mathbb{E}(\sum_{k=1}^{n} l_{ki}l_{kj}) = \sum_{k=1}^{n} \mathbb{E}(l_{ki})\mathbb{E}(l_{kj}) = \sum_{k=1}^{n_1} \mathbb{E}(l_{ki})\mathbb{E}(l_{kj}) \\
&= \frac{d_i d_j}{\sum_{i\in C_1} d_i} \cdot (-\frac{1}{d_i}) + \frac{d_i d_j}{\sum_{i\in C_1} d_i} \cdot (-\frac{1}{d_j}) + \frac{d_k d_i}{\sum_{i\in C_1} d_i} \cdot \frac{d_k d_j}{\sum_{i\in C_1} d_i} \cdot (\frac{1}{d_k})^2 \\
&= -\frac{d_i + d_j}{\sum_{i\in C_1} d_i} + \frac{d_i d_j}{(\sum_{i\in C_1} d_i)^2} = -\frac{2}{n_1} + \frac{1}{n_1^2}.
\end{aligned}$$

By the law of large numbers, $s_{ij} \to -\frac{2}{n_1} + \frac{1}{n_1^2}$ almost surely as $n_1 \to \infty$. Therefore for $i \in C_1 \setminus T$, we have

$$\sum_{j\in C_1\setminus T, j\ne i} |s_{ij}| \to |C_1 \setminus T| \cdot (\frac{2}{n_1} - \frac{1}{n_1^2}) = \frac{n_1}{4} \cdot (\frac{2}{n_1} - \frac{1}{n_1^2}) \le \frac{1}{2}$$

almost surely as $n_1 \to \infty$. Similarly, for each $i \in C_k \cap (\Omega \setminus C_1)$, $k \ge 2$, we have $s_{ii} = 1 + \frac{1}{d_i^{in}}$, and $\sum_{j\in C_k\cap(\Omega\setminus C_1), j\ne i} |s_{ij}| \to \frac{n_1}{4} \cdot (\frac{2}{n_k} - \frac{1}{n_k^2}) \le \frac{1}{2}$ almost surely as $n_1 \to \infty$.

Now we apply Gershgorin's circle theorem to bound the spectrum of $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$. For all $i \in \Omega \setminus T$, the circles are centered at $1 + \frac{1}{d_i}$, with radius less than or equal to $\frac{1}{2}$ almost surely, hence $\sigma_{\min}((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}) \ge \frac{1}{2}$ and $\sigma_{\max}((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}) \le \frac{3}{2} + \frac{1}{d_i} \le 2$. almost surely. Therefore we have

$$\mathrm{cond}\big((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big) = \frac{\sigma_{\max}((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T})}{\sigma_{\min}((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T})} \le 4$$

almost surely, as desired. □

Remark 7. *Note that there is a minor difficulty in estimating the expectation of inner product between two different columns of $L^{in}_{\Omega\setminus T}$. The computation assumes the independence of degree distribution of each individual vertex within each cluster, but this may not be true in general for arbitrary graph. However, the independence will occur if the asymptotic uniformity of the degree distribution within each cluster is assumed, that is why our model needs this assumption.*

Now the perturbed Problem (5) is equivalent to solving $(L^\top_{\Omega\setminus T}L_{\Omega\setminus T})\mathbf{x}^\# = L^\top_{\Omega\setminus T}\tilde{\mathbf{y}} = L^\top_{\Omega\setminus T}(L\mathbf{1}_\Omega)$, while the unperturbed Problem (4) is to solve $(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\mathbf{x}^* = (L^{in}_{\Omega\setminus T})^\top\mathbf{y} = (L^{in}_{\Omega\setminus T})^\top(L^{in}\mathbf{1}_\Omega)$. Let $M := L - L^{in}$, $M_\Omega := L_\Omega - L^{in}_\Omega$, and $M_{\Omega\setminus T} := L_{\Omega\setminus T} - L^{in}_{\Omega\setminus T}$. Now let us give an estimate for M.

Lemma 8. *Let L be the graph Laplacian of G and $M := L - L^{in}$. Let $\epsilon_i := \frac{d^{out}_i}{d_i}$ for all i and $\epsilon_{max} := \max_{i\in[n]}\epsilon_i$. Then $\|M\| \le 2\epsilon_{\max}$.*

Proof. Let δ_{ij} denote the Kronecker delta symbol, observe that

$$L_{ij} := \delta_{ij} - \frac{1}{d_i}A_{ij} = \delta_{ij} - \frac{1}{d^{in}_i + d^{out}_i}(A^{in}_{ij} + A^{out}_{ij}).$$

Since $\epsilon_i := \frac{d^{out}_i}{d_i}$, we have $\frac{1}{d_i} = \frac{1}{d^{in}_i + d^{out}_i} = \frac{1}{d^{in}_i} - \frac{\epsilon_i}{d^{in}_i}$. So we have

$$\begin{aligned} L_{ij} &= \delta_{ij} - \Big(\frac{1}{d^{in}_i} - \frac{\epsilon_i}{d^{in}_i}\Big)(A^{in}_{ij} + A^{out}_{ij}) \\ &= \Big(\delta_{ij} - \frac{1}{d^{in}_i}A^{in}_{ij}\Big) - \frac{1}{d^{in}_i}A^{out}_{ij} + \frac{\epsilon_i}{d^{in}_i}(A^{in}_{ij} + A^{out}_{ij}) \\ &= L^{in}_{ij} - \frac{1-\epsilon_i}{d^{in}_i}A^{out}_{ij} + \frac{\epsilon_i}{d^{in}_i}A^{in}_{ij}. \end{aligned}$$

Therefore $M_{ij} = -\frac{1-\epsilon_i}{d^{in}_i}A^{out}_{ij} + \frac{\epsilon_i}{d^{in}_i}A^{in}_{ij}$. To bound the spectral norm we apply Gershgorin's circle theorem, noting that $M_{ii} = 0$ for all i, hence

$$\begin{aligned} \|M\|_2 &= \max\{|\lambda_i| : \lambda_i \quad \text{eigenvalue of} \quad M\} \le \max_i \sum_j |M_{ij}| \\ &= \max_i \Big\{\frac{1-\epsilon_i}{d^{in}_i}\sum_j A^{out}_{ij} + \frac{\epsilon_i}{d^{in}_i}\sum_j A^{in}_{ij}\Big\} \\ &= \max_i \Big\{\frac{1-\epsilon_i}{d^{in}_i}d^{out}_i + \frac{\epsilon_i}{d^{in}_i}d^{in}_i\Big\} = 2\max_i \epsilon_i = 2\epsilon_{\max}. \end{aligned}$$

□

Next we will have the following result.

Lemma 9. $\|(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega \mathbf{1}_\Omega\| \ge \frac{\sqrt{|\Omega\setminus C_1|}}{2} = \frac{\sqrt{n_1}}{4}$ *almost surely.*

Proof. Note that $\|(L^{in}_{\Omega\setminus T})^\top (L^{in}\mathbf{1}_\Omega)\| = \|(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega \mathbf{1}\|$. Let us estimate $\|(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega \mathbf{1}\|$. Similar to the computation we did in Lemma 6, for each $i \in C_1 \setminus T$, we have $s_{ii} = 1 + \frac{1}{d^{in}_i}$, $\sum_{j\in C_1} s_{ij} = 0$, and $\sum_{j\in\Omega\setminus C_1} s_{ij} = 0$. For each $i \in C_k \cap (\Omega\setminus C_1)$, $k \geq 2$, we have $s_{ii} = 1 + \frac{1}{d^{in}_i}$, $\sum_{j\in C_1} s_{ij} = 0$, and $\sum_{j\in C_k\cap(\Omega\setminus C_1), j\neq i} s_{ij} \to \frac{n_1}{4}\cdot(-\frac{2}{n_k} + \frac{1}{n_k^2}) \geq -\frac{1}{2}$ almost surely. Therefore, the row sum of $(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega$ for row $i \in C_1 \setminus T$ equals to zero, and the row sum $(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega$ for row $i \in \Omega \setminus C_1$ larger than $\frac{1}{2}$ almost surely. Hence $\|(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega \mathbf{1}\| \geq \frac{\sqrt{|\Omega\setminus C_1|}}{2} = \frac{\sqrt{n_1}}{4}$ almost surely. □

Now let us use previous results to establish the results that the difference between perturbed solution and unperturbed solution is small in the order of $\epsilon_{\max}$.

Theorem 10. *Under the same assumptions as Lemma 6, let $\mathbf{x}^\#$ be the solution to the perturbed problem (5), and $\mathbf{x}^* = \mathbf{1}_{C_1^c} \in \mathbb{R}^{|\Omega|-|T|}$ which is the solution to the unperturbed problem (4). Then*

$$\frac{\|\mathbf{x}^\# - \mathbf{x}^*\|}{\|\mathbf{x}^*\|} = O(\epsilon_{\max}).$$

almost surely for large n_1.

Proof. By Lemma 8, we have $\|M\| \leq 2\epsilon_{\max}$. Therefore

$$\begin{aligned}
\|\tilde{A} - A\| &= \big\|(L_{\Omega\setminus T})^\top L_{\Omega\setminus T} - (L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big\| = \big\|(L^{in}_{\Omega\setminus T})^\top M_{\Omega\setminus T} + M^\top_{\Omega\setminus T} L^{in}_{\Omega\setminus T} + M^\top_{\Omega\setminus T} M_{\Omega\setminus T}\big\| \\
&\leq \big\|(L^{in}_{\Omega\setminus T})^\top M_{\Omega\setminus T}\big\| + \big\|M^\top_{\Omega\setminus T} L^{in}_{\Omega\setminus T}\big\| + \big\|M^\top_{\Omega\setminus T} M_{\Omega\setminus T}\big\| \\
&\leq \big(2\|L^{in}_{\Omega\setminus T}\| + \|M_{\Omega\setminus T}\|\big)\cdot\|M_{\Omega\setminus T}\| \\
&\leq \big(2\|L^{in}_{\Omega\setminus T}\| + \|M\|\big)\cdot\|M\| \leq 4\epsilon_{\max}\cdot\big(\|L^{in}_{\Omega\setminus T}\| + \epsilon_{\max}\big).
\end{aligned}$$

For each $i \in \Omega \setminus T$, we have $\|L_i\| \geq 1$, and $\sigma_{\max}((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}) = \|(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\| = \sigma^2_{\max}(L^{in}_{\Omega\setminus T}) = \|L^{in}_{\Omega\setminus T}\|^2 \geq \max_{i\in\Omega\setminus T}\|L_i\|^2 \geq 1$. Hence

$$\begin{aligned}
\frac{\big\|(L_{\Omega\setminus T})^\top L_{\Omega\setminus T} - (L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big\|}{\big\|(L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big\|} &\leq \frac{\big(2\|L^{in}_{\Omega\setminus T}\| + \|M\|\big)\cdot\|M\|}{\|L^{in}_{\Omega\setminus T}\|^2} \\
&\leq \frac{4\epsilon_{\max}}{\|L^{in}_{\Omega\setminus T}\|} + \frac{4\epsilon^2_{\max}}{\|L^{in}_{\Omega\setminus T}\|^2} \leq 4(\epsilon_{\max} + \epsilon^2_{\max}). \qquad (8)
\end{aligned}$$

We also have

$$\begin{aligned}
\|\tilde{\mathbf{y}} - \mathbf{y}\| &= \|(L_{\Omega\setminus T})^\top (L\mathbf{1}_\Omega) - (L^{in}_{\Omega\setminus T})^\top (L^{in}\mathbf{1}_\Omega)\| = \|(L^{in}_{\Omega\setminus T} + M_{\Omega\setminus T})^\top (L_\Omega\mathbf{1}_\Omega) - (L^{in}_{\Omega\setminus T})^\top (L^{in}_\Omega\mathbf{1}_\Omega)\| \\
&= \|\big((L^{in}_{\Omega\setminus T})^\top M_\Omega + M^\top_{\Omega\setminus T} L^{in}_\Omega + M^\top_{\Omega\setminus T} M_\Omega\big)\cdot\mathbf{1}_\Omega\| \\
&\leq \sqrt{|\Omega|}\cdot\big(\|(L^{in}_{\Omega\setminus T})^\top M_\Omega\| + \|M^\top_{\Omega\setminus T} L^{in}_\Omega\| + \|M^\top_{\Omega\setminus T} M_\Omega\|\big) \\
&\leq \sqrt{|\Omega|}\cdot\big(2\|L^{in}_\Omega\| + \|M_\Omega\|\big)\cdot\|M_\Omega\| \\
&\leq \sqrt{|\Omega|}\cdot\big(2\|L^{in}_\Omega\| + 2\epsilon_{\max}\big)\cdot 2\epsilon_{\max} = 2\sqrt{5n_1}\cdot\big(\|L^{in}_\Omega\| + \epsilon_{\max}\big)\cdot\epsilon_{\max}.
\end{aligned}$$

Next by Lemma 9, $\|(L^{in}_{\Omega\setminus T})^\top L^{in}_\Omega \mathbf{1}_\Omega\| \geq \frac{\sqrt{|\Omega\setminus C_1|}}{2} = \frac{\sqrt{n_1}}{4}$ almost surely, therefore we have

$$\frac{\|(L_{\Omega\setminus T})^\top L\mathbf{1}_\Omega - (L^{in}_{\Omega\setminus T})^\top L^{in}\mathbf{1}_\Omega\|}{\|(L^{in}_{\Omega\setminus T})^\top L^{in}\mathbf{1}_\Omega\|} \leq \frac{2\sqrt{5n_1}\cdot\big(\|L^{in}_\Omega\|+\epsilon_{\max}\big)\cdot\epsilon_{\max}}{\sqrt{n_1}/4} = 8\sqrt{5}\epsilon_{\max}\cdot\big(\|L^{in}_\Omega\|+\epsilon_{\max}\big)$$
$$\leq 8\sqrt{5}\epsilon_{\max}\cdot\big(\sqrt{2}+\epsilon_{\max}\big) = 8\sqrt{10}\epsilon_{\max}+8\sqrt{5}\epsilon^2_{\max}.$$

The second inequality holds because $\sigma_{\max}((L^{in}_\Omega)^\top L^{in}_\Omega) \leq 2$ for the similar reasoning in Lemma 6 by using Gershgorin's circle theorem, so $\|L^{in}_\Omega\| \leq \sqrt{2}$. Now applying Lemma 6 and Lemma 5 with $A = (L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}$, $\tilde{A} = (L_{\Omega\setminus T})^\top L_{\Omega\setminus T}$, $\mathbf{y} = L^{in}\mathbf{1}_\Omega$, $\tilde{\mathbf{y}} = L\mathbf{1}_\Omega$, we have

$$\frac{\|\mathbf{x}^\# - \mathbf{x}^*\|}{\|\mathbf{x}^*\|} \leq \frac{cond\big((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big)\cdot\big(4\epsilon_{\max}+4\epsilon^2_{\max}+8\sqrt{10}\epsilon_{\max}+8\sqrt{5}\epsilon^2_{\max}\big)}{1-cond\big((L^{in}_{\Omega\setminus T})^\top L^{in}_{\Omega\setminus T}\big)\cdot\big(4\epsilon_{\max}+4\epsilon^2_{\max}\big)}$$
$$\leq \frac{16\big((1+2\sqrt{10})\epsilon_{\max}+(1+2\sqrt{5})\epsilon^2_{\max}\big)}{1-16\epsilon_{\max}(1+\epsilon_{\max})} = O(\epsilon_{\max}).$$

□

Next we can estimate the size of the symmetric difference between output $C_1^\#$ and the true cluster C_1 relative to the size of C_1, the symmetric difference is defined as $C_1^\# \triangle C_1 := (C_1^\# \setminus C_1)\cup(C_1\setminus C_1^\#)$. Let us state another lemma before we establish the final result.

Lemma 11. *Let $T\subset[n]$, $\mathbf{v}\in\mathbb{R}^n$, and $W^\# = \{i : \mathbf{v}_i > R\}$. Suppose $\|\mathbf{1}_T - \mathbf{v}\| \leq D$, then $|T\triangle W^\#| \leq \frac{D^2}{R^2}$.*

Proof. Let $U^\# = [n]\setminus W^\#$ and write $\mathbf{v} = \mathbf{v}_{U^\#} + \mathbf{v}_{W^\#}$, where $\mathbf{v}_{U^\#}$ and $\mathbf{v}_{W^\#}$ are the parts of $\mathbf{v}$ supported on $U^\#$ and $W^\#$. Then we can write

$$\|\mathbf{1}_T - \mathbf{v}\|^2 = \|\mathbf{1}_{T\cap W^\#} - (\mathbf{v}_{W^\#})_{T\cap W^\#}\|^2 + \|(\mathbf{v}_{W^\#})_{W^\#\setminus T}\|^2 + \|\mathbf{1}_{T\setminus W^\#} - \mathbf{v}_{U^\#}\|^2.$$

Since $\|(\mathbf{v}_{W^\#})_{W^\#\setminus T}\|^2 \geq R^2\cdot|W^\#\setminus T|$ and $\|\mathbf{1}_{T\setminus W^\#} - \mathbf{v}_{U^\#}\|^2 \geq R^2\cdot|T\setminus W^\#|$, we have

$$R^2\cdot|T\triangle W^\#| = R^2\cdot(|T\setminus W^\#|+|W^\#\setminus T|) \leq \|(\mathbf{v}_{W^\#})_{W^\#\setminus T}\|^2 + \|\mathbf{1}_{T\setminus W^\#} - \mathbf{v}_{U^\#}\|^2 \leq \|\mathbf{1}_T - \mathbf{v}\|^2.$$

Hence $|T\triangle W^\#| \leq \frac{\|\mathbf{1}_T-\mathbf{v}\|^2}{R^2} \leq \frac{D^2}{R^2}$. □

Theorem 12. *Suppose $0.1 \leq R \leq 0.9$. Under the same assumptions as Theorem 10, we have*

$$\frac{|C_1^\#\triangle C_1|}{|C_1|} \leq O(\epsilon^2_{\max}).$$

In other words, the error rate of successfully recovering C_1 is at most a constant multiple of $\epsilon^2_{\max}$.

Proof. From Theorem 10, we have $\|\mathbf{x}^\# - \mathbf{x}^*\| = \|\mathbf{x}^\# - \mathbf{1}_{\Omega\setminus C_1}\| \leq O(\epsilon_{\max})\cdot\|\mathbf{x}^*\| \leq O(\epsilon_{\max}\sqrt{n_1})$. Then by Lemma 11, we get $|W^\#\triangle(\Omega\setminus C_1)| \leq O(\epsilon^2_{\max}n_1)$. Since $C_1^\# = \Omega\setminus W^\#$, it then follows $|C_1^\#\triangle C_1| \leq O(\epsilon^2_{\max}n_1)$, hence $\frac{|C_1^\#\triangle C_1|}{|C_1|} = O(\epsilon^2_{\max})$ as desired. □

3.2 Random Walk Threshold

In order to apply Algorithm 1, we have to have a "good" superset which contains C_1. The task for this subsection is to find such a superset Ω from the given seed vertices Γ. We will apply a simple diffusion based random walk algorithm on G to find such Ω. Note that there are also other sophisticated algorithms such as Andersen et al. (2007), Kloster and Gleich (2014), Wang et al. (2017) to achieve this goal, but we do not analyze them here as our purpose is just to implement a fast way of obtaining a set $\Omega \supset C_1$. This leads to Algorithm 2, note that this algorithm is also described in Lai and Mckenzie (2020), but the difference is that in implementation, we will allow a larger ϵ to decrease the chances of missing any vertices in C_1 based on the natural differences of our approaches.

Algorithm 2 *RandomWalkThreshold*

Require: Adjacency matrix A, a random walk threshold parameter $\epsilon \in (0,1)$, a set of seed vertices $\Gamma \subset C_1$, estimated size $\hat{n}_1 \approx |C_1|$, and depth of random walk $t \in \mathbb{Z}^+$.

- Compute $P = AD^{-1}$ and $\mathbf{v}^0 = D\mathbf{1}_\Gamma$.
- Compute $\mathbf{v}^{(t)} = P^t\mathbf{v}^{(0)}$.
- Define $\Omega = \mathcal{L}_{(1+\epsilon)\hat{n}_1}(\mathbf{v}^{(t)})$.

Ensure: $\Omega = \Omega \cup \Gamma$.

The thresholding operator $\mathcal{L}_s(\cdot)$ is defined as

$$\mathcal{L}_s(\mathbf{v}) := \{i \in [n] : v_i \text{ among } s \text{ largest entries in } \mathbf{v}\}.$$

The motivation of *RandomWalkThreshold* is the following intuitive observation. Suppose we are given seed vertices $\Gamma \subset C_1$, then by starting from Γ, since the edges within each cluster are more dense than those between different clusters, the probability of staying within C_1 will be much higher than entering other clusters C_i, for $i \neq 1$, in a short amount of depth t. Therefore, by performing a random walk up to a certain depth, we will have a well approximated set Ω such that C_1 is almost surely contained in Ω. Let us make this more precisely in Theorem 13.

Theorem 13. *With the input from Algorithm 2, i.e.* $|\Gamma| = O(1)$ *and* $t = O(1)$*, the probability* $\mathbb{P}\big(C_1 \subset \Omega\big) = \mathbb{P}\big(\sum_{j\in C_1}\mathbf{v}_j^{(t)} = \|\mathbf{v}^{(t)}\|_1\big) \geq 1 - O(\epsilon_{\max})$*. In other words, the probability that the* t*-steps random walk with seed vertices* Γ *being not in* C_1 *is at most a constant multiple of* $\epsilon_{\max}$*.*

Proof. Let us first consider the case $|\Gamma| = 1$. Suppose $\Gamma = \{s\}$. Then we have $\mathbb{P}\big(\sum_{j\in C_1}\mathbf{v}_j^{(0)} = \|\mathbf{v}^{(t)}\|_1\big) = \mathbb{P}(\mathbf{v}_s^{(0)} = \|\mathbf{v}^{(t)}\|_1) = 1$. It is also direct to see that $\mathbb{P}\big(\sum_{j\in C_1}\mathbf{v}_j^{(1)} = \|\mathbf{v}^{(t)}\|_1\big) = d_i^{in}/d_i = 1 - \epsilon_i \geq 1 - \epsilon_{\max}$. For $t \geq 2$, we have $\mathbb{P}\big(\sum_{j\in C_1}\mathbf{v}_j^{(t)} = \|\mathbf{v}^{(t)}\|_1\big) \geq (1-\epsilon_{\max}) \cdot \mathbb{P}\big(\sum_{j\in C_1}\mathbf{v}_j^{(t-1)} = \|\mathbf{v}^{(t)}\|_1\big)$. So by supposing $\mathbb{P}\big(\sum_{j\in C_1}\mathbf{v}_j^{(t-1)} = \|\mathbf{v}^{(t)}\|_1\big) \geq (1-\epsilon_{\max})^{t-1} \geq 1 - (t-1)\epsilon_{\max}$, we have $\mathbb{P}\big(\sum_{j\in C_1}\mathbf{v}_j^{(t)} = \|\mathbf{v}^{(t)}\|_1\big) \geq (1-\epsilon_{\max})^t \geq 1 - t\epsilon_{\max} = 1 - O(\epsilon_{\max})$.

Suppose now $|\Gamma| > 1$, we can apply the above argument to each individual vertex in Γ,

where the random walk starting from each vertex can be considered independently, therefore we have $\mathbb{P}\big(\sum_{j\in C_1}\mathbf{v}_j^{(t)} = \|\mathbf{v}^{(t)}\|_1\big) \geq (1-t\epsilon_{\max})^{|\Gamma|} \geq 1 - t\epsilon_{\max}|\Gamma| = 1 - O(\epsilon_{\max})$. □

Remark 14. *It is worthwhile to note that we do not want t to be too large, one reason is that Theorem 13 tells us that the probability of staying within the target cluster C_1 decreases as t increases. An alternative interpretation is that we can treat our graph G, suppose connected, as a time homogeneous finite state Markov chain with evenly distributed transition probability determined by the vertex degree between adjacent vertices. Since G is connected, then G is certainly irreducible and aperiodic. By Theorem 19 in the appendix, we get that the limiting probability of finally being at each vertex will be the same, regardless of what the seed set Γ is. We provide further details about finite state Markov chains in the appendix. Meanwhile, we do not want t to be too small as well, otherwise the random walk will not be able to explore all the reachable vertices. There is a trade-off between the size of Γ and the random walk depth t, where a smaller size of Γ usually induces a larger t in order to fully explore the target cluster.*

3.3 Local Cluster Extraction

Let us now combine the previous two subroutines into our local clustering algorithm *LeastSqureClustering*. In practice, we may want to vary the number of iterations $MaxIter$ based on the number of examples in the data set in order to achieve a better performance. For the purpose theoretical analysis, let us fix $MaxIter = 1$.

Algorithm 3 *LeastSqureClustering*

Require: Adjacency matrix A, a random walk threshold parameter $\epsilon \in (0,1)$, a set of seed vertices $\Gamma \subset C_1$, estimated size $\hat{n}_1 \approx |C_1|$, depth of random walk $t \in \mathbb{Z}^+$, least square parameter $\gamma \in (0, 0.8)$, and rejection parameter $R \in [0,1)$.

- *for* $i = 1, \cdots, MaxIter$
- $\quad \Omega \longleftarrow$ *RandomWalkThreshold(*A, Γ, $\hat{n}_1$, ϵ, t*).*
- $\quad \Gamma \longleftarrow$ *LeastSquareClusterPursuit(*A, Ω, R, γ*).*
- *end*
- Let $C_1^{\#} = \Gamma$.

Ensure: $C_1^{\#}$.

Remark 15. *The hyperparameter $MaxIter$ in the algorithm is usually choosen based on the size of initial seed vertices Γ relative to n, we do not have a formal way of choosing the best $MaxIter$ rather than choose it heuristically. In practice, we believe $MaxIter \leq 3$ will do a very good job mostly.*

The analysis in previous two subsections give that the difference between true cluster C_1 and the estimated $C_1^{\#}$ is relative small compared to the size of C_1, this can be written more formally using the asymptotic notation.

Theorem 16. *Suppose* $\epsilon_{\max} = o(1)$ *and* $MaxIter = 1$*, then under the assumptions of Theorem 12 and 13, we have* $\mathbb{P}\Big(\frac{|C_1^{\#} \triangle C_1|}{|C_1|} \leq o(1)\Big) = 1 - o(1)$.

Proof. By Theorem 13, we have that the probability of $\Omega \supset C_1$ after *RandomWalkThreshold* is $1-O(\epsilon_{\max}) = 1-o(1)$. Then by Theorem 12, the error rate is at most a constant multiple of $\epsilon_{\max}^2$ after *LeastSquareClusterPursuit*. Putting them together, we have $\mathbb{P}\Big(\frac{|C_1^{\#} \triangle C_1|}{|C_1|} \leq o(1)\Big) = 1 - o(1)$. □

3.4 From Local to Global

We can then further apply *LeastSquareClustering* iteratively on the entire graph to extract all the underlying clusters. That is, we remove $C_i^{\#}$ each time after the algorithm finds it, and updates the graph G by removing the subgraph spanned by vertices $C_i^{\#}$ successively. We summarize the algorithm as *IterativeLeastSqureClustering*. However, we will not analyze further the theoretical guarantees of the iterative version the algorithm, but rather provide with numerical examples in the later section to show its efficiency and accuracy.

Algorithm 4 *IterativeLeastSqureClustering*

Require: Adjacency matrix A, random walk threshold parameter $\epsilon \in (0, 1)$, least square parameter $\gamma \in (0, 0.8)$, rejection parameter $R \in [0, 1)$, depth of random walk $t \in \mathbb{Z}^+$. Seed vertices for each cluster $\mathit{\Gamma}_i \subset C_i$, estimated size $\hat{n}_i \approx |C_i|$ for $i = 1, \cdots k$.

- *for* $i = 1, \cdots, k$
- Let $C_i^{\#}$ be the output of *LeastSquareClustering*.
- Let $G^{(i)}$ be the subgraph spanned by $C_i^{\#}$.
- Updates $G \leftarrow G \setminus G^{(i)}$.
- *end*
- Let $C_1^{\#} = \mathit{\Gamma}$.

Ensure: $C_1^{\#}, \cdots, C_k^{\#}$.

4. Computational Complexity

First, note that if A, D, P are stored as sparse matrices, then for each t in the second step of Algorithm 2, the computation will have a complexity $O(nd_{\max})$, where $d_{\max}$ is the maximal degrees among all the vertices. Then the algorithm *RandomWalkThreshold* has a time complexity $O(nd_{\max}t + n\log(n))$, where the $O(n\log(n))$ part comes from the third step of sorting. If we take t to be a $O(1)$ with respect to n, then we have the time complexity $O(nd_{\max} + n\log(n))$.

For algorithm *LeastSquareClusterPursuit*, the first step takes time $O(nd_{\max})$, second step takes time $O(nd_{\max} + n\log(n))$, where the $O(nd_{\max})$ part comes from matrix vector multiplication, and $O(n\log(n))$ part comes from sorting.

Note that the standard way of solving the least square problem in the third step by finding the matrix inverse could cost a lot in computation. However, if an iterative method such as conjugate gradient descent is used (during our implementation, we used the *lsqr* function in Matlab) instead, given that the matrices are associated with well behaved condition numbers, then as pointed out by Needell and Tropp (2009), it requires only a constant number of iterations to get a well approximated least square solution. In the implementation, we apply ten iterations for *lsqr*. Since the cost for each iteration in conjugate gradient descent equals to a few operations of matrix vector multiplication, which is $O(nd_{\max})$, the total cost for *LeastSquareClusterPursuit* is $O(nd_{\max} + n\log(n))$.

As a consequence, the total run time for *LeastSquareClustering* is $O(nd_{\max} + n\log(n))$, and the total run time for *IterativeLeastSquareClustering* is $O(knd_{\max} + kn\log(n))$. Therefore, in the case of $k = O(1)$, the total run time are $O(nd_{\max} + n\log(n))$, whereas the run time in Lai and Mckenzie (2020) is $O(nd_{\max}\log(n))$. For the regime $d_{\max} = O(\omega(n)\log n)$ where $\omega(n) \to \infty$, our algorithm is slightly favored in terms of efficiency, which can also be seen from the numerical examples given in the next section.

5. Numerical Experiments

For single cluster extraction, we compare our algorithm *LeastSquareClustering (LSC)* with its counterpart, the state-of-the-art algorithm *CP+RWT* in Lai and Mckenzie (2020) on the synthetic SSBM model and the network data on political blogs, we also apply the baseline Lasso method as a subroutine to replace the least square step in LSC for comparison as well. For multiple cluster extractions, we compare *IterativeLeastSquareClustering (ILSC)* with its counterpart, algorithm *ICP+RWT* in Lai and Mckenzie (2020) on the MNIST data, AT&T human faces data and YaleB human faces data, we also apply the baseline Lasso method as a subroutine to replace the least square step in ILSC and implement the standard spectral clustering (SC) algorithm as another baseline method for comparison as well. For the implementation of LSC (or ILSC), we use Matlab *lsqr* function as our iterative least square solver to solve equation (5). For the implementation of CP+RWT (or ICP+RWT) we replace (5) in step 3 of Algorithm 1 by solving (6) the same way as Lai and Mckenzie (2020). For the implementation of Lasso method, we solve (7) using the standard Matlab Lasso solver as a subroutine. We tune the rejection parameters R for all algorithms and regularized parameter λ in Lasso appropriately to make the output $C_i^{\#}$ of each algorithm approximately the same size for comparison purpose. Further implementation details are given in the appendix.

5.1 Synthetic Data

We first test our algorithms on the symmetric stochastic block model $SSBM(n, k, p, q)$ with different choices of parameters. The paramenter n indicates the total number of vertices, k indicates the number of clusters, p is the probability of having an edge between any two vertices within each cluster, and q is the probability of having an edge between any two vertices from different clusters. Figure 1 gives an illustration of such a synthetic random graph model with three underlying clusters. By tuning the parameters, we achieve the

experimental results shown in Table 1. Meanwhile, we also run the experiments on non-symmetric stochastic block model and obtained gaps in accuracy and run time for these algorithms similar to Table 1. For the implementation of symmetric stochastic block model, we use three vertices with given label as our seed vertices under 100 repetitions, and we focus on only recovering the first cluster C_1.

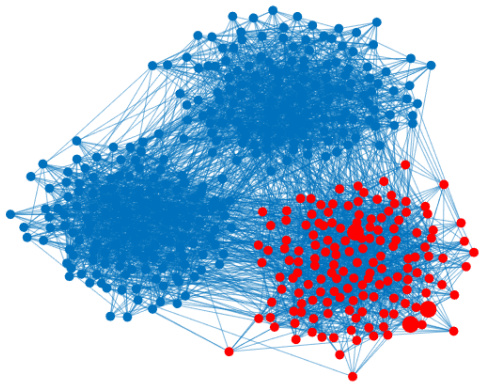

Figure 1: A Symmetric Stochastic Block Random Graph with Three Underlying Clusters.

n	p	q	LSC	Time	CP+RWT	Time	LASSO	Time
300	0.1	0.005	99.8%	4.7 ms	99.6%	17.7 ms	99.2%	13.7 ms
300	0.1	0.01	97.7%	4.6 ms	95.1%	13.5 ms	96.6%	14.7 ms
1200	0.1	0.01	100%	35.9 ms	99.9%	52.4 ms	99.9%	117.2 ms
1200	0.1	0.02	99.6%	37.6 ms	97.4%	42.9 ms	99.6%	126.8 ms
4800	0.1	0.01	100%	0.37 s	100%	0.43 s	100%	1.65 s
4800	0.1	0.03	99.9%	0.39 s	99.4%	0.50 s	99.9%	1.69 s
4800	0.1	0.035	98.6%	0.39 s	92.7%	0.44 s	98.6%	1.70 s
19200	0.1	0.01	100%	4.30 s	100%	9.12 s	100%	28.2 s
19200	0.1	0.04	99.9%	6.70 s	99.3%	11.35 s	99.9%	30.6 s
19200	0.1	0.045	99.0%	7.71 s	93.8%	12.40 s	99.0%	33.6 s

Table 1: Performance of finding C_1 using LSC, CP+RWT and LASSO with $k = 3$.

5.2 Network Data

We test our algorithms on the data from "The political blogosphere and the 2004 US Election"Adamic and Glance (2005), which contains a list of political blogs that were classified as liberal or conservative, and links between the blogs. As explained by Abbe and Sandon (2015), the simplified version of thier algorithm gave a reasonably good classification 37 times out of 40 trials. Each of these trials classified all but 56 to 67 of the 1222 vertices in the graph main component correctly. According to Abbe and Sandon (2015), the state-of-the-art described in Zhang et al. (2015) before the work in Abbe and Sandon (2015) gives a

lowest value at 58, with the best algorithms around 60 while algorithms regularized spectral methods such as the one in Qin and Rohe (2013) obtain about 80 errors.

In our experiments, with 8 seeds and out of 40 trials, our algorithm *LSC* gave a good classification 36 times using the cut-off upper bound 68 misclassified vertices. Each trial classified all but 19 to 68 of the 1222 vertices in 36 trials while the other 4 trials found 74, 74, 79, and 110 misclassifed vertices. If a tolerance is set at at most misclassifed 74 out of 1220 vertices, our algorithm achieved successes 38 out of 40 trials. The accuracy of all 40 trials are also shown in Table 2.

# of Misclassified by LSC	# of Misclassified by CP+RWT	# of Misclassified by LASSO
19	147	73
21	150	73
24	162	79
27	163	80
28	166	81
29	167	85
29	167	88
30	167	97
30	169	98
31	171	99
31	174	100
33	174	101
34	175	102
35	177	102
35	180	103
35	180	104
36	181	109
36	184	116
36	188	116
38	190	121
39	191	121
40	191	122
44	193	122
45	197	124
45	201	125
46	201	126
48	201	126
49	202	133
53	214	142
53	226	156
55	233	158
56	235	162
61	236	164
61	236	165
63	236	167
68	248	183
74	254	188
74	255	189
79	275	199
110	289	207
Average Accuracy 91.2%	Average Accuracy 68.2%	Average Accuracy 79.0%

Table 2: Number of Misclassified Vertices and Overall Accuracy

5.3 MNIST Data

We test our algorithms on the MNIST data, which is a famous machine learning benchmark dataset in classification that consists of 70000 grayscale images of the handwritten digits 0-9 of size 28×28 with approximately 7000 images of each digit. Figure 2 shows a sample of the data set.

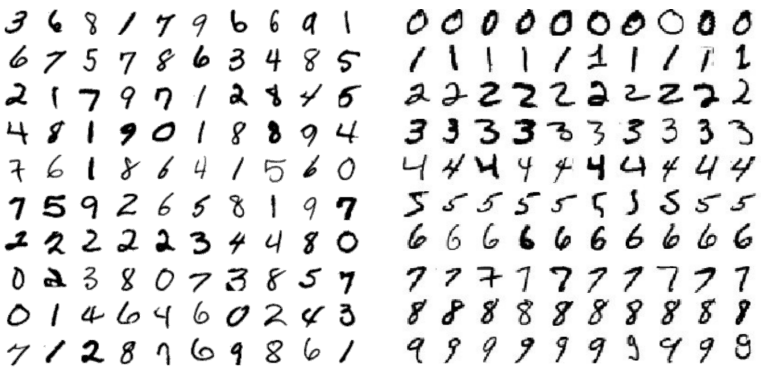

Figure 2: An Experiment on MNIST Data before (on the left) and after (on the right).

We used a certain percentage of vertices with given label as our seed vertices. The performance ILSC and ICP+RWT are summarized in Tables 3 under 100 repetitions.

Labeled Data %	ILSC	Time	ICP+RWT	Time
0.5	97.30%	15.5 s	96.41%	18.1 s
1	97.73%	15.3 s	97.32%	19.1 s
1.5	98.03%	15.4 s	97.44%	19.8 s
2	98.17%	15.5 s	97.52%	21.4 s
2.5	98.27%	15.4 s	97.50%	22.1 s

Table 3: Performance of ILSC and ICP+RWT for Finding All Clusters in MNIST with Labeled Data.

We also compare our algorithm with several other state-of-the-art semi-supervised methods on MNIST. As we can see in Table 4, ILSC outperforms the other algorithms except for the Ladder Networks which uses more information of labels and involved in a deep neural network architecture that requires training on GPUs for hours.

5.4 Human Faces Data

The ability of being successfully make human faces images into clusters is also a good measure to demonstrate the effectiveness of our algorithm. We implemented AT&T and YaleB human faces images for the illustration.

Methods	Labeled Data	Accuracy
LapRF (Yin and Tai, 2018)	600	95.6%
TVRF (Yin and Tai, 2018)	600	96.8%
ICP+RWT (Lai and Mckenzie, 2020)	700	97.3%
Multi-Class MBO with Auction Dynamics (Jacobs et al., 2018)	700	97.4%
ILSC (this paper)	700	97.7%
AtlasRBF (Pitelis et al., 2014)	1000	96.4%
Pseudo-label (Lee, 2013)	1000	96.6%
DGN (Kingma et al., 2014)	1000	97.6%
ILSC (this paper)	1000	98.0%
Ladder Networks (Rasmus et al., 2015)	1000	99.2%

Table 4: Comparing ILSC to other State-of-the-Art Semi-supervised Algorithms on MNIST.

5.4.1 AT&T Faces Data

The AT&T "The Database of Faces[1]" human faces data, which contains gray scale images for 40 different people of pixel size 56×46. The images of each person are taken under 10 different conditions, by varying the three perspectives of faces, lighting conditions, and facial expressions. Figure 3 shows part of this data set.

Figure 3: An Experiment on the AT&T Faces (before (on the left) and after (on the right))

We use part of this data set by randomly selecting 10 people such that each individual has 10 images, then we randomly permute the images as shown in the left of Figure 3, compute its adjacency matrix based on the preprocessing method summarized in the appendix. Then we apply Algorithm 4 which tries to recover all the 10 images which belong to the same individual in each row. The expected output after Algorithm 4 are shown on the right of the

1. "The Database of Faces", AT&T Laboratories Cambridge, (2002) Available: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Figure 3. The performances of our algorithm compared with other methods are summarized in Table 5 under 100 repetitions. Note that the spectral clustering method is unsupervised, hence its accuracy does not affected by the percentage of labeled data.

Labeled Data %	10	20	30
ILSC	94.8%	97.7%	98.2%
ICP+RWT	94.3%	97.4%	98.7%
LASSO	94.2%	96.2%	96.6%
SC	95.8%	95.8%	95.8%

Table 5: Average Accuracy of Recovering All Clusters for AT&T Data.

5.4.2 Yale Faces Data

The "Extended Yale Face Database B (YaleB)" data, refering to Georghiades et al. (2001), contains 16128 gray scale images of 28 human subjects under 9 poses and 64 illumination conditions. We use part of this data set by randomly selecting 20 images from each person after some data preprocessing. We first randomly permute the images as shown in the left side of Figure 4, then we aim to recover all the 20 images which belong to the same individual into each row, as shown in the right side of Figure 4. The performances of our algorithm compared with others are summarized in Table 6 under 100 repetitions.

Labeled Data %	10	20	30
ILSC	96.0%	96.2%	96.3%
ICP+RWT	94.4%	94.7%	94.9%
LASSO	93.4%	93.5%	93.5%
SC	93.8%	93.8%	93.8%

Table 6: Average Accuracy of Recovering All Clusters for YaleB Data.

To sum up, we proposed a semi-supervised local cluster extraction algorithm *LeastSquareClustering (LSC)* and its iterative version *IterativeLeastSquareClustering (ILSC)*. The *LSC* is obtained through a two stages approach. In the first stage, we found a superset of the target cluster which will almost certain to contain the seed vertices set by running *RandomWalkThreshold* on the seed vertices. In the second stage, we developed a least square based approach *LeastSquareClusterPuruist* as our pursuit step to find the complement of target cluster within the superset. Finally, we validate our model by testing it on SSBM, political blog, MNIST, AT&T human faces data, and YaleB human faces data. Our algorithm *LSC* and *ILSC* achieves a better performance than their counterparts *CP+RWT* and *ICP+RWT* in Lai and Mckenzie (2020) both in accuracy and efficiency. It also achieves very competitive results compared with other state-of-the-art semi-supervised clustering algorithms.

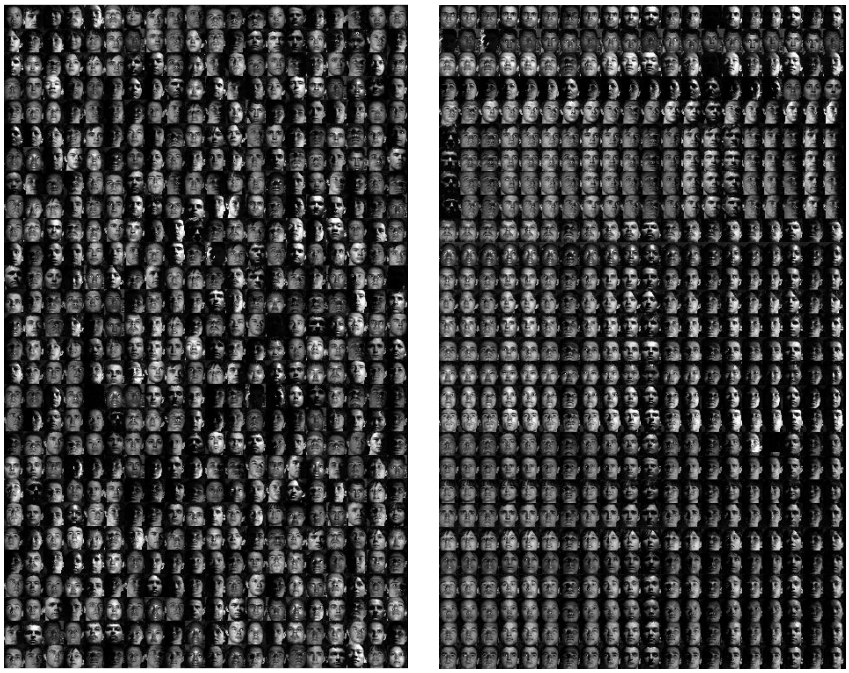

Figure 4: An Experiment on the Extended YaleB Data (before (on the left) and after (on the right))

Acknowledgements

The authors want to thank Dr.Daniel Mckenzie[2] for his kindness of sharing his code for our comparison implementation.

2. mckenzie@math.ucla.edu. Department of Mathematics, University of California, Los Angeles, CA 155505.

Appendix A.

For completeness, we briefly introduce the concept of Markov chains and state the fundamental theorem of Markov chains in this appendix.

For a finite state space Ω, we say a sequence of random variables X_t on Ω is a *Markov chain* if for all t and all $x_0, \cdots, x_t, y \in \Omega$, we have

$$\mathbb{P}(X_{t+1} = y | X_0 = x_0, \cdots, X_t = x_t) = \mathbb{P}(X_{t+1} = y | X_t = x_t).$$

In other words, the probability of being at the next state is only determined by the immediate previous state. We consider time homogeneous one step transition matrix as

$$P(x, y) = \mathbb{P}(X_{t+1} = y | X_t = x).$$

The t-step transition matrix is naturally defined as

$$P^t(x, y) = \begin{cases} P(x, y) & t = 1, \\ \sum_{z \in \Omega} P(x, z) P^{t-1}(z, y) & t > 1. \end{cases}$$

We say a distribution π over Ω is a stationary distribution if it is invariant with respect to the transition matrix, i.e.,

$$\pi(y) = \sum_{x \in \Omega} \pi(x) P(x, y), \quad \text{for all} \quad y \in \Omega.$$

Definition 17. *A Markov chain is called* irreducible *if for all $x, y \in \Omega$, there exists t such that $P^t(x, y) > 0$.*

Definition 18. *A Markov chain is called* aperiodic *if for all $x \in \Omega$, $gcd\{t : P^t(x, x) > 0\} = 1$.*

The following theorem, originally proved in Doeblin (1938), details the essential property of irreducible and aperiodic Markov chains.

Theorem 19. *(Fundamental Theorem of Markov Chains) For a finite irreducible and aperiodic Markov Chain, there exists a unique stationary distribution π such that*

$$\lim_{t \to \infty} P^t(x, y) = \pi(y) \quad \textit{for all} \quad x, y \in \Omega.$$

Proof. We omit the proof here by referring to Doeblin (1938), Aldous (1983) and Lindvall (1992) for interested readers. □

Appendix B.

We provide with some more specific details in this appendix for implementing the numerical experiments in the previous sections.

Parameters Tuning

For each cluster to be recovered, we sampled the seed vertices Γ_i uniformly from C_i during all implementations. We fix the random walk depth with $t = 3$ and we use random walk threshold parameter $\epsilon = 0.8$ for implementing MNIST data and political blogs data, and use $\epsilon = 0.6$ for implementing SSBM, ATT, and YaleB data. We vary the rejection parameters $R \in (0, 1)$ for each specific experiments based on the estimated sizes of clusters. In the case of no knowledge of estimated sizes of clusters nor the number of clusters are given, we may have to refer to the spectra of graph Laplacian and use the large gap between two consecutive spectrum to estimate the number of clusters, and then use the average size to estimate the size of each cluster.

We fix the least square threshold parameter with $\gamma = |\hat{n}_1|/5$, which is totally heuristic. However, the performance of algorithms will not perturbed too much by varying $\gamma \in (0, 0.5)$, as long as we do not push γ too extremely. The hyperparameter $MaxIter$ is choosen according to the size of initial seed vertices relative to the total number of vertices in the cluster. For purely comparison purpose, we keep $MaxIter = 1$ for MNIST data. By experimenting on different choices of $MaxIter$, we implement with $MaxIter = 1$ for AT&T data and $MaxIter = 2$ for YaleB data which give the best results.

Image Data Preprocessing

We performed some data preprocessing techinuqes on YaleB data to avoid the poor quality images. Specifically, we abandoned the pictures which are too dark, and we cropped each image into size of 54×46 to reduce the effect of background noise. For the remaining qualified pictures, we randomly selected 20 images for each person.

All the image data in MNIST, AT&T, YaleB needs to be firstly constructed into an auxiliary graph before the implementation. Let $\mathbf{x}_i \in \mathbb{R}^n$ be the vectorization of each image image from the original data set, we define the following affinity matrix of the K-NN auxiliary graph Jacobs et al. (2018) Zelnik-Manor and Perona (2004) based on Gaussian kernel.

$$A_{ij} = \begin{cases} e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma_i\sigma_j} & \text{if} \quad \mathbf{x}_j \in NN(\mathbf{x}_i, K) \\ 0 & \text{otherwise} \end{cases}$$

The notation $NN(\mathbf{x}_i, K)$ is the set of K-nearest neighbours of $\mathbf{x}_i$, and $\sigma_i = \|\mathbf{x}_i - \mathbf{x}_i^{(r)}\|$ where $\mathbf{x}_i^{(r)}$ is the r-th closest point of $\mathbf{x}_i$. Note that the above A_{ij} is not necessary symmetric, so we consider $\tilde{A}_{ij} = A^T A$ for symmetrization. Alternatively, one may also want to consider $\tilde{A} = \max\{A_{ij}, A_{ji}\}$ or $\tilde{A} = (A_{ij} + A_{ji})/2$. We then use $\tilde{A}$ as the input adjacency matrix for our algorithms.

We fix the local scaling parameters $K = 15$, $r = 10$ for the MNIST data, $K = 8$, $r = 5$ for the YaleB data, and $K = 5$, $r = 3$ for the AT&T data during implementation.

References

E. Abbe and C. Sandon. Recovering communities in the general stochastic block model without knowing the parameters. *In Advances in Neural Information Processing Systems*, pages 676–684, 2015.

L. A. Adamic and N. Glance. The political blogosphere and the 2004 us election: Divided they blog. *In Proceedings of the 3rd International Workshop on Link Discovery*, pages 36–43, 2005.

D. J. Aldous. Random walks on finite groups and rapidly mixing markov chains. *In Seminaire de Probabilites*, XVII:243–297, 1983.

R. Andersen, F. Chung, and K. Lang. Using pagerank to locally partition a graph. *Internet Mathematics*, 4(1):35–64, 2007.

G. Camps-Valls, T. Bandos, and D. Zhou. Semisupervised graph-based hyperspectral image classification. *IEEE Trans. Geosci. Remote Sensing*, 45(10):3044–3054, 2007.

Y. Chen, J. Z. Wang, and R. Krovetz. Clue: Cluster-based retrieval of images by unsupervised learning. *IEEE Trans. Image Process*, 14(8):1187–1201, 2005.

F. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

F. Chung and L. Lu. *Complex Graphs and Networks*. American Mathematical Society, 2006.

W. Dai and O. Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Trans. Inform. Theory*, 55(5):2230–2249, 2009.

S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. *In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274, 2001.

S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. *In Proc. of the 10th ACM SIGKDD Conference*, 2004.

C. Ding, X. He, H. Zha, M. Gu, , and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. *In Proceedings of IEEE ICDM 2001*, pages 107–114, 2001.

W. Doeblin. Expose de la theorie des chaines simples constantes de markov a un nombre fini d'etats. *Mathematique de l'Union Interbalkanique*, 2:77–105, 1938.

P. Erdős and A. Rényi. On random graphs I. *Publicationes Mathematicae*, 6:290–297, 1959.

S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *PAMI*, 23:643–660, 2001.

W. Ha, K. Fountoulakis, and M. W. Mahoney. Statistical guarantees for local graph clustering. *The 23rd International Conference on Artificial Intelligence and Statistics*, 2020.

P. W. Holland, K. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps, social networks. 5(2):109–137, 1983.

D. Hric, R. K. Darst, and S. Fortunato. Community detection in networks: Structural clusters versus ground truth. *Physical Review E*, 9(062805), 2014.

M. Jacobs, E. Merkurjev, and S. Esedoglu. Auction dynamics: A volume constrained mbo scheme. *Journal of Computational Physics*, 354:288–310, 2018.

D. P. Kingma, S. Mohamed, D. J. Rezende, and Max Welling. Semi-supervised learning with deep generative models. *In Advances in Neural Information Processing Systems*, 27 (NIPS2014):3581–3589, 2014.

K. Kloster and D. F. Gleich. Heat kernel based community detection. *In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1386–1395, 2014.

G. Kossinets and D. J. Watts. Empirical analysis of an evolving social network. *Science*, 311(5757):88–90, 2016.

B. Kulis, S. Basu, I. Dhillon, and R. Mooney. Semi-supervised graph clustering: A kernel approach. *Proc. ICML*, 2015.

M.-J. Lai and D. Mckenzie. Compressive sensing approach to cut improvement and local clustering. *SIAM Journal on Mathematics of Data Science*, 2:368–395, 2020.

D-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. in workshop on challenges in representation learning. *ICML*, 2013.

T. Lindvall. *Lectures on the Coupling Method*. John Wiley & Sons Inc, New York, NY, 1992.

U. V. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

R. Mihalcea and D. Radev. *Graph-based Natural Language Processing and Information Retrieval*. Cambridge university press, 2001.

D. Needell and Joel A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

A. Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *In Advances in Neural Information Processing Systems*, pages 849–856, 2002.

N. Pitelis, C. Russell, and L. Agapito. Semi-supervised learning using an unsupervised atlas. *In Machine Learning and Knowledge Discovery in Databases*, pages 565–580, 2014.

T. Qin and K. Rohe. Regularized spectral clustering under the degree-corrected stochastic blockmodel. *Advances in Neural Information Processing Systemss*, 26:3120–3128, 2013.

A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. *In Advances in Neural Information Processing Systems*, pages 3546–3554, 2015.

F. Santosa and W. Symes. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1307–1330, 1986.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (methodological). Wiley*, 58(1):267–288, 1996.

N. Veldt, C. Klymko, and D. F. Gleich. Flow-based local graph clustering with better seed set inclusion. *In Proceedings of the SIAM International Conference on Data Mining*, 2019.

D. Wang, K. Fountoulakis, M. Henziger, Michael W. Mahoney, and S. Rao. Capacity releasing diffusion for speed and locality. *Proceedings of the 34th International Conference on Machine Learning*, 70:3598–3607, 2017.

Y. Yan, Y. Bian, D. Luo, D. Lee, and X. Zhang. Constrained local graph clustering by colored random walk. *in Proc. World Wide Web Conf*, pages 2137–2146, 2019.

H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich. Local higher-order graph clustering. *In Proceedings of the 23rd ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 555–564, 2017.

K. Yin and X.-C. Tai. An effective region force for some variational models for learning and clustering. *Journal of Scientific Computing*, 74:175–196, 2018.

L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. *In Advances in neural information processing systems*, pages 1601–1608, 2004.

A. Y. Zhang, H. H. Zhou, C. Gao, and Z. Ma. Achieving optimal misclassification proportion in stochastic block model. *arXiv:1505.03772*, 2015.