A Divide and Conquer Framework for Distributed Graph Clustering

Wenzhuo Yang A0096049@NUS.EDU.SG
Department of Mechanical Engineering, National University of Singapore, Singapore 117576

Huan Xu MPEXUH@NUS.EDU.SG
Department of Mechanical Engineering, National University of Singapore, Singapore 117576

Abstract

Graph clustering is about identifying clusters of closely connected nodes, and is a fundamental technique of data analysis with many applications including community detection, VLSI network partitioning, collaborative filtering, etc. In order to improve the scalability of existing graph clustering algorithms, we propose a novel divide and conquer framework for graph clustering, and establish theoretical guarantees of exact recovery of the clusters. One additional advantage of the proposed framework is that it can identify small clusters – the size of the smallest cluster can be of size $o(\sqrt{n})$, in contrast to $\Omega(\sqrt{n})$ required by standard methods. Extensive experiments on synthetic and real-world datasets demonstrate the efficiency and effectiveness of our framework.

1. Introduction

Graph clustering is a widely-used tool for exploiting the relationships between nodes connected in a graph. Such graph usually consists of a large number of nodes, e.g., the friendship graph on Facebook, the co-author graph in bibliographic records. The pairwise connection between two nodes indicates their similarity or affinity, whereas the disconnection represents their dissimilarity or difference. The goal of graph clustering is to partition the nodes into clusters so that the nodes within the same cluster have more connections than those in different clusters, in other words, the closely connected nodes are grouped together.

Various kinds of graph models have been proposed and studied in literature (e.g., Holland et al., 1983; Watts & Strogatz, 1998; Goldenberg et al., 2010), among which the stochastic block model (Holland et al., 1983), also known as the planted partition model (Condon & Karp, 2001), is arguably the most natural one for clustering. The stochastic block model assumes that the nodes are partitioned into several clusters and the probability of the connection between each pair of nodes depends on their cluster membership, i.e., an edge is generated with probability p if they are in the same cluster or with probability q if they are in different clusters with $p > q$, and the goal is to recover the ground truth clusters from the observed graph. Many graph clustering algorithms have been analyzed under this model. One of the most popular approaches is spectral clustering (Luxburg, 2007). Rohe et al. (2011) analyzed the performance of spectral clustering under the stochastic block model and provided an upper bound of the misclassified nodes. Another approach developed recently and has attracted great attention is based on convex optimization using trace-norm and l_1 norm as a surrogate of the rank and sparsity functions respectively (e.g., Oymak & Hassibi, 2011; Jalali et al., 2011; Chen et al., 2014b; Ames & Vavasis, 2011; Chen et al., 2014a). The main advantages of the trace-norm based clustering are their theoretically guarantees for the exact recovery of the ground-truth clustering and their outstanding empirical performance. Indeed, one of them, proposed by Chen et al. (2014b), can provably recover the ground-truth clustering even in the face of outlier nodes (nodes that are not in any cluster) and can be used to solve planted clique and planted coloring problems.

However, both spectral clustering and trace-norm based clustering algorithms have difficulties tackling very large graphs. Indeed, for a graph with n nodes, both approaches need to compute the spectral decomposition of a $n \times n$ matrix which requires $O(n^3)$ operations and $O(n^2)$ storage. Hence it is impractical to apply them even on medium-sized graphs with 10,000 nodes on a desktop PC. How to reduce the memory and computational cost of these methods to improve their scalability is undoubtedly a significant problem, as real-world applications often involve graphs with tens of thousands to even millions of nodes (Günnemann et al., 2010; Macropol & Singh, 2010; Whang et al., 2012). To address this, one approach is to perform

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

a multilevel clustering that reduces the size of the graph by collapsing vertices and edges, partitions the smaller graph and then uncoarsens to recover clusters in the original graph (Karypis & Kumar, 1998a;b; Yan et al., 2009; Chen et al., 2011; Liu et al., 2013). Another approach is to apply the Nyström method to approximate the similarity matrix for acceleration (Fowlkes et al., 2004; Drineas & Mahoney, 2005). While these approaches are empirically evaluated by the experiments on synthetic or real-world datasets, it is not clear whether any theoretical guarantees on *exact recovery* of the ground truth can be obtained. Moreover, it is hard to extend them to improve the scalability of trace-norm based clustering algorithms.

The goal of this paper is to make trace-norm based clustering algorithms applicable for clustering large graphs. Specifically, we develop and analyze a novel divide and conquer framework for improving the scalability of a wide range of graph clustering algorithms including spectral clustering and trace-norm based clustering. Our framework divides the original graph into several easily handled subgraphs, executes a selected graph clustering algorithm on the subgraphs in parallel, and then combines the results using a new algorithm called "graph clustering with high confidence nodes". Our analysis provides sufficient conditions on when the proposed framework can exactly recover the ground-truth clustering. Moreover, applying our framework on trace-norm based clustering algorithms offers additional benefits for identifying small clusters – the size of the smallest cluster can in fact be smaller than $O(\sqrt{n})$.

Notations. We use lower-case boldface letters to denote column vectors and upper-case boldface letters to denote matrices. The minimum of an empty set is defined as $+\infty$. Three matrix norms are used: $\|\mathbf{M}\|_2$ is the spectral norm, $\|\mathbf{M}\|_*$ is the nuclear norm, and $\|\mathbf{M}\|_1$ is the element-wise ℓ_1 norm. We use $[r]$ to denote the set $\{1, 2, \cdots, r\}$ for integer r, and $|\mathcal{S}|$ to denote the cardinality of set $\mathcal{S}$. For a set of matrix indices Ω, we let $\mathcal{P}_\Omega(\mathbf{X})$ be the matrix by setting the entries outside Ω to zero.

2. Divide and Conquer for Graph Clustering

We first provide a brief introduction about the generalized stochastic blockmodel (Chen et al., 2014b) which is an extension of the standard stochastic block model (Holland et al., 1983; Condon & Karp, 2001), and then present our graph clustering framework in details.

2.1. Generalized Stochastic Blockmodel (GSBM)

We consider a random graph with n nodes. These n nodes are divided into two sets $\mathcal{V}_1$ and $\mathcal{V}_2$ which have n_1 inlier nodes and n_2 outlier nodes, respectively. The n_1 inlier nodes are partitioned into r disjoint clusters, which we will refer to as the true clusters. The sizes of these r clusters are $K_1, \cdots, K_r$. Let $K = \min_{i\in[r]} K_i$, i.e., the minimum cluster size. The edges of the graph are generated as follows: For each pair of nodes i, j that belong to the same cluster, edge (i, j) exists with probability at least p; for each pair of nodes that are in different clusters the edge exists with probability at most q. The n_2 outlier nodes are loosely connected to other nodes, namely, edge (i, j) exists with probability at most q for outlier node i and each node $j \neq i$. The adjacency matrix of the graph is denoted by $\mathbf{A}$. Our goal is to recover, from $\mathbf{A}$, the adjacency matrix $\mathbf{Y}^*$ of the ground-truth clusters, i.e., 1) $Y^*_{ii} = 1$ if node i is an inlier node and $Y^*_{ii} = 0$ otherwise, 2) for $i \neq j$, $Y^*_{ij} = 1$ if i, j are in the same cluster and equals 0 otherwise.

2.2. Intuition

Before formally presenting the proposed framework, we explain its intuition first. Our framework involves three steps: 1) dividing step: uniformly randomly partition the n nodes into m groups and then construct the corresponding m subgraphs of $\mathbf{A}$, 2) clustering step: recover clusters in each subgraph in parallel *using some graph clustering algorithm*, and 3) combining step: construct the final clustering by *combining* the recovered clusters in the subgraphs.

We need to answer four questions to make the above procedure concrete: 1) How does m – the number of subgraphs – affect the performance of this framework? 2) How to choose the graph clustering algorithm used in the "clustering step"? 3) How to "combine" the recovered clusters from the subgraphs? 4) Under what conditions can we recover the ground-truth clustering via this framework?

Question 1, 2 and 4 will be studied in Section 3. We now focus on Question 3. For convenience, we suppose that the nodes in the graph are labeled by consecutive integers $\{1, \cdots, n\}$. We denote the m subgraphs by $\{g_1, \cdots, g_m\}$ and the set of the clusters recovered in subgraph g_i by $\mathcal{S}_{g_i}$. To illustrate our main idea, we take the graph shown in Figure 1 containing 12 nodes and two clusters as an example. Suppose that this graph is divided into two subgraphs g_1 and g_2 in the "dividing step" as shown in Figure 1(b), and the clusters in each subgraph are recovered by a certain algorithm, i.e., $\mathcal{S}_{g_1} = \{\{1, 2, 3\}, \{7, 8, 9\}\}$ and $\mathcal{S}_{g_2} = \{\{4, 5, 6\}, \{10, 11, 12\}\}$, in the "clustering" step.

Now we discuss how to combine these clusters. For clusters $\mathcal{U}$ and $\mathcal{V}$ which may or may not belong to the same subgraph, let $\mathcal{U} \leftrightarrow \mathcal{V}$ represent that the nodes in $\mathcal{U}$ and $\mathcal{V}$ belong to the same cluster, and $\mathcal{U} \not\leftrightarrow \mathcal{V}$ denote otherwise. Under the GSBM, we observe that for the bipartite graph generated by nodes in $\mathcal{U}$ and in $\mathcal{V}$, the expected edge density is at least p if $\mathcal{U} \leftrightarrow \mathcal{V}$, while it is at most q if $\mathcal{U} \not\leftrightarrow \mathcal{V}$. Therefore, if the edge density is greater than some threshold t where $q \leq t \leq p$, we have $\mathcal{U} \leftrightarrow \mathcal{V}$ w.h.p. In our

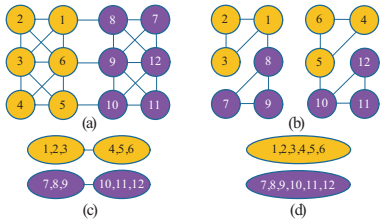

Figure 1. A simple graph with 12 nodes and two clusters YELLOW and PURPLE. (a) The observed graph. (b) Two subgraphs generated in the "dividing" step. (c) The fused graph. (d) The final recovered clusters.

example, the edge density corresponding to $\{7,8,9\}$ and $\{10,11,12\}$ is 0.56, while the edge density corresponding to $\{7,8,9\}$ and $\{4,5,6\}$ is 0.11, so select $t = 0.3$ and we conclude that w.h.p $\{7,8,9\} \leftrightarrow \{10,11,12\}$ and $\{7,8,9\} \nleftrightarrow \{4,5,6\}$. Of course, estimating $\leftrightarrow$ and $\nleftrightarrow$ relationship based on edge density is not always correct, and may even lead to results contradicting each other. Therefore, instead of naively merging clusters based on the edge density, we build a new (meta-)graph whose nodes correspond to the clusters in $\mathcal{S}_{g_1} \cup \mathcal{S}_{g_2}$ and edges are formed according to the edge density between two clusters, i.e., an edge is constructed between them if their edge density is greater than threshold t. We call this graph "fused graph" and call its nodes "super nodes", as shown in Figure 1(c). Then the final clustering result can be obtained by performing a certain graph clustering algorithm on this fused graph, which is shown in Figure 1(d).

There is an interesting property of these super nodes: each edge/no-edge connecting with a super node is very likely to be correct, because it is estimated by averaging many edges in the original graph (recall that a super-node corresponds to a cluster in a subgraph). We call a node with such property a *high confidence node*, formally defined as follows:

Definition 1. *For a graph with n nodes, node i is called a high confidence node if 1) for each node j in the same cluster as i, edge (i,j) exists with probability at least τ_1, and 2) for each node j in different clusters from i, edge (i,j) exists with probability at most $1-\tau_2$, where τ_1, τ_2 satisfies $\min\{\tau_1,\tau_2\} \geq 0.8$ and $1-\min\{\tau_1,\tau_2\} \leq c_\tau \frac{K}{n}$ for some constant c_τ. Node i is an ordinary node if it is a not high confident.*

To cluster the fused graph, we used a variant of the trace-norm based optimization approach for graph clustering. In particular, since we know that the super-nodes are of high confidence, we put more weights on the penalty of the edge disagreement related to high confidence nodes:

$$\begin{aligned} \min_{\mathbf{Y},\mathbf{S}} \quad & \|\mathbf{Y}\|_* + c_{\mathcal{A}}\|P_{\mathcal{A}\cap\mathcal{C}^c}\mathbf{S}\|_1 + c_{\mathcal{A}^c}\|P_{\mathcal{A}^c\cap\mathcal{C}^c}\mathbf{S}\|_1 + c_{\mathcal{C}}\|P_{\mathcal{C}}\mathbf{S}\|_1 \\ \text{s.t.} \quad & \mathbf{Y}+\mathbf{S}=\mathbf{A},\ 0\leq\mathbf{Y}\leq 1, \end{aligned} \tag{1}$$

where $c_{\mathcal{A}}, c_{\mathcal{A}^c}, c_{\mathcal{C}}, c_{\mathcal{C}^c}$ are parameters whose values are determined later, $\mathcal{A} \triangleq \{(i,j) : A_{ij} = 1\}$ and $\mathcal{C} \triangleq \{(i,j) : i \text{ or } j \text{ is a high confidence node}\}$. The solution $\mathbf{Y}$ is the clustering relationship we recover, and $\mathbf{S}$ is the disagreement between the observation and the recovered clustering. Thus, when we set $c_{\mathcal{C}}$ large, $\mathbf{Y}$ and $\mathbf{A}$ are less likely to disagree on edges/no-edges connecting to high confidence nodes. We analyze this formulation in Section 3 and show that it offers important benefits.

2.3. Algorithm

Based on the intuitive idea discussed in the previous section, we design the following framework shown in Algorithm 1 for distributed graph clustering. Clearly, if a cluster from a subgraph is too small, it may not be truly high confident. Hence, we break up the clusters with size less than some threshold T into single nodes, so that only the large clusters form high confidence nodes. Moreover, as we will show in the next section, high confidence nodes can assist in achieving the exact recovery of clustering, so we divide each large cluster recovered in subgraphs into l parts to generate more high confidence nodes for more help for recovering clusters in the fused graph.

Algorithm 1 Large Graph Clustering

Input: Graph clustering algorithm $\mathfrak{A}$, observed adjacency matrix $\mathbf{A}$ and parameter m, l, t, T.

Procedure:

1) Uniformly randomly partition the n nodes into m groups and then extract the corresponding m subgraphs $\{g_1, \cdots, g_m\}$;
2) Find clusters in each subgraph in parallel by applying $\mathfrak{A}$. Denote the set of the recovered clusters in g_i by $\mathcal{S}_{g_i}$;
3) Build the fused graph $\mathcal{G}$ based on $\{\mathcal{S}_{g_1}, \cdots, \mathcal{S}_{g_m}\}$ by Algorithm 2;
4) Find clusters in $\mathcal{G}$ by solving (1). Denote the set of the recovered clusters by $\mathcal{S}_{\mathcal{G}}$;
5) Merge $\{\mathcal{S}_{g_1}, \cdots, \mathcal{S}_{g_m}\}$ based on $\mathcal{S}_{\mathcal{G}}$.

3. Theoretical Analysis

In this section we establish the conditions under which the proposed framework provably recovers the ground truth. Indeed, even when the clusters in the subgraphs are partially recovered, we can still recover the correct clustering relationship of the entire graph. We first investigate the theoretical guarantee for the weighted clustering shown in Equation (1). Based on this, we then provide the performance guarantees of Algorithm 1. Along the way, we also provide answers to Question 1 and 2 mentioned in the previous section.

Algorithm 2 Build a Fused Graph

Input: Observed adjacency matrix $\mathbf{A}$, the sets of the recovered clusters $\mathcal{S}_{g_1}, \cdots, \mathcal{S}_{g_m}$ and parameter l, t, T.

Procedure:

1) Break up small clusters: For each $g \in \{g_1, \cdots, g_m\}$ and each $\mathcal{U} \in \mathcal{S}_g$, if $|\mathcal{U}| < T$, let $\mathcal{S}_g := (\mathcal{S}_g \setminus \mathcal{U}) \cup \{\{k\} \text{ for each } k \in \mathcal{U}\}$. Otherwise, uniformly randomly partition the nodes in $\mathcal{U}$ into l clusters $\mathcal{U}_1, \cdots, \mathcal{U}_l$ and let $\mathcal{S}_g := (\mathcal{S}_g \setminus \mathcal{U}) \cup \{\mathcal{U}_1, \cdots, \mathcal{U}_l\}$;

2) Create super nodes: Create a corresponding super node V for each cluster $\mathcal{U}$ in $\bigcup_{i=1}^m \mathcal{S}_{g_i}$. This relationship is represented by $V \sim \mathcal{U}$;

3) Build the fused graph $\mathcal{G}$: For super nodes V_i and V_j where $V_i \sim \mathcal{U}_i$ and $V_j \sim \mathcal{U}_j$, edge E_{ij} that connects them is generated as follows: If $|\mathcal{U}_i| = |\mathcal{U}_j| = 1$, let $E_{ij} := A_{uv}$ where $u \in \mathcal{U}_i$ and $v \in \mathcal{U}_j$. Otherwise, we compute

$$\hat{E}(V_i, V_j) := \frac{\sum_{u \in \mathcal{U}_i} \sum_{v \in \mathcal{U}_j} A_{uv}}{\sum_{u \in \mathcal{U}_i} \sum_{v \in \mathcal{U}_j} 1}.$$

Let $E_{ij} := 1$ if $\hat{E}(V_i, V_j) \geq t$ or $E_{ij} := 0$ otherwise;

4) Construct the set of high confidence nodes: $\mathcal{H} \triangleq \{V_i : |\mathcal{U}_i| > 1 \text{ for } V_i \in \mathcal{V}\}$;

5) Return $\mathcal{G}$ and $\mathcal{H}$.

3.1. Graph Clustering with High Confidence Nodes

Let us consider a graph with high confidence nodes: more specifically, the graph is generated according to the GSBM which contains n nodes and r clusters, and its ith cluster contains s_i high confidence nodes. Let $s \triangleq \sum_{i=1}^r s_i$, i.e., the total number of the high confidence nodes, K is the size of the smallest cluster, and K^* be the minimum size of the clusters containing at least one ordinary node. We have the following theorem:

Theorem 1. *Let* $\lambda = \frac{c_0}{\sqrt{\max\{n-s, K^*\} \log n}}$, $c_{\mathcal{A}} = \lambda\sqrt{\frac{1-t}{t}}$, $c_{\mathcal{A}^c} = \lambda\sqrt{\frac{t}{1-t}}$ *and* $c_{\mathcal{C}} = \frac{c_0}{\sqrt{K \log n}}$, *where* t *satisfies* $\frac{1}{4}p + \frac{3}{4}q \leq t \leq \frac{3}{4}p + \frac{1}{4}q$, *then* $\mathbf{Y}^*$ *is the unique optimal solution of* (1) *with high probability if* $K \geq c_K \log n$ *and*

$$\frac{p-q}{\sqrt{p(1-q)}} \geq c_1 \max\left\{\frac{\sqrt{(n-s)\log n}}{K^*}, \sqrt{\frac{\log n}{K^*}}\right\} \quad (2)$$

hold for some universal constants c_K, c_0 *and* c_1. *Furthermore, if there exist no outliers in the graph and* $\lambda = \frac{c_0}{\sqrt{\max\{K^*, \max_i\{\sum_{j \neq i}(K_i - s_i)\}\} \log n}}$, (2) *can be replaced by a milder condition*

$$\frac{p-q}{\sqrt{p(1-q)}} \geq c_1 \max\left\{\frac{\sqrt{\max_i\{\sum_{j \neq i}(K_i - s_i)\} \log n}}{K^*}, \sqrt{\frac{\log n}{K^*}}\right\}.$$

Remark. This theorem makes no assumptions on how the confidence nodes are distributed over different clusters. Consider an extreme case: fix $r = O(1)$ and let the r clusters have equal size $\frac{n}{r}$ and the nodes in the first $\frac{r}{2}$ clusters are all highly confident while the nodes in the remaining $\frac{r}{2}$ clusters are all ordinary. By an information-theoretic argument (Chen et al., 2014b;a), one can show that for any algorithm to correctly recover the clusters with probability at least $\frac{3}{4}$, the inequality

$$\frac{p-q}{\sqrt{p(1-q)}} \geq c\frac{1}{\sqrt{n}}, \quad (3)$$

should hold for some universal constant c. Theorem 1 shows that in this case p, q should satisfy

$$\frac{p-q}{\sqrt{p(1-q)}} \geq c_1\sqrt{\frac{r \log n}{n}},$$

This matches the condition (3) up to a logarithmic factor, and thus cannot be substantially improved unless additional assumptions are made.

3.2. Performance Guarantee of Algorithm 1

Graph clustering algorithm $\mathfrak{A}$ used to find clusters in subgraphs is crucial to our framework. Clearly, if $\mathfrak{A}$ generates many misclassified nodes, it is impossible for Algorithm 1 to correctly recover the true clusters. We now characterize two sets of algorithms used in our framework.

Definition 2 (Workable Algorithm)**.** *For vector* $\boldsymbol{\lambda} \in \mathbb{R}^r$ *and non-empty set* $\mathcal{I} \subseteq [r]$, $\mathfrak{A}$ *is* $(\boldsymbol{\lambda}, \mathcal{I})$-workable *if the followings hold with probability at least* $1 - n^{-2}$*: when* (p, q) *is in a set* $\mathfrak{C}$ *parameterized by* $(n, K_1, \cdots, K_r, \boldsymbol{\lambda}, \mathcal{I})$, $\mathfrak{A}$ *is able to recover a set of clusters* $\mathcal{R}_{\mathcal{C}}$ *satisfying 1)* $\forall i \in \mathcal{I}$, $\exists \mathcal{C}_i \in \mathcal{R}_{\mathcal{C}}$ *so that* $\mathcal{C}_i$ *is a subset of the* i*th cluster and* $|\mathcal{C}_i| \geq \lambda_i K_i$, *and 2)* $|\mathcal{C}| \leq \min_{i \in \mathcal{I}} \rho\lambda_i K_i$ *for any* $\mathcal{C} \in \mathcal{R}_{\mathcal{C}} \setminus \bigcup_{i \in \mathcal{I}} \mathcal{C}_i$, *for some constant* $\rho < 1$.

Example 1. *Most of the trace-norm based graph clustering algorithms (e.g., Oymak & Hassibi, 2011; Ames & Vavasis, 2011; Chen et al., 2014b; Jalali et al., 2011) are workable. For example, the algorithm proposed by Chen et al. (2014b) has a performance guarantee that when* $\frac{p-q}{\sqrt{p(1-q)}} \geq c \max\{\frac{\sqrt{n}}{K}, \sqrt{\frac{\log^2 n}{K}}\}$ *for universal constant* c, *its output is the same as the true adjacency matrix* $\mathbf{Y}^*$ *with probability at least* $1 - n^{-8}$, *which means that it is* $(\mathbf{1}, [r])$-*workable. Ailon et al. (2013) provided a refined analysis of this algorithm showing that small clusters do not hinder recovery of sufficiently large ones under some mild conditions, i.e., it is* $(\boldsymbol{\lambda}, [r])$-*workable where* $\lambda_i = 1$ *if the* i*th cluster is relatively large or 0 otherwise.*

Definition 3 (Pseudo-workable Algorithm)**.** *For vector* $\boldsymbol{\lambda}, \boldsymbol{\epsilon} \in \mathbb{R}^r$ *and non-empty set* $\mathcal{I} \subseteq [r]$, $\mathfrak{A}$ *is* $(\boldsymbol{\lambda}, \mathcal{I}, \boldsymbol{\epsilon})$-pseudo-workable *if the followings hold with probability at*

least $1-n^{-2}$: when (p,q) is in a set $\mathfrak{C}$ parameterized by $(n, K_1, \cdots, K_r, \boldsymbol{\lambda}, \mathcal{I}, \boldsymbol{\epsilon})$, $\mathfrak{A}$ is able to recover a set of clusters $\mathcal{R}_{\mathcal{C}}$ satisfying 1) $\forall i \in \mathcal{I}$, $\exists \mathcal{C}_i \in \mathcal{R}_{\mathcal{C}}$ so that $\mathcal{C}_i$ contains at least $\lambda_i K_i$ nodes in the ith cluster and at most $\epsilon_i K_i$ nodes not in the ith cluster, and 2) $|\mathcal{C}| \leq \min_{i\in\mathcal{I}} \rho\lambda_i K_i$ for any $\mathcal{C} \in \mathcal{R}_{\mathcal{C}} \setminus \bigcup_{i\in\mathcal{I}} \mathcal{C}_i$, for some constant $\rho < 1$.

Example 2. *Rohe et al. (2011) analyzed the performance of spectral clustering under the standard stochastic block-model and provided an upper bound of the number of misclassified nodes. For clarity, we here present a specialized version of their theorem as shown in Proposition 1 where the clusters are equal-sized, from which one can easily observe that spectral clustering is indeed $(\mathbf{1}-\boldsymbol{\epsilon}, [r], \boldsymbol{\epsilon})$-pseudo-workable where $\boldsymbol{\epsilon} = \frac{cr^4 \log^2 n}{n}\mathbf{1}$.*

Proposition 1. *Suppose that the graph has n nodes with r equal-sized clusters and satisfies that the probability of a connection between two nodes in the same cluster is p and the probability of a connection between two nodes in different clusters is q. If p and q do not vary as n and $r = O(n^{\frac{1}{4}} \log^{-1} n)$, then the size of the set of the misclassified nodes $\mathcal{M}$ after running spectral clustering on this graph is upper bounded by $|\mathcal{M}| \leq cr^3 \log^2 n$ for some constant c.*

We now present our main theorems, which establish the performance guarantee of Algorithm 1 when $\mathfrak{A}$ used to find clusters in subgraphs is either workable or pseudo-workable, respectively.

Theorem 2. *If $\mathfrak{A}$ is $(\boldsymbol{\lambda}, \mathcal{I})$-workable, $t \in (\frac{1}{4}p + \frac{3}{4}q, \frac{3}{4}p + \frac{1}{4}q)$, and $\frac{T}{\min_{i\in\mathcal{I}} \lambda_i K_i} \in (\frac{3\rho+\rho^2}{2(1+\rho)m}, \frac{1+3\rho}{2(1+\rho)m})$, then when (p,q) belongs to the set $\mathfrak{C}(n/m, K_1/m, \cdots, K_r/m, \boldsymbol{\lambda}, \mathcal{I})$, $c_3 \log n \leq m \leq \frac{1-\rho}{4(1+\rho)}\sqrt{\frac{K}{\log n}}$, and*

$$p - q \geq \min_{l\geq l\geq 1} \max\left\{ c_2\sqrt{\frac{(1+\rho)ml\log\frac{n}{K}}{(1+3\rho)\min_{i\in\mathcal{I}}\lambda_i K_i}}, c_1\sqrt{p(1-q)}\max\left\{\frac{\sqrt{(n-\sum_{i\in\mathcal{I}}\lambda_i K_i)\log n}}{S(m,l)}, \sqrt{\frac{\log n}{S(m,l)}}\right\}\right\},$$

with probability at least $1-\max\{(mr)^{-10}, n^{-1}\}$ the true clusters can be recovered by Algorithm 1, where $S(m,l) = \min\{\min_{i\in\mathcal{I}:\lambda_i\neq 1}\{ml + (1-\lambda_i)K_i\}, \min_{i\in\mathcal{I}^c} K_i\}$, $\bar{l} = \max\left\{\frac{1}{4}\sqrt{\frac{(1+3\rho)\min_{i\in\mathcal{I}}\lambda_i K_i}{2(1+\rho)m\log n}}, 1\right\}$, and c_1, c_2, c_3 are universal constants.

Remark. Our framework can guarantee exact recovery even when each cluster in the subgraphs is partially recovered. This happens especially when trace-norm based clustering algorithms are applied on a sparse graph where $p - q < 0.2$ as shown in the experiments in (Chen et al., 2014b) or spectral clustering runs with a relatively large cluster number. Parameter T and m are well defined when $\rho < 1$ and $K = \Omega(\log^3 n)$. Recall that m controls the number of the separated subgraphs. From the bound related to (p,q), we observe that increasing m leads to a smaller $\mathfrak{C}$ which increases the difficulty of recovering the true clusters in subgraphs, but a larger $S(m,l)$ which means that the difficulty of recovering clusters in the fused graph is reduced.

Theorem 3. *Suppose that $\mathfrak{A}$ is $(\boldsymbol{\lambda}, \mathcal{I}, \boldsymbol{\epsilon})$-pseudo-workable, $t \in (\frac{1}{4}p + \frac{3}{4}q + \varphi, (\frac{3}{4}p + \frac{1}{4}q)(1-\varphi))$, $\frac{T}{\min_{i\in\mathcal{I}}\lambda_i K_i} \in (\frac{3\rho+\rho^2}{2(1+\rho)m}, \frac{1+3\rho}{2(1+\rho)m})$, then when (p,q) belongs to the set $\mathfrak{C}(n/m, K_1/m, \cdots, K_r/m, \boldsymbol{\lambda}, \mathcal{I}, \boldsymbol{\epsilon})$, $c_3 \log n \leq m \leq \frac{1-\rho}{4(1+\rho)}\sqrt{\frac{K}{\log n}}$, and*

$$p - q \geq \max\left\{ c_2\sqrt{\frac{(1+\rho)ml\log\frac{n}{K}}{(1+3\rho)\min_{i\in\mathcal{I}}\lambda_i K_i}}, 72\max_{i\in\mathcal{I}}\frac{\epsilon_i}{\lambda_i}, c_1\sqrt{p(1-q)}\max\left\{\frac{\sqrt{(n-\sum_{i\in\mathcal{I}}\lambda_i K_i)\log n}}{S(m,l)}, \sqrt{\frac{\log n}{S(m,l)}}\right\}\right\},$$

with probability at least $1-\max\{(mr)^{-10}, n^{-1}\}$ the clusters recovered by Algorithm 1 contain at most $\sum_{i\in\mathcal{I}}\epsilon_i K_i$ misclassified nodes, where

$$l \leq \min\left\{\max\left\{\frac{1}{4}\sqrt{\frac{(1+3\rho)\min_{i\in\mathcal{I}}\lambda_i K_i}{2(1+\rho)m\log n}}, 1\right\}, \frac{p-q}{72}\min_{i\in\mathcal{I}}\frac{\lambda_i}{\epsilon_i}\right\},$$

$$S(m,l) = \min\left\{\min_{i\in\mathcal{I}:\lambda_i\neq 1}\left\{ml + [(1-\lambda_i)K_i - \sum_{j\in\mathcal{I}, j\neq i}\epsilon_j K_j]_+\right\}, \max\left\{\min_{i\in\mathcal{I}^c} K_i - \sum_{j\in\mathcal{I}}\epsilon_j K_j, 1\right\}\right\},$$

$\varphi = 18\max_{i\in\mathcal{I}}\frac{\epsilon_i l}{\lambda_i}$, *and c_1, c_2, c_3 are universal constants.*

Remark. Theorem 3 states the "error-tolerant" property of Algorithm 1. In particular, even when $\mathfrak{A}$ does not guarantee exact recovery, the total number of the misclassified nodes in the clusters recovered by Algorithm 1 is still small as long as the recovered clusters in subgraphs do not contain too many misclassified nodes.

We now specialize our main theorem. For simplicity, we set $l = 1$ in Algorithm 1. The following corollaries consider the case where $\mathfrak{A}$ is the trace-norm based clustering algorithm proposed by Chen et al. (2014b), i.e., $\mathfrak{A}$ solves the optimization problem shown in (1) with no high confidence nodes, i.e., $\mathcal{C} = \emptyset$. We call this algorithm CSX following author acronym in the sequel.

Corollary 1. *If $\mathfrak{A}$ is CSX, $t \in (\frac{1}{4}p + \frac{3}{4}q, \frac{3}{4}p + \frac{1}{4}q)$, $0 \leq T \leq \frac{\min_i K_i}{2m}$, $c_3 \log n \leq m \leq \frac{1}{4}\sqrt{\frac{K}{\log n}}$, and*

$$p - q \geq \max\left\{ c_1\frac{\sqrt{p(1-q)mn\log n}}{K}, c_2\sqrt{\frac{m\log n}{K}}\right\} \tag{4}$$

where c_1, c_2, c_3 are universal constants, then Algorithm 1 recovers the true clusters with probability at least $1-\max\{(mr)^{-10}, n^{-1}\}$.

Remark. "Speedup is not a free lunch": Increasing m reduces the computational time but requires stronger conditions to guarantee exact recovery, i.e., the lower bound of $p-q$ is $\sqrt{m}$ times greater than the one shown in (Chen et al., 2014b). In practice $p-q$ can be smaller than (4) while exact recovery is still achieved, since the exact recovery of clusters *in subgraphs* is not necessary as stated in Theorem 1. More discussion can be found in Section 4.

Our framework also offers benefits to dealing with small clusters. Most of graph clustering algorithms (Giesen & Mitsche, 2005; Shamir & Tsur, 2007; Chen et al., 2014b; Oymak & Hassibi, 2011; Ames, 2014; Chaudhuri et al., 2012) require $K=\Omega(\sqrt{n})$, but our algorithm requires a much weaker condition where K can be $o(\sqrt{n})$.

Corollary 2. *If $\mathfrak{A}$ is CSX, then there exist universal constants c_0, c_1, c_2, c_3, c_4 such that the following holds with probability at least $1-\max\{(mr)^{-10}, n^{-1}\}$: Let*

$$u = c_1\frac{\sqrt{p(1-q)mn}}{p-q}\log^2 n, \text{ and } l = c_2\frac{\sqrt{p(1-q)mn}}{p-q}.$$

if for all $i\in[r]$, either $K_i\geq u$ or $K_i\leq l$ and if $t\in(\frac{1}{4}p+\frac{3}{4}q, \frac{3}{4}p+\frac{1}{4}q)$, $0\leq T\leq \frac{\min_{K_i\geq u}K_i}{2m}$,

$$c_0\log n \leq m \leq \min\left\{\frac{1}{4}\sqrt{\frac{K}{\log n}}, c_4\frac{(p-q)^2\min_{K_i\geq u}K_i}{\log n}\right\}$$

and

$$K\geq \max\left\{c_3\frac{\sqrt{p(1-q)\sum_{K_i\leq l}K_i\log n}}{p-q}, c_3^2\frac{p(1-q)\log n}{(p-q)^2}\right\},$$

Algorithm 1 recovers the true clusters.

Remark. We may observe that if the graph only has $O(1)$ small clusters whose sizes have the same order of magnitude, then there exists a constant c so that $K=c\log^3 n$ satisfies the conditions above. Note that the iterative algorithm proposed by Ailon et al. (2013) can recover almost all clusters via a so-called peeling strategy. But it has computational complexity $O(n^3)$ which becomes unaffordable when the number of nodes is large, say more than 5000. In contrast, as we show in Section 3.3, the computational complexity of our algorithm is much lower.

We now address clustering partially observed graph. The graph is first generated by the GSBM, and then A_{ij} is set to ? (unknown) with probability $1-\mu$ for each (i,j) independently. Clearly, if we construct a new graph by setting $A_{ij}=0$ when $A_{ij}=?$, we can convert it into the fully observed case. Based on Corollary 1, we have

Corollary 3. *If $\mathfrak{A}$ is CSX, $t\in(\frac{1}{4}\mu p+\frac{3}{4}\mu q, \frac{3}{4}\mu p+\frac{1}{4}\mu q)$, $0\leq T\leq\frac{\min_i K_i}{2m}$, $c_3\log n\leq m\leq\frac{1}{4}\sqrt{\frac{K}{\log n}}$ and*

$$\sqrt{\mu}(p-q)\geq\max\left\{c_1\frac{\sqrt{pmn\log n}}{K}, c_2\sqrt{\frac{m\log n}{K}}\right\},$$

where c_1, c_2, c_3 are universal constants, then Algorithm 1 recovers the true clusters with probability at least $1-(mr)^{-10}$.

3.3. Complexity

The computational cost of our graph clustering framework can be much less than the trace-norm based clustering, e.g., (Oymak & Hassibi, 2011; Ames & Vavasis, 2011; Chen et al., 2014b; Jalali et al., 2011) and spectral clustering (Luxburg, 2007).

Theorem 4. *If $\mathfrak{A}$ is $(\boldsymbol{\lambda},\mathcal{I})$-workable or $(\boldsymbol{\lambda},\mathcal{I},\boldsymbol{\epsilon})$-pseudo-workable, and has computational complexity $O(f(n))$ and memory complexity $O(g(n))$, then the computational and memory complexity of Algorithm 1 are $O\left(f(\frac{n}{m})m+(mrl+n-\sum_{i\in\mathcal{I}}\lambda_i K_i)^3\right)$, and $O\left(g(\frac{n}{m})m+(mrl+n-\sum_{i\in\mathcal{I}}\lambda_i K_i)^2\right)$, respectively.*

When we use CSX for $\mathfrak{A}$, and Algorithm 1 runs on a machine with B cores, we obtain the following corollary.

Corollary 4. *If $\mathfrak{A}$ is CSX, then when the conditions in Corollary 1 are satisfied, the computational and memory complexity for Algorithm 1 running in parallel on a machine with B cores are $O\left(\frac{n^3}{Bm^2}+(mrl)^3\right)$ and $O\left(\frac{Bn^2}{m}+(mrl)^2\right)$, respectively.*

Clearly, when $r,l\ll n$, our algorithm is approximately Bm^2 times faster than applying $\mathfrak{A}$ on the entire graph.

4. Experiments

We now investigate the performance of Algorithm 1 on a variety of simulated and real-world datasets, and compare it with other algorithms, e.g., spectral clustering (Luxburg, 2007), Metis (Karypis & Kumar, 1998b), CSX (Chen et al., 2014b), and ESCG that is recently proposed by Liu et al. (2013) and is designed for large graph clustering based on spectral clustering. Except for Metis implemented by Karypis & Kumar (1998b), we implement the other algorithms in Python. The experiments are conducted on a desktop PC with an i7 3.4GHz CPU and 4GB memory.

4.1. Simulations on Synthetic Data

The simulated graphs are generated based on the standard stochastic blockmodel with n nodes, r clusters, and probabilities p and q. We select CSX – the trace-norm based algorithm proposed by Chen et al. (2014b) – as $\mathfrak{A}$. Given the output refined adjacency matrices generated by $\mathfrak{A}$ or the clustering algorithm shown in (1), the clusters are constructed by simply finding the connected components. In the simulations, we repeat each test 10 times and use the averaged *ratio of non-exact recovery* (RNER) and the wall-clock time to measure the performance of each algorithm.

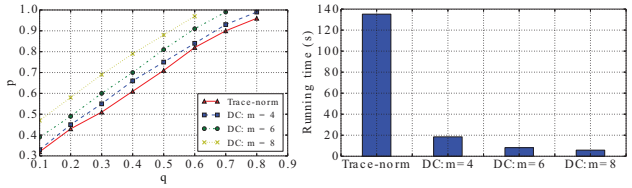

Figure 2. Comparison between DC and Trace-norm when $n = 2000$ and $m = 4, 6, 8$. (Left) Each curve presents the pairs of p and q for which a method succeeds. (Right) The running time.

RNER is the fraction of attempts where the output clustering is not exactly the same as the the ground truth.

We set $l = 1$, $t = (p + q)/2$ and $T = 10$ in the following experiments. When p and q are unknown, we use Algorithm 2 in Chen et al. (2014b) to estimate t. One can also apply the cross validation to determine t such that the in-cluster edge density of the recovered clusters is maximized. Parameter λ and $c_{\mathcal{C}}$ in (1) are set to 1.6 and 5, respectively. For simplicity, we use DC, Trace-norm and Spectral to denote the proposed Algorithm 1, CSX, and spectral clustering, respectively. For small graphs with around 1000 to 2000 nodes, Trace-norm has been extensively compared with Spectral, SLINK (Sibson, 1973), etc., (Chen et al., 2014b), and typically outperforms the others. Hence in the first two experiments, we mainly compare the performance of DC and Trace-norm.

The first experiment compares the performance of DC and Trace-norm when m varies. Corollary 1 is established based on analyzing the conditions when the exact recovery of clustering for each subgraph is achieved, which leads to a lower bound of $p - q$ that is $\sqrt{m}$ times larger than that of Trace-norm. This lower bound is conservative since Theorem 1 shows that the it is possible to achieve exact recovery when $\mathfrak{A}$ correctly classifies at least $\lambda_i K_i$ nodes in the ith cluster, which means that empirically, DC can do better than predicted by Corollary 1. Figure 2 shows the comparison results. It can be observed that $p - q$ is approximately 0.2 for Trace-norm, and 0.25 for DC when $m = 4$. Clearly, 0.25 for DC is less than 0.4 predicted by Corollary 1. Figure 2 also shows their running time. We observe that DC is almost 7 times faster than Trace-norm when $m = 4$. Therefore, DC is able to recover clusters much faster that Trace-norm without much loss of performance.

The second experiment compares the performance of DC and Trace-norm when small clusters exist. Figures 3(a) and (b) plot the ratio of exact recovery and running time of each algorithm against the number of clusters and the size of the smallest cluster, respectively. As shown in Corollary 2, DC requires a weaker condition on K than Trace-norm, which implies that DC will perform better than Trace-norm in the face of small clusters. The simulation results empirically

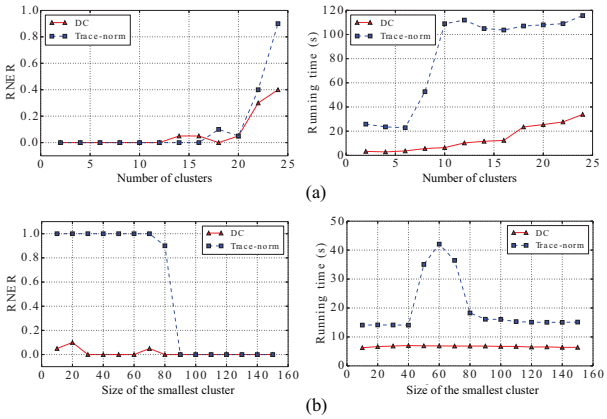

Figure 3. The comparison between DC and Trace-norm ($n = 1000, p = 0.8, q = 0.2$). (a) Nearly equal-sized clusters: the sizes of the first $r - 1$ clusters are $[\frac{n}{r}]$ while the size of the rth cluster is $n - (r - 1)[\frac{n}{r}]$. (b) Small clusters: there exist three clusters whose sizes are $500, 500 - K$ and K, respectively.

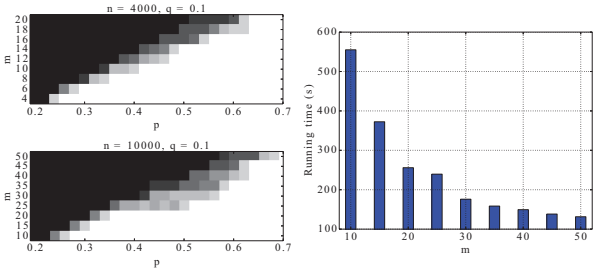

Figure 4. (Left) The fraction of successes for different m and p. (Right) The average running time.

validate this conclusion, from which we observe that Trace-norm fails when K is less than 50 but DC still succeeds. Moreover, the running time of DC is again much shorter than that of Trace-norm. Interestingly, we observe form Figure 3(b) that the running time of Trace-norm has a sharp rise when K is about 40, because it appears more iterations needed to converge.

In the third experiment, we investigate the performance of DC for different values of p, q and m. We repeat each test 10 times. Figure 4 displays the fraction of successes when p, m vary and the corresponding averaged running time for different m when $n = 10000$. Obviously, when the gap between p and q becomes larger, one can increase m to achieve more speedup without affecting the clustering performance. This agrees with our theoretic result shown in Corollary 1 and 4. Figure 5 shows the fraction of successes when p, q vary. We observe that in this case when $p - q \geq 0.15$, DC works pretty well. In real applications, when the gap between p and q is large enough, one can try to apply DC with proper m instead of performing Trace-norm or Spectral on the entire graph.

Table 1. Running time (second unit) of different clustering algorithms ("-" means "out of memory").

n	500	1000	2000	3000	4000	6000	10000	20000	50000	100000
DC	1.550	2.501	5.120	8.392	13.036	23.108	53.134	607.690	2355.972	10281.718
Trace-norm	3.419	21.072	177.752	3031.910	-	-	-	-	-	-
Spectral	0.128	0.736	5.543	18.142	41.585	-	-	-	-	-
ESCG	9.176	12.391	24.747	46.046	74.940	165.108	376.787	1041.690	6516.708	26658.519

Table 2. Running time (second unit) of different plant clique algorithms ("-" means "out of memory").

n	500	600	700	800	1000	2000	3000
DC	1.655	1.830	2.367	2.670	2.445	5.099	8.450
(Ames & Vavasis, 2014)	116.2672	244.4297	490.5083	819.0086	-	-	-
(Ames, 2014)	1.8915	49.5097	113.5927	150.8029	550.4	4485.0	15197.5

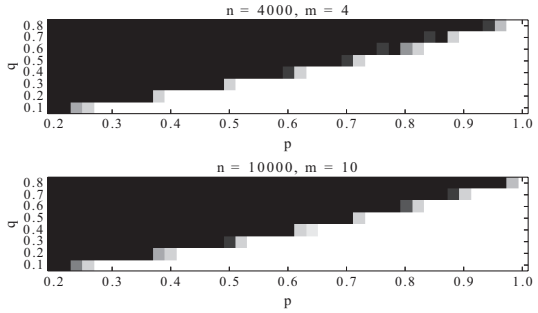

Figure 5. The fraction of successes for different p and q.

In the fourth experiment, we compare the running time of DC with other methods. Table 1 shows the running times of four algorithms, i.e., DC, Trace-norm, Spectral and ESCG. The simulated graphs are generated with different n but fixed p, q, r ($p = 0.8, q = 0.2, r = 5$). All these methods can recover the true clusters in this setting. Obviously, DC has the least running time when $n \geq 3000$. Indeed, when n is relatively large,Trace-norm and Spectral break down due to high memory and computational demand. Since the plant clique model is a special case of the GSBM, we also compare DC with two recently proposed plant clique algorithms as shown in Table 2. The graphs are generated with $p = 1.0, q = 0.2$ and $r = 5$. The computational advantage of the proposed DC algorithm is obvious.

4.2. Real-world Datasets

We now evaluate DC on three real-world datasets, namely, MNIST (LeCun et al., 1995), Arxiv[1] and DBLP[2]. For MNIST, 10,000 samples were randomly selected from the digit images with label 0, 1, or 7, and the graph was generated by connecting each sample to its 1000-nearest neighbors where the distance measure is the Euclidean distance. For Arxiv, we used the texts from the years 1999-2003 to build the graph whose nodes and edges represent texts and citations, respectively. For DBLP, we randomly selected a subset from it and constructed a co-author-graph where each node corresponds to an author and each edge corresponds to a co-authorship between two authors.

[1]http://www.cs.cornell.edu/projects/kddcup/datasets.html
[2]http://dblp.uni-trier.de/

Table 3. The performance of different algorithms on the real world datasets MNIST, Arxiv and DBLP, which is measured by computing the in-cluster and cross-cluster edge densities.

MNIST (10000 nodes, 13005198 edges)				
Algorithm	DC	Metis	Spectral	ESCG
In-cluster (10^{-2})	35.00	33.94	34.94	35.05
Cross-cluster (10^{-2})	1.77	2.53	1.81	1.74
Arxiv (12480 nodes, 252906 edges)				
Algorithm	DC	Metis	Spectral	ESCG
In-cluster (10^{-4})	114.46	106.22	22.39	16.55
Cross-cluster (10^{-4})	8.40	9.73	15.61	0.22
DBLP (24599 nodes, 376711 edges)				
Algorithm	DC	Metis	Spectral	ESCG
In-cluster (10^{-4})	14.53	14.42	7.55	3.53
Cross-cluster (10^{-4})	5.63	5.64	6.07	44.30

The setting of Algorithm 1 is different from that in the simulations. We choose Spectral or Metis as $\mathcal{A}$ depending on whose result has a higher in-cluster edge density. Given the output of (1) performed on the fuse graph, we apply Metis to classify the nodes into a specific number of clusters. Due to the memory limitation, for two of the baseline algorithms, namely, Spectral and Metis, we apply a similar algorithm as Algorithm 1 but replacing (1) with Spectral and Metis respectively. For fair comparison, all the algorithms are forced to partition the nodes of MNIST into 3 clusters and partition the nodes of Arxiv and DBLP into 15 clusters. The performance of each algorithm is measured by computing the in-cluster and cross-cluster edge densities (Table 3). When the graph is dense (e.g., MNIST), DC and ESCG have similar performance. But when the graph is sparse (e.g., DBLP), Spectral and ESCG tend to classify most of the nodes into one big cluster, so that the corresponding in-cluster densities are low. In general, our algorithm DC outperforms all three baselines on these datasets.

Acknowledgments

This work is partially supported by the Ministry of Education of Singapore AcRF Tier Two grants R265000443112 and R265000519112, and A*STAR Public Sector Funding R265000540305.

References

Ailon, N., Chen, Y., and Xu, H. Breaking the small cluster barrier of graph clustering. In *ICML*, 2013.

Ames, Brendan P. W. Guaranteed clustering and biclustering via semidefinite programming. *Mathematical Programming*, 143(1-2):429–465, 2014.

Ames, Brendan P. W. and Vavasis, Stephen A. Nuclear norm minimization for the planted clique and biclique problems. *Mathematical Programming*, 129(1):69–89, September 2011.

Ames, Brendan P. W. and Vavasis, Stephen A. Convex optimization for the planted k-disjoint-clique problem. *Mathematical Programming*, 143(1-2):299–337, 2014.

Chaudhuri, K., Graham, F. C., and Tsiatas, A. Spectral clustering of graphs with general degrees in the extended planted partition model. In *COLT*, 2012.

Chen, W., Song, Y., Bai, H., Lin, C., and Chang, E. Y. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(3):568–586, 2011.

Chen, Y., Lim, S. H., and Xu, H. Weighted graph clustering with non-uniform uncertainties. In *ICML*, 2014a.

Chen, Y., Sanghavi, S., and Xu, H. Improved graph clustering. *IEEE Transactions on Information Theory*, 60(10): 6440–6455, 2014b.

Condon, A. and Karp, R. Algorithms for graph partitioning on the planted partition model. *Random Structures Algorithms*, 18(2):116–140, 2001.

Drineas, P. and Mahoney, M. W. On the Nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

Fowlkes, C., Belongie, S., Chung, F., and Malik, J. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(2):214–225, 2004.

Giesen, J. and Mitsche, D. Reconstructing many partitions using spectral techniques. *Fundamentals of Computation Theory*, pp. 433–444, 2005.

Goldenberg, A., Zheng, A. X., Fienberg, S. E., and Airoldi, E. M. A survey of statistical network models. *Foundations and Trends in Machine Learning*, 2(2):129–233, 2010.

Günnemann, S., Färber, I., Boden, B., and Seidl, T. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *ICDM*, 2010.

Holland, P., Laskey, K., and Leinhardt, S. Stochastic blockmodels: Some first steps. *Social Networks*, 5:109–137, 1983.

Jalali, A., Chen, Y., Sanghavi, S., and Xu, H. Clustering partially observed graphs via convex optimization. In *ICML*, 2011.

Karypis, G. and Kumar, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998a.

Karypis, G. and Kumar, V. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48:96–129, 1998b.

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Müller, U., Säckinger, E., Simard, P., and Vapnik, V. Comparison of learning algorithms for handwritten digit recognition. In *International Conference on Artificial Neural Networks*, pp. 53–60, 1995.

Liu, J., Wang, C., Danilevsky, M., and Han, J. Large-scale spectral clustering on graphs. In *IJCAI*, 2013.

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

Macropol, K. and Singh, A. Scalable discovery of best clusters on large graphs. *Proceedings of the VLDB Endowment*, 3(1-2):693–702, 2010.

Oymak, S. and Hassibi, B. Finding dense clusters via "low rank + sparse" decomposition. *Arxiv preprint*, 2011.

Rohe, Karl, Chatterjee, Sourav, and Yu, Bin. Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals of Statistics*, 39(4):1878–1915, 08 2011.

Shamir, R. and Tsur, D. Improved algorithms for the random cluster graph model. *Random Structures Algorithms*, 31(4):418–449, 2007.

Sibson, R. Slink: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16 (1):30–34, 1973.

Watts, D. J. and Strogatz, S. H. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):409–10, 1998.

Whang, J. J., Sui, X., and Dhillon, I. S. Scalable and memory-efficient clustering of large-scale social networks. In *ICDM*, 2012.

Yan, D., Huang, L., and Jordan, M. I. Fast approximate spectral clustering. In *KDD*, pp. 907–916, 2009.