Spectral Clustering by Recursive Partitioning

Conference Paper · September 2006

DOI: 10.1007/11841036_25 · Source: DBLP

4 authors, including:

Anirban Dasgupta
Indian Institute of Technology Gandhinagar
76 PUBLICATIONS **6,663** CITATIONS
SEE PROFILE

Ravindran Kannan
Indian Institute of Science
177 PUBLICATIONS **13,464** CITATIONS
SEE PROFILE

Pradipta Mitra
Google Inc.
29 PUBLICATIONS **484** CITATIONS
SEE PROFILE

Yale University
Department of Computer Science

Spectral Clustering by Recursive Partitioning

Anirban Dasgupta
Yahoo! Research Labs

John Hopcroft
Cornell University

Ravi Kannan
Yale University

Pradipta Mitra
Yale University

YALEU/DCS/TR-1354
April 17, 2006
Updated July 5, 2006

Pradipta Mitra is supported by NSF's ITR program under grant number 0331548. Ravi Kannan is supported by NSF Award CCF 0310805

Abstract

In this paper, we analyze the second eigenvector technique of spectral partitioning on the planted partition random graph model, by constructing a recursive algorithm using the second eigenvectors in order to learn the planted partitions. The correctness of our algorithm is not based on the ratio-cut interpretation of the second eigenvector, but exploits instead the stability of the eigenvector subspace. As a result, we get an improved cluster separation bound in terms of dependence on the maximum variance. We also extend our results for a clustering problem in the case of sparse graphs.

1 Introduction

Clustering of graphs is an extremely general framework that captures a number of important problems on graphs e.g. coloring, bisection, and finding dense communities. In a general setting, the clustering problem is to partition the vertex set of a graph into "clusters", where each cluster contains vertices of only "one type". The exact notion of what the vertex "type" represents is dependent on the particular application of the clustering framework and may be defined in many ways. We will deal with the clustering problem on graphs generated by the versatile planted partition model (See [18, 5]). In this probabilistic model, the vertex set of the graph is partitioned into k subsets $T_1, T_2, \ldots, T_k$. Each edge (u, v) is then a random variable that is independently chosen to be present with a probability A_{uv}, and absent otherwise. The probabilities A_{uv} depend only on the parts to which the two endpoints u and v belong. The adjacency matrix $\widehat{A}$ of the random graph so generated is presented as input. Our task then is to identify the latent clusters $T_1, T_2, \ldots, T_k$ from $\widehat{A}$.

Spectral methods have been widely used for clustering problems, both for theoretical analysis involving generative models and empirical and application areas. The underlying idea in spectral methods is to use information about the eigenvectors of $\hat{A}$ in order to extract structure. There are different variations to this basic theme of spectral clustering, which can be essentially divided into 2 classes of algorithms.

1. Projection heuristics, in which the top few eigenvectors of the adjacency matrix $\widehat{A}$ are used to construct a low-dimensional representation of the data, which is then clustered.

2. The second eigenvector heuristic, in which the coordinates of the second eigenvector of $\widehat{A}$ is used to find a split of the vertex set into two parts. This technique is then applied recursively to each of the parts obtained.

Experimental results claiming the goodness of both the spectral heuristics abound. Relatively fewer are results that strive to demonstrate provable guarantees about the heuristics. Perhaps more importantly, the worst case guarantees [17] that have been obtained do not seem to match the stellar performance of spectral methods on most inputs, and thus it is still an open question to characterize the class of inputs for which spectral heuristics do work well. An equally pressing open question is to figure out which particular spectral heuristic is more appropriate for any one setting. Is it possible to characterize instances for which any one of the spectral methods will outperform the other, or do the two methods give competitive results for most interesting problem instances?

In order to be able to formalize the average case behavior of spectral analysis, researchers have analyzed its performance on graphs generated by random models with latent structure [4, 18]. As described above, these graphs are generated by zero-one entries from a independently chosen according to a low-rank probability matrix. The low rank of the probability matrix reflects the small number of vertex types present in the unperturbed data. The intuition developed by Azar et al. [4] is that in such models, the random perturbations may cause the individual eigenvectors to vary significantly, but the subspace spanned by the top few eigenvectors remains stable. Thus, the projection heuristic can be formalized as trying to approximate the idealized subspace that contains all the required cluster information encoded in the probability matrix. From this perspective, however, the second eigenvector technique does not seem to be well motivated, and it remains an open question as to whether we can claim anything better than the worst case bounds for the second eigenvector heuristic in this setting.

In this paper, we prove the goodness of the second eigenvector partitioning for the planted partition random graph model [11, 4, 18, 10]. We demonstrate that in spite of the fact that the second eigenvector itself is not stable, we can use it to recover the embedded structure. The key argument behind our approach is to exploit the fact that the stability of the eigenvector subspace implies that the number of possible configurations of the second eigenvector is small. We first create identical copies of our dataset by randomly partitioning our input matrix into $O(\log n)$ submatrices. The second eigenvector partitioning is then applied on each of these submatrices, creating different approximately good bi-partitions. We then finally combine all these bi-partitions to create a clean bi-partition. This procedure is then performed recursively on each of the resulting parts.

Our main aim in analyzing the planted partition model using the second eigenvector technique is to try to bridge the gap between the worst case analysis and the actual performance. However, in doing so, we achieve a number of other goals too. The most significant among these is that we can get tighter guarantees than McSherry [18] in terms of the dependence on the maximum variance. The required separation between the columns clusters T_r and T_s can now be in terms of $\sigma_r + \sigma_s$, the maximum variances in each of these two clusters, instead of the maximum variance $\sigma_{\max}$ in the entire matrix. This gain could be significant if the maximum variance $\sigma_{\max}$ is due to only one cluster, for instance one community being more densely connected than everyone else, and thus can potentially lead to identification of a finer structure in the data. Our separation bounds are however worse than [18, 1] in terms of dependence on k, the number of clusters.

The algorithm that we develop is also quite natural in terms of applicability. As a counterpoint, the analysis of the spectral projection heuristic in McSherry [18] and in Dasgupta et al.[18, 10] for the planted partition graph model requires the construction of an intricate projection matrix. Unlike previous work that tries to capture the goodness of the second eigenvector partitioning in terms of the stability of the second eigenvector, we base our proof on perturbation analysis of the random matrix $\widehat{A} - \mathbf{E}\left[\widehat{A}\right]$.

At each stage in our algorithm we divide up the vertices into two (or more) parts each of which respects the original structure. In effect, we never need to know the actual number of clusters that are present. We do, however, need an upper bound on the number of clusters in estimating the number of submatrices required for cleaning up the bi-partitions. In terms of computational efficiency, we have to pay a small penalty as at each stage we are computing the second eigenvectors of $O(\log n)$ submatrices instead of a single large matrix.

Another contribution of the paper is to model and solve a restricted clustering problem for sparse (constant degree) graphs. Graphs clustered in practice are often "sparse", even of very low constant degree. A concern about analysis of many heuristics on random models [18, 10] is that they don't cover sparse graphs.

In this paper, we use a random regular model motivated by random regular graphs (see [14, 6], for example) for the clustering problem that allows us to use strong concentration results which are available in that setting. We will use some extra assumptions on the degrees of the vertices and finally show that expansion properties of the model will allow us to achieve a clean clustering through a simple algorithm.

2 Model

We have n vertices, that we denote by 1 to n. A is matrix of probabilities where the entry A_{uv} is the probability of an edge between the vertices u and v. The vertices are partitioned into k

different clusters $T_1, T_2, \ldots, T_k$. The size of the r^{th} cluster T_r is denoted by n_r and the minimum size is denoted by $n_{\min} = \min_r\{n_r\}$. Let, $w_{\min} = n_{\min}/n$. We also assume that the minimum size $n_{\min} \in \Omega(n/k)$ i.e. $w_{\min} = \Omega(1/k)$. The characteristic vector of the cluster T_r is denoted by $\mathbf{g}^{(r)}$ defined as $\mathbf{g}^{(r)}(i) = 1/\sqrt{n_r}$ for $i \in T_r$ and 0 elsewhere. The probability matrix A is block structured in the following sense: the probability A_{uv} depends only on the two clusters in which the vertices u and v belong to. Given the probability matrix A, the random graph $\widehat{A}$ is then generated by independently setting each $\widehat{A}_{uv} (= \widehat{A}_{vu})$ to 1 with probability A_{uv} and 0 otherwise. Thus, the expectation of the random variable $\widehat{A}_{uv}$ is equal to A_{uv}. The variance of $\widehat{A}_{uv}$ is thus $A_{uv}(1 - A_{uv})$. The maximum variance of any entry of $\widehat{A}$ is denoted σ^2, and the maximum variance for all vertices belonging to a cluster T_r as denoted as σ_r^2. We usually denote a matrix of random variables by $\widehat{X}$ and the expectation of $\widehat{X}$ as $X = \mathbf{E}\left[\widehat{X}\right]$. We will also denote vectors by boldface, e.g. $\mathbf{x}, \mathbf{u}$ being vectors. $\mathbf{x}$ has the i^{th} coordinate $\mathbf{x}(i)$. For a matrix X, X_i denotes the column i.

Obviously, we will need to assume that the clusters are sufficiently separated in order for us to be able to learn them. Our separation condition for the clusters is similar to that of [18].

Separation Condition. Each of the variances σ_r satisfies $\sigma_r^2 \geq \log^6 n/n$. Furthermore, there exists a large enough constant c such that for vertices $u \in T_r$ and $v \in T_s$, the columns A_u and A_v of the probability matrix A corresponding to different clusters T_r and T_s satisfy

$$\|A_u - A_v\|_2^2 \geq 64c^2k^5 \left(\sigma_r + \sigma_s\right)^2 \frac{\log(n)}{w_{\min}} \tag{1}$$

For clarity of exposition, we will make no attempt to optimize the constants or exponents of k. Similarly, we will ignore the term $w_{\min}$ for the most part, as it is a constant in our model (and can be easily incorporated in the proofs presented). Given the above separation, our aim is to present a recursive algorithm based on the singular vectors in order to learn the clusters T_r. As a result of our recursive algorithm, at each stage, we will have a partitioning of the set of vertices. We say that a partitioning $(S_1, \ldots, S_l)$, respects the original clustering if the vertices of each T_r lie wholly in any one of the S_j. We will refer to the parts S_j as super-clusters, being the union of one or more clusters T_r. We say that a partitioning $(S_1, \ldots, S_l)$ agrees with the underlying clusters if each S_i is exactly equal to some T_r (i.e. $l = k$). The aim is to prove the following theorem.

Theorem 1. *Given $\widehat{A}$ that is generated as above, i.e. $A = \mathbf{E}\left[\widehat{A}\right]$ satisfies condition 1, we can cluster the vertices such that the partitioning agrees with the underlying clusters with probability at least $1 - \frac{1}{n^\delta}$, for suitably large δ.*

3 Related Work

The second eigenvector technique has been analyzed before, but mostly from the viewpoint of constructing cuts in the graph that have a small ratio of edges cut to vertices separated. There has been a series of results [13, 2, 5, 19] relating the gap between the first and second eigenvalues, known as the Fiedler gap, to the quality of the cut induced by the second eigenvector. Spielman and Teng [20] demonstrated that the second eigenvector partitioning heuristic is good for meshes and planar graphs. Kannan et al. [17] gave a bicriteria approximation for clustering using the second eigenvector method. Cheng et al. [7] showed how to use the second eigenvector method combined

with a particular cluster objective function in order to devise a divide and merge algorithm for spectral clustering. In the random graph setting, there has been results by Alon et al. [3], and Coja-oghlan [8] in using the coordinates of the second eigenvector in order to perform coloring and bisection. In each of these algorithms, however, the cleanup phase is very specific to the particular clustering task at hand, and in particular is not similar to our algorithm.

Experimental studies done on the relative benefits of the two heuristics often show that the two techniques outperform each other on different data sets [21]. In fact results by Meila et al.[21] demonstrate that the recursive methods using the second eigenvector are actually more stable than the multiway spectral clustering methods if the noise is high. The recursive partitioning done by the second eigenvector technique can also be consider as a hierarchical clustering technique. Experimental results by Zhao et al. [23] show that recursive clustering using the second eigenvector performs better than a number of other hierarchical clustering algorithms.

4 Algorithm

4.1 Algorithm Sketch

For the sake of simplicity, in most of the paper, we will be discussing the basic bipartitioning step that is at the core of of our algorithm. In Section 4.3 we will describe how to apply it recursively to learn all the k clusters.

Define the matrix $J = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$. Note that for any vector $\mathbf{z}$ such that $\sum_i \mathbf{z}_i = 0$, i.e. $\mathbf{z}$ is orthogonal to the all ones vector, $J\mathbf{z} = \mathbf{z}$. Given the original matrix $\widehat{A}$ we will create $\Theta(n/(k \log n))$ submatrices by partitioning the set of rows into $\Theta(n/(k \log n))$ parts randomly. Suppose $\widehat{C}$ denotes any one of these parts. Given the matrix $\widehat{C}$ as input, we will first find the top right singular vector $\mathbf{u}$ of the matrix $\widehat{C}J$. The coordinates of this vector will induce a mapping from the columns (vertices) of $\widehat{C}$ to the real numbers. We will find a large "gap" such that substantial number of vertices are mapped to both sides of the gap. This gives us a natural bipartition of the set of vertices of $\widehat{C}$. We will prove that this classifies all vertices correctly, except possibly a small fraction. This will be shown in Lemmas 3 to 7 in section 4.2.

We next need to "clean up" this bi-partitioning, and this will be done using a correlation graph construction along with a Chernoff bound. For this, we utilize the bi-partitions that we get from the different random submatrices created, and combine them to get a clean bi-partitioning. The algorithm and a proof will be furnished in Lemma 8. This completes one stage of recursion in which we create a number of superclusters all of which respect the original clustering. Subsequent stages proceed similarly, by just working on the appropriate set of columns i.e. vertices corresponding to each supercluster. In what follows, we will be using the terms "column" and "vertex" interchangeably, noting that vertex x corresponds to column $\widehat{C}_x$.

4.2 Proof

For the standard linear algebraic techniques used in this section, we refer the reader to [16]. Recall that each $\widehat{C}$ is a $\frac{n}{2k \log n} \times n$ matrix where the rows are chosen randomly. Denote the expectation of $\widehat{C}$ by $\mathbf{E}\left[\widehat{C}\right] = C$, and by $\mathbf{u}$ the top right singular vector of $\widehat{C}J$, i.e. the top eigenvector of $(\widehat{C}J)^T\widehat{C}J$. In what follows, we demonstrate that for each of random submatrices $\widehat{C}$, we can utilize the second right singular vector $\mathbf{u}$ to create a partitioning of the columns of $\widehat{C}J$ that respects the

original clustering. The following fact is intuitive and will be proven later in lemma 9, when we illustrate the full algorithm.

Fact 2. *$\widehat{C}$ has at least $\frac{n_r}{2k\log n}$ rows for each cluster T_r.*

Let $\sigma = \max_r\{\sigma_r\}$, where the maximum is taken only over clusters present in $\widehat{C}$ (and therefore, potentially much smaller than $\sigma_{\max}$). We also denote $C(r,s)$ for the entries of C corresponding to vertices of T_r and T_s. The following result is from Furedi-Komlos and more recently, Vu [22, 15] claiming that a matrix of i.i.d. random variables is close to its expectation in the spectral norm.

Lemma 3. (Furedi, Komlos; Vu) *If $\widehat{X}$ is a 0/1 random matrix with expectation $X = \mathbf{E}\left[\widehat{X}\right]$, and the maximum variance of the entries of $\widehat{X}$ is σ^2 which satisfies $\sigma^2 \geq \log^6 n/n$,[1] then with probability $1 - o(1)$,*

$$\|X - \widehat{X}\|_2 < 3\sigma\sqrt{n}$$

In particular, we have $\|C - \widehat{C}\|_2 < 3\sigma\sqrt{n}$.

We now embark into a series of lemmas whose final goal is to show that the top right singular vector $\mathbf{u}$ of $\widehat{C}J$ gives us an approximately good bi-partition. First we claim that the topmost singular value of the expectation matrix CJ is large relative to the size of the perturbation that will be applied to it.

Lemma 4. *The first singular value λ_1 of the expected matrix CJ satisfies $\lambda_1(CJ) \geq 2c(\sigma_r+\sigma_s)k^2\sqrt{n}$ for each pair of clusters r and s that belong to C. Thus, in particular, $\lambda_1(CJ) \geq 2c\nu k^2\sqrt{n}$.*

Proof. Suppose $\widehat{C}$ has the clusters T_r and T_s, $r \neq s$. Assume $n_r \leq n_s$. Consider the vector $\mathbf{z}$ defined as :

$$\mathbf{z}_x = \begin{cases} \frac{1}{\sqrt{2n_r}} & \text{if } x \in T_r \\ -\frac{\sqrt{n_r}}{n_s\sqrt{2}} & \text{if } x \in T_s \\ 0 & \text{otherwise} \end{cases}$$

Now, $\sum_x \mathbf{z}(x) = \frac{n_r}{\sqrt{2n_r}} - \frac{\sqrt{n_r}}{\sqrt{2}n_s}n_s = 0$. Also, $\|\mathbf{z}\|^2 = \frac{n_r}{2n_r} + \frac{n_r n_s}{2n_s^2} = \frac{1}{2} + \frac{1}{2}\frac{n_r}{n_s} \leq 1$. Clearly, $\|\mathbf{z}\| \leq 1$. For any row C^j from a cluster T_t, it can be shown that $C^j \cdot \mathbf{z} = \sqrt{\frac{n_r}{2}}(C(r,t) - C(s,t))$. We also know from fact 2 that there are at least $n_t/(2k\log n)$ such rows. Now,

$$\begin{aligned} \|CJ\mathbf{z}\|^2 &\geq \sum_j (C^j \cdot \mathbf{z})^2 = \sum_t \sum_{j \in T_t} (C^j \cdot \mathbf{z})^2 \\ &\geq \sum_t \frac{n_t}{2k\log n}\frac{n_r}{2}(C(r,t) - C(s,t))^2 = \frac{n_r}{4k\log n}\sum_t n_t(C(r,t) - C(s,t))^2 \\ &= \frac{n_r}{4k\log n}\|C_r - C_s\|_2^2 \geq 64\frac{n_r}{4k\log n}c^2k^5\left(\sigma_r + \sigma_s\right)^2\log(n)/w_{\min} \geq 16c^2nk^4(\sigma_r+\sigma_s)^2 \end{aligned}$$

using the separation condition and the fact that n_r is at least $w_{\min}n$. And thus $\lambda_1(CJ)$ is at least $4c(\sigma_r + \sigma_s)k^2\sqrt{n}$. Note that the 4th step uses the separation condition (1). □ □

[1] In fact, in light of recent results in [12] this holds for $\sigma^2 \geq C'\log n/n$, with a different constant in the concentration bound.

The above result, combined with the fact the the spectral norm of the random perturbation being small immediately implies that the norm of the matrix $\widehat{C}J$ is large too. Thus,

Lemma 5. *The top singular value of $\widehat{C}J$ is at least $c\nu k^2\sqrt{n}$.*

Proof. Suppose $\mathbf{z}$ is defined as in Lemma 4. Then, $J\mathbf{z} = \mathbf{z}$. Now,

$$\begin{aligned}\|\widehat{C}J\mathbf{z}\| &= \|(C - (C - \widehat{C}))J\mathbf{z}\| \geq \|CJ\mathbf{z}\| - \|(C - \widehat{C})\mathbf{z}\| \\ &\geq \|CJ\mathbf{z}\| - \|C - \widehat{C}\|\|\mathbf{z}\| \geq \|CJ\mathbf{z}\| - \|C - \widehat{C}\| \\ &\geq \|CJ\mathbf{z}\| - \|C - \widehat{C}\|\end{aligned}$$

For the last step note that $\|\mathbf{z}\| \leq 1$. Applying the results of Lemma 3 and Lemma 4, we have that, if the constant $c > 1$, $\|\widehat{C}J\mathbf{z}\| \geq c\sigma k^2\sqrt{n}$. □ □

We now claim that the coordinates $\mathbf{u}(x)$ of the singular vector $\mathbf{u}$ of $\widehat{C}J$ are almost constant on each cluster. This will then enable us to use this singular vector for bi-partition.

Lemma 6. *The vector $\mathbf{u}$, the top right singular vector of $\widehat{C}J$ can be written as $\mathbf{u} = \mathbf{v}+\mathbf{w}$ where both $\mathbf{v}, \mathbf{w}$ are orthogonal to $\mathbf{1}$ and further, $\mathbf{v}$ is a linear combination of the indicator vectors $\mathbf{g}^{(1)}, \mathbf{g}^{(2)}, \ldots$ for clusters T_r that have vertices in the columns of $\widehat{C}$. Also, $\mathbf{w}$ sums to zero on each T_r. Moreover,*

$$\|\mathbf{w}\| \leq \frac{4}{ck^2} \tag{2}$$

Proof. We may define the two vectors $\mathbf{v}$ and $\mathbf{w}$ as follows.

$$\mathbf{v} = \sum_r (\mathbf{g}^{(r)} \cdot \mathbf{u})\mathbf{g}^{(r)}, \quad \mathbf{w} = \mathbf{u} - \mathbf{v}.$$

It is easy to check that $\mathbf{w}$ is orthogonal to $\mathbf{v}$, and that $\sum_{x\in T_r} \mathbf{w}(x) = 0$ on every cluster T_r. Thus both $\mathbf{v}$ and $\mathbf{w}$ are orthogonal to $\mathbf{1}$. As $\mathbf{v}$ is orthogonal to $\mathbf{w}$, $\|\mathbf{v}\|^2 + \|\mathbf{w}\|^2 = \|\mathbf{u}\|^2 = 1$. Now,

$$\begin{aligned}\lambda_1(\widehat{C}J) = \|\widehat{C}J\mathbf{u}\| &\leq \|\widehat{C}J\mathbf{v}\| + \|\widehat{C}J\mathbf{w}\| \\ &\leq \lambda_1(\widehat{C}J)\|\mathbf{v}\| + \|CJ\mathbf{w}\| + \|CJ - \widehat{C}J\|\|\mathbf{w}\| \\ &\leq \lambda_1(\widehat{C}J)(1 - \|\mathbf{w}\|^2/2) + \|C - \widehat{C}\|\|\mathbf{w}\|\end{aligned}$$

using the fact that $(1-x)^{1/2} \leq 1 - \frac{x}{2}$ for $0 \leq x \leq 1$, and also noting that $J\mathbf{w} = \mathbf{w}$, and therefore $CJ\mathbf{w} = C\mathbf{w} = 0$. Thus, from the above,

$$\|\mathbf{w}\| \leq \frac{2\|C - \widehat{C}\|}{\lambda_1(\widehat{C}J)} \leq \frac{4\sigma\sqrt{n}}{c\nu k^2\sqrt{n}} \leq \frac{4}{ck^2}$$

using Lemma 2 and Lemma 5. □ □

We now show that in bi-partitioning each $\widehat{C}$ using the vector $\mathbf{u}$, we only make mistakes for a small fraction of the columns.

Lemma 7. *Given the top right singular vector* $\mathbf{u}$ *of* $\widehat{C}$*, there is a way to bipartition the columns of* $\widehat{C}$ *based on* $\mathbf{u}$*, such that all but* $\frac{n_{\min}}{ck}$ *columns respect the underlying clustering of the probability matrix* C*.*

Proof. Consider the following algorithm. Consider the real values $\mathbf{u}(x)$ corresponding to the columns $\widehat{C}_x$.

1. Find β such that at most $\frac{n}{ck^2}$ of the $\mathbf{u}(x)$ lies in $(\beta, \beta + \frac{2}{k\sqrt{n}})$. Moreover, define $L = \{x : \mathbf{u}(x) < \beta + \frac{1}{k\sqrt{n}}\}; R = \{x : \mathbf{u}(x) \geq \beta + \frac{1}{k\sqrt{n}}\}$. It must be that both $|L|$ and $|R|$ are at least $n_{\min}/2$. Note that $\widehat{C} = L \cup R$. If we cannot find any such gap, don't proceed (a cluster has been found that can't be partitioned further).

2. Take $L \cup R$ as the bipartition.

We must show that, if the vertices contain at least two clusters, a gap of $\frac{2}{k\sqrt{n}}$ exists with at least $n_{\min}/2$ vertices on each side. For simplicity, for this proof we assume that all clusters are of equal size (the general case will be quite similar). Let $\mathbf{v} = \sum_{r=1}^{k} \alpha_r \mathbf{g}^{(r)}$. Recall that $\mathbf{v}$ is orthogonal to $\mathbf{1}$, and thus $\sum_{r=1}^{k} \alpha_r \sqrt{\frac{k}{n}} = 0$. Now note that $1 = \|\mathbf{u}\|^2 = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2$. This and lemma 6 gives us

$$\sum_{r=1}^{k} \alpha_r^2 \geq 1 - \frac{16}{c^2 k} \geq \frac{1}{2} \tag{3}$$

We now show that there is an interval of $\Theta\left(\frac{1}{k\sqrt{k}}\right)$ on the real line such that no α_r lies in this interval and at least one α_r lies on each side of the interval. We will call such a gap a "proper gap". Note that a proper gap will partition the set of vertices into two parts such that there are at least $n_{\min}/2$ vertices on each side of it.

The above claim can be proved using basic algebra. Assume that the maximum gap between the α_r's is Δ. Since α_r's sum to zero, there has to be two α_r's, one positive and one negative, with absolute value most Δ. Without loss of generality, if $\alpha_1 \leq \ldots \leq \alpha_i < \alpha_{i+1} = 0 = ... = \alpha_{j-1} < \alpha_j \leq \ldots \leq \alpha_k$, then clearly, $\alpha_i^2 + \alpha_j^2 \leq \Delta^2$. Further, the consecutive separations are each smaller than Δ. Thus, the norm of the vector α can be bounded by $\Delta^2 + 2\Delta^2 \sum_{i<=k} i^2 \leq \Delta^2 k^3$. Which together with equation 3 gives us the desired claim that $\Delta \geq \Theta\left(\frac{1}{k\sqrt{k}}\right)$.

Thus, it can been seen easily that for some constant c, there will be a proper gap of $\frac{1}{ck\sqrt{n}}$ in the vector $\mathbf{v}$. We then argue that most of the coordinates of $\mathbf{w}$ are small and do not spoil the gap. Since the norm of $\|\mathbf{w}\|^2$ is bounded by $16/(c^2k^4)$, it is straightforward to show that at most $\frac{n}{ck^2}$ vertices x can have $\mathbf{w}(x)$ over $\frac{4}{k\sqrt{cn}}$. This shows that for most vertices $\mathbf{w}(x)$ is small and will not "spoil" the proper gap in $\mathbf{v}$. Thus, with high probability, the above algorithm of finding a gap in $\mathbf{u}$ always succeeds. Next we have to show that any such gap that is found from $\mathbf{u}$ actually corresponds to a proper gap in $\mathbf{v}$. This essentially follows from the definition of the gap in $\mathbf{u}$. Since there must be at least $n_{\min}/2$ vertices on each side of the gap in $\mathbf{u}$, and since the values $\mathbf{u}(x)$ and $\mathbf{v}(x)$ are close (i.e. $\mathbf{w}(x) = \mathbf{u}(x) - \mathbf{v}(x)$ is smaller than $1/(2k\sqrt{n})$) except for $\frac{n}{ck}$ vertices, it follows that a proper gap found in $\mathbf{u}$ must correspond to a proper gap in $\mathbf{v}$. Thus the only vertices that can be misclassified using this bi-partition are the vertices that are either in the gap, or have $\mathbf{w}(x)$ larger than $\frac{1}{k\sqrt{n}}$. Given this claim, it can be seen a using a proper gap a bi-partition of the vertices can be found with at most $\frac{n}{2ck^2} \approx \Theta\left(\frac{n_{\min}}{ck}\right)$ vertices on the wrong side of the gap. □ □

Only the "clean up" phase now remains. A natural idea would be to use $\log n$ independent samples of $\widehat{C}$ (thus requiring the $\log n$ factor in the separation) and try to use a Chernoff bound argument. This argument doesn't work, unfortunately, the reason being that the singular vector can induce different bi-partitions for each of the $\widehat{C}$'s. For instance, if there are 3 clusters in the original data, then in the first step we could split any one of the three clusters from the other two. This means a naive approach will need to account for all possible bi-partitionings and hence require an extra 2^k in the separation condition. The way we deal with this is to construct a correlation graph combining $O(\log n)$ bi-partitions, where vertices are connected iff they were placed together for "most" samples. Intuitively, we utilize the fact that although the second singular vector itself is not stable, there is only a small number of "configurations" possible for this singular vector. This intuition is made rigorous in the following lemma.

Lemma 8. *Suppose we are given set V that is the union of a number of clusters $T_1 \cup \ldots \cup T_t$. Given $p = ck \log n$ independent bi-partitions of the set of columns V, such that each bi-partition agrees with the underlying clusters for all but $\frac{n_{\min}}{4ck}$ vertices, there exists an algorithm that, with high probability, will compute a partitioning of the set V such that*

- *The partitioning respects the underlying clusters of the set V.*
- *The partitioning is non-trivial, that is, if the set V contains at least two clusters, then the algorithm finds at least two partitions.*

Proof. Consider the following algorithm. Denote $\varepsilon = \frac{1}{4ck}$.

1. Construct a (correlation) graph H over the vertex set V.
2. Two vertices x and y are adjacent if they are on the same L or R for at least $(1-2\varepsilon)$ fraction of the bi-partitions.
3. Let $N_1, \ldots, N_l$ be the connected components of this graph. Return $N_1, \ldots, N_l$.

We now need to prove that the following claims hold with high probability :

- Each N_j respects the cluster boundary, i.e. each cluster T_r that is present in V satisfies $T_r \subseteq N_{j_r}$ for some j_r.
- If there are at least two clusters present in V, i.e. $t \geq 2$, then there are at least two components in H.

For two vertices $x, y \in H$, let the **support** $s(x, y)$ equal the fraction of tests such that x and y are on the same side of the bi-partition. For the first claim, we define a vertex x to be a "bad" vertex for the i^{th} test if $|w(x)| > \frac{1}{k\sqrt{cn}}$. From lemma 7 the number of bad vertices is clearly at most $\frac{1}{ck} n_{\min}$. It is clear that a misclassified vertex x must either lie in the gap $(\beta, \beta + \frac{2}{k\sqrt{n}})$ or it must be a bad one. So for any vertex x, the probability that x is misclassified in the i^{th} test is at most $\varepsilon = 1/(4ck)$. If there are p tests, then the expected times that a vertex x is misclassified is at most εp. To show the first part of our claim we just need to use Chernoff bounds. Supposing Y_x^i is the indicator random variable for the vertex x being misclassified in the i^{th} test. Thus,

$$\mathbf{Pr}\left[\sum_i Y_x^i > 2\varepsilon p\right] < \exp\left(-\frac{16p}{ck}\right) < \frac{1}{n^3}$$

since $p = ck \log n$. Thus, each pair of vertices in a cluster, are on the same side of the bipartition for at least $(1 - 2\varepsilon)$ fraction of the tests. Thus, the components N_j always obey the cluster partitions.

Next, we have to show that we have at least two components in the correlation graph. For contradiction, assume there is only one connected component. We know, that if $x, y \in T_r$ for some r, the fraction of tests on which they landed on same side of partition is $s(x, y) \geq (1 - 2\varepsilon)$. Hence the subgraph induced by each T_r is complete. With at most k clusters in V, this means that any two vertices x, y (not necessarily in the same cluster) are separated by a path of length at most k. Clearly $s(x, y) \geq (1 - 2k\varepsilon)$. Hence, the total support of inter-cluster vertex pairs is

$$\sum_{r \neq s} \sum_{x \in T_r, y \in T_s} s(x, y) \quad \geq \quad (1 - 2k\varepsilon) \sum_{r \neq s} n_r n_s \geq \sum_{r \neq s} n_r n_s - 2k\varepsilon \sum_{r \neq s} n_r n_s. \tag{4}$$

Let us count this same quantity by another method. From Lemma 7, it is clear that for each test at least one cluster was separated from the rest (apart from small errors). Since by the above argument, all but ε vertices are good, we have that, at least $n_{\min}(1 - \varepsilon)$ vertices were separated from the rest. Hence the total support is

$$\sum_{r \neq s} \sum_{x \in T_r, y \in T_s} s(x, y) \leq \sum_{r \neq s} n_r n_s - n_{\min}(1 - \varepsilon)(n - n_{\min}(1 - \varepsilon)) < \sum_{r \neq s} n_r n_s - n_{\min} n/2$$

But this contradicts equation 4 if

$$2k\varepsilon \sum_{r \neq s} n_r n_s < n_{\min} n/2$$

i.e. $\varepsilon < \frac{n_{\min} n/4}{k \sum_{r \neq s} n_r n_s} < \frac{n_{\min} n/2}{kn^2} \leq \frac{1}{2ck}$. Hence, with the choice of $\varepsilon = 1/(4ck)$, we get a contradiction, and so must have more than one component in the correlation graph. Hence, with the error probability of $\frac{1}{4ck}$ for each bi-partition, and $p \geq ck \log n$ independent bi-partitions, the correlation graph satisfies the properties claimed. □ □

4.3 Final Algorithm

This section describes the complete algorithm. Basically, it is the bi-partitioning technique presented in the previous section repeated (at most) k times applied to the matrix $\widehat{A}$. As the split in every level is "clean", as we have shown above, the whole analysis goes through for recursive steps without any problems.

In order to de-condition the steps of the recursion, we have to first create k independent instances of the data by partitioning the rows of the matrix $\widehat{A}$ into a k equally sized randomly chosen sets. This creates a collection of rectangular matrices $\widehat{B}_1, \ldots, \widehat{B}_k$, each belonging to $\Re^{n/k \times n}$. Each matrix $\widehat{B}_i$ will be used for only one stage of recursion. At each stage of the recursion, we will have in our hands a set of l superclusters $S_1, \ldots, S_l$ of the columns, where l is less than or equal to k. In order to further subdivide this super-clustering, for each supercluster S_j, we invoke the module **Bi-Partition** on appropriate columns of any one of the matrices $\widehat{B}_{i_j}$.

The module **Bi-Partition**$(\widehat{X}, k)$ on being invoked with the matrix $\widehat{X}$ and the cluster parameter k consists of two phases: an approximate partitioning by the singular vector, followed by a clean-up phase. The rows of matrix $\widehat{X}$ are further subdivided to create a number of rectangular matrices $\widehat{C}^{(i)}$. First we partition the columns of each of these rectangular matrices $\widehat{C}^{(i)}$ using the top right

Algorithm 1 Cluster $(\widehat{A}, k)$

Partition the set of rows into k random equal parts, each part to be used in the corresponding step of recursion. Name the i^{th} part to be $\widehat{B}_i$.
Let $(S_1, \ldots, S_l)$ =**Bi-Partition** $(\widehat{B}_1, k)$.
Recursively call **Bi-Partition** on each of S_i, and on each of the results, using the appropriate columns of a separate $\widehat{B}_j$ for each call. The recursion ends when the current call returns only one S_i. Let $\widehat{T}_1, \ldots, \widehat{T}_k$ be the final groups.

Algorithm 2 Bi-Partition $(\widehat{X}, k)$

Partition the set of rows into $c_1 \log n$ equal parts randomly. The i^{th} set of rows forms the matrix $\widehat{C}^{(i)}$.
For each $\widehat{C}^{(i)}$, find the right singular vector of $\widehat{C}^{(i)}J$ and call it $\mathbf{u}_i$.
Split:
Find a proper gap β, such that $(\beta, \beta + \frac{2}{k\sqrt{n}})$ has at most $\frac{n}{c_2 k}$ vertices and define

$$L_i = \{x : u_i(x) < \beta + \frac{1}{k\sqrt{n}}\}$$

$$R_i = \{x : u_i(x) \geq \beta + \frac{1}{k\sqrt{n}}\}$$

$$|L_i| \geq n_{\min}/2; |R_i| \geq n_{\min}/2$$

$$\widehat{C}^{(i)} = L_i \cup R_i$$

If no such gap exists, return.
Cleanup:
Construct a (correlation) graph with the columns of $\widehat{X}$ as the vertices.
Connect two vertices x and y if they are on the same L_i or R_i for at least $(1 - \frac{1}{2ck}) \log n$ times.
Let $N_1, \ldots, N_l$ be the connected components of this graph. Return $N_1, \ldots, N_l$.

singular vector of $\widehat{C}^{(i)}J$. The different bi-partitions are then combined as in Lemma 8 to create a final clean bi-partition of the set of columns (vertices) in $\widehat{X}$.

One thing we still need to prove that the fact 2 made for $\hat{C}$ in the beginning of section 4.2 is valid for $C^{(i)}$

Lemma 9. *Consider each matrix $C^{(i)} = \mathbf{E}\left[\widehat{C}^{(i)}\right]$. W.h.p. there are at least $\frac{n_r}{2c_1 k \log n}$ rows in $C^{(i)}$ corresponding to T_r.*

Proof. In each $\widehat{C}^{(i)}$, the expected number of rows from each T_r is $\frac{n_j}{k \times c_1 \log n}$. Using Chernoff bound, the number of rows contributed by each cluster T_r to the matrix $\widehat{C}^{(i)}$ is at least $\frac{n_j}{2c_1 k \log n}$ with probability $1 - \exp[-\frac{n_j}{2c_1 k \log n}] \geq 1 - \frac{1}{n^3}$. Thus, over the all random partitions, w.p. $1 - \frac{1}{n^2}$, the statement is true. □

5 Skewed degree graphs

In this section, we extend our clustering algorithm for graphs with skewed degree distributions. Our model for a skewed degree random graph with structure is same as that of [10]. That is, in addition to the block structured probability matrix A, we now have a set of degrees $\{d_u, u = 1 \ldots n\}$. The rescaled set of probabilities are given in the new probability matrix G, defined as $M_{uv} = d_u A_{uv} d_v$. Thus, $M = DAD$. The actual graph is represented by the matrix $\widehat{M}$, that is generated by independently rounding the $(u, v)^{th}$ entry to 1 with probability M_{uv} and 0 with probability $1 - M_{uv}$. Our aim is then to recover the clusters $T_1, T_2, \ldots T_k$ from the matrix $\widehat{M}$.

The intuition behind this algorithm is similar to that of [10]. We need to rescale the variances, so that the error introduced by clustering is a function of the average degree of the graph, rather than the maximum degree. In order to achieve this, [10] perform a scaling of the matrix $\widehat{M}$ and work with the normalized Laplacian $\widehat{L} = D^{-1/2}\widehat{M}D^{-1/2}$. As expected, the intuition carries through for us too.

Our algorithm for the bipartition follows the same structure as the previous case, with two changes. We simply compute call the algorithm **Cluster** with the matrix $\widehat{L}$. Secondly, instead of using the topmost singular vector $\mathbf{u}$ to bi-partition the rows in step 1 of **Partition**, we use the vector $D^{-1/2}\mathbf{u}$.

6 Sparse Graphs

6.1 Model

The input $\hat{A}$ is a n-vertex undirected graph. There will be k clusters in the graph, with $n_r = \Omega(n)$ being the size of cluster T_r. Let x $\in T_r$. Then we assume that the number of edges from x to vertices of T_s:

$$e(x, T_s) = d_{rs} \tag{5}$$

For some constant d_{rs}. We assume that these constants satisfy $n_r d_{rs} = n_s d_{sr}$.

Let $\hat{A}^{(rs)}$ be the submatrix of $\hat{A}$ containg rows corresponding to T_r and columns corresponding to T_s. Then $\hat{A}^{(rs)}$ is a matrix randomly chosen from all matrices satisfying equation 5 (to account for symmetry $\hat{A}^{(rs)} = (\hat{A}^{(rs)})^T$). For $n_r = n_s$, [14] provides an efficient way to generate such a

matrix, but this does not of much concern here, as we assume that the data is given to us (one of the motivations for [14] was efficient constructions of an expander).

Let $A = \mathbf{E}\left[\hat{A}\right]$. If vertex $x \in T_r$, let $A_x = \mu_r$. It is easy to see that $\mu_r(y) = \frac{d_{rs}}{n_s}$ if $y \in T_s$ Note that $\mu_r(x) = \mu_s(y)$ where $y \in T_s; x \in T_r$ due to symmetry. Let d be a upper bound for vertex degree in the graph.

We will assume, for all r, and some constant c_0

$$d_{rr} \geq \frac{1}{2}d + c_0\sqrt{dk} \tag{6}$$

Which will now imply something we need

$$\|\mu_r - \mu_s\|_2^2 \geq c_0^2 k^2 \frac{d}{n} \tag{7}$$

We seek to find the clusters. It is instructive to note that compared to the non-sparse, we don't have an $\log n$ term in the seperation, neither the possibility of using independence of samples (entries are not independent).

6.2 Related work

Among previous works on "sparse" graphs are results by Alon and Kahale [3] (3 coloring) and Coja-Oghlan [8, 9] (bisection, clustering). Our results are not comparable to theirs as those models are only sparse "on average". Nevertheless, the gap required in the latter, improving on [5], is $np' - np = \Theta(\sqrt{np' \log np'})$ in a $G(n, p')$ model, which put in our terminology is $\Theta(\sqrt{d \log d})$, similar to our seperation (infact we don't need the $\log d$ factor). The seperation in [3] is d. It should be emphasized again that both settings are quite different from ours.

A d-regular model for bisection was studied by Bui et. al. [6]. They present an algorithm that finds bisections of width (cardinality of the bisection) $o(n^{1-1/(d/2)})$ from a graph that is randomly chosen from d-regular graphs having such a bisection. We depend on having different d_{rr} for different cluster for a notion of partitioning, and in any case we seek to solve a more general problem.

6.3 Algorithm

For sets (of vertices) U and W, let $e(U, W)$ be the the number of edges between U and W. The module **Cluster** presented earlier remains the same. The change is that module **Partition** gets replaced by **SparsePartition**.

Algorithm 3 SparsePartition(A)

Find the right singular vector of AJ, and call it $\mathbf{u}$.
Find a gap of size $\frac{2c}{\sqrt{n}}$ such that there is no vertex with $u(x)$ in this gap.
Create two partitions with vertices from each side of this gap
Let $V = P_1' \cup P_2'$ to be the bipartitioning. Test that both P_1' and P_2' have at least $\frac{n_{\min}}{2}$ vertices. Return otherwise.
Call **SparseCleanup**(P_1', P_2', d)

The algorithm **SparseCleanup** is substantially different from the cleanup phase presented earlier.

Our main theorem is

Algorithm 4 SparseCleanup(P_1, P_2, d)

{Stage one}
loop
  find a vertex v in P_2' such that $e(v, P_1') > e(v, P_2')(1 + \frac{1}{2\sqrt{d}})$
  if no vertex can be found, end loop.
  move v to P_1'
end loop
{Stage two}
loop
  find a vertex v in P_1' such that $e(v, P_2') > e(v, P_1')(1 + \frac{1}{2\sqrt{d}})$
  if no vertex can be found, end loop.
  move v to P_2'
end loop

Theorem 10. *If d is a large enough constant, and $\hat{A}$ is randomly generated as described in the model, SparsePartiton($\hat{A}$) will successfully bipartition the the columns $\hat{A}_x$ of the data along cluster lines.*

Proof. The proof crucially uses expansion properties of the model for a cleanup to be possible. We refer the reader to the appendix for details. □

Proofs (Sparse model)

Again, for simplicty we will concentrate on proving that a single step of bipartitioning works cleanly. Given that, its easy to see that the recursive steps work.

We start by quoting the following result due to Friedman, Kahn and Szemeredi [14].

Theorem 11. (Friedman, Kahn, Szemeredi) *Let A be the adjacency matrix of a random d regular graph. Then, almost surely,*

$$\sigma_2(A) \leq 3\sqrt{d}$$

Where, $\sigma_2(A)$ is the second largest singular value of A.

From this it follows that

Lemma 12. *With high probability, only ϵm (or less) objects will be misclassified after in algorithm 3 before calling* **SparseCleanUp**.

Proof. This will follow the trajectory of the lemmas 4 to 7. The main change will the use of theorem 11 instead of theorem 3. □

We now need to show that the cleanup phase works. Though the method makes intuitive sense, we have to prove that small subsets of vertices of a cluster will not have most of their edges within themselves, because then the cleanup phase will not succeed. For example, if misclassified U is such that $|U| = d + 1$, and $e(U, U) = d^2/2$, then cleanup phase will fail. To prove that such a situation does not arise, we will expansion properties of the graph.

The following lemma is from [14].

Lemma 13. *There is a constant C such that with probability $1 - n^{-\Omega(\sqrt{d})}$* **every pair** *$A, B \subset [n]$ satisfies (at least) one of*

(a) $e(A, B) \leq \mu(A, B)$

(b) $e(A, B) \log \frac{e(A,B)}{\mu(A,B)} \leq C|B| \log \frac{n}{|B|}$

(c) $|A|$, $|B|$ *and* $e(A, B)$ *are all at most* $C\sqrt{d}$

Here, $e(A, B)$ is the number of edges from A to B, and $\mu(A, B) = \frac{|A||B|d}{n}$ is the expected value of $e(A, B)$.

In what follows, we will assume d is much larger than C. Now we shall prove the following lemma.

Lemma 14. *Let, V be a subset of vertices of the input graph. For large enough value of d, and c_1, if $|V| \leq \frac{n}{c_1\sqrt{d}}$, then*

$$e(V, V) \leq |V| \frac{\sqrt{d}}{10}$$

Proof. It is easy to see that if cases (a) or (c) is true in lemma 13, we are already done. So we will deal with case (b).

Let, $|V| = \frac{n^\alpha}{c_1\sqrt{d}}$ where $1 \geq \alpha \geq \frac{c_{10}}{\log n}$, for some appropriate constants $c_1 \gg C$ and c_{10}. All possible sizes of V are covered by this, except very small sizes for which the theorem is trivially true. Now,

$$\begin{aligned} \frac{e(V,V)}{\mu(V,V)} \log \frac{e(V,V)}{\mu(V,V)} &\leq \frac{C|V|n}{|V||V|d} \log \frac{n}{|V|} \\ &= \frac{Cc_1}{\sqrt{d}} n^{1-\alpha} \log (n^{1-\alpha} c_1 \sqrt{d}) \end{aligned} \tag{8}$$

Now we claim,

$$\frac{e(V,V)}{\mu(V,V)} \leq \frac{c_1}{10} n^{1-\alpha}$$

Because if not,

$$\frac{e(V,V)}{\mu(V,V)} \log \frac{e(V,V)}{\mu(V,V)} > \frac{c_1}{10} n^{1-\alpha} \log (\frac{c_1}{10} n^{1-\alpha})$$

Which will fail to meet condition 8 if c_1 and d is large enough (basically due to $\sqrt{d}$ being much larger than $\log d$). Hence,

$$\begin{aligned} e(V, V) &\leq \frac{c_1}{10} n^{1-\alpha} \frac{d}{n} |V||V| \\ &\leq \frac{c_1}{10} n^{1-\alpha} \frac{d}{n} \frac{n^\alpha}{c_1\sqrt{d}} |V| \\ &\leq \frac{\sqrt{d}}{10} |V| \end{aligned}$$

This completes the proof of the lemma.

□

Now we introduce some more terminology. As only a small fraction of vertices are miscalssified in $P_i'; i = 1, 2$, it makes sense to define P_i as the correct bipartitioning (one that respects cluster boundaries) "closest" to P_i'. Let

$$E_{ri} = \{v : v \in T_r \wedge v \notin P_i' \wedge T_r \subset P_i\}$$

In other words, the subset of T_r that should be in P_i', but have been misplaced. Or, the subset of T_r mistakenly placed in P_i' that don't belong in it.

The following lemma proves that after the first stage of the cleanup, there are no vertices mistakenly placed in P_2.

Lemma 15. *Assume $T_r \subset P_1$. After Stage one of algorithm 4, E_{r1} is empty $\forall r$. Moreover, no vertex of T_r is misclassified during Stage two.*

Proof. Consider E_{r1} before cleanup starts. From lemma 12, E_{r1} fulfils the conditions of lemma 14 (as ϵ is small). Hence,

$$e(E_{r1}, E_{r1}) \leq |E_{r1}|\frac{\sqrt{d_{rr}}}{10} \leq |E_{r1}|\frac{\sqrt{d}}{10}$$

Now,

$$\begin{aligned} e(E_{r1}, P_2') &< |E_{r1}|(\frac{d}{2} - c_0\sqrt{kd}) + |E_{r1}|\frac{\sqrt{d}}{10} \\ &< |E_{r1}|(\frac{d}{2} - \sqrt{d}) \end{aligned}$$

And,

$$\begin{aligned} e(E_{r1}, P_1') &\geq e(E_{r1}, T_r - E_{r1})|E_{r1}| \geq (\frac{d}{2} + c_0\sqrt{kd}))|E_{r1}| - |E_{r1}|\frac{\sqrt{d}}{10} \\ &\geq |E_{r1}|(\frac{d}{2} + \sqrt{d}) \end{aligned}$$

Now from an averaging argument, there has to exist atleast one vertex $x \in E_{ri}$ such that $\frac{e(x,P_1')}{e(x,P_2')} > \frac{d/2+\sqrt{d}}{d/2-\sqrt{d}} > 1 + 2/\sqrt{d}$. Hence x will moved from P_2' to P_1'. Once that happens, we now have a new E_{r1} and the whole argument applies again and we a second vertex moves. Proceeding in this way, it is clear that all vertices of E_{r1} will move to P_1'.

Now, in stage two, it is clear that no vertex belonging to T_r can be re-misclassified as it will have majority of its edges in P_1'. □

The following lemma proves that after the second stage of the cleanup, there are no vertices mistakenly placed in P_1.

Lemma 16. *Assume $T_r \subset P_2$. After Stages one and two of algorithm 4, E_{r2} is empty $\forall r$.*

Proof. This might seem very similar to lemma 15, and it is, but an important difference exists. We know that E_{r2} satisfies the condition of lemma 14 **before** cleanup phase starts. But does it still satisfy the condition after stage one? Or, in other words, can E_{r2} grow substantially during stage one? We claim the answer is no, which we prove below. Given that the rest is an argument identical to lemma 15.

Let F_{r2} be the set of vertices belonging to T_r that get misclassified in the course of stage one. We argue that, $|F_{r2}| < |E_{r2}|$, which proves our claim (as this implies that the size of E_{r2} can atmost double during stage one).

For contradiction assume $|F_{r2}| \geq |E_{r2}|$. Now consider the execution of stage one until $|F_{r2}| = |E_{r2}|$. During this phase, let a vertex $x \in T_r$ moves from P_2' to P_1'. Then $e(x, P_1' \cup F_{r2}) \geq (1 + 2/\sqrt{d})e(x, P_2' - F_{r2})$. Let $e(x, E_{r2} \cup F_{r2}) = \gamma d$. Then we get,

$$\begin{aligned} e(x, P_1' \cup F_{r2}) &> e(x, P_2' - F_{r2}) \\ \Rightarrow e(x, P_1') + e(x, F_{r2}) &> e(x, P_2') - e(x, F_{r2}) \\ \Rightarrow \frac{d}{2} - c_0\sqrt{kd} + \gamma d &\geq \frac{d}{2} + c_0\sqrt{kd} - \gamma d \\ \Rightarrow 2\gamma d &\geq 2c_0\sqrt{kd} \\ \Rightarrow \gamma &\geq c_0\sqrt{k/d} \end{aligned}$$

Now this is true for every $x \in F_{r2}$. Hence,

$$\begin{aligned} e(E_{ri} \cup F_{r2}, E_{r2} \cup F_{r2}) &\geq e(F_{r2}, E_{r2} \cup F_{r2}) = \sum_{x \in F_{r2}} e(x, E_{r2} \cup F_{r2}) \\ &> \gamma d|F_{r2}| \geq c_0\sqrt{kd}|F_{r2}| = \frac{c_0}{2}\sqrt{kd}|E_{r2} \cup F_{r2}| \end{aligned}$$

But this contradicts lemma 14 (the assumption of the lemma still holds, as the size increased by only a factor of 2). □

Lemma 17. *After the cleanup phase, all vertices are correctly classified.*

Proof of theorem 10

Proof. Clear from lemmas 15 and 16. □

7 Conclusion

Our first model depends crucially on the independence of all the entries in the probability matrix. Both the random matrix bound of Lemma 3, and the technique of partitioning the matrix randomly to create $O(\log n)$ identical submatrices exploit the row-wise as well as column-wise independence. It is an interesting question, whether the technique generalizes for models with limited independence. Further, the second eigenvector technique has provable guarantees for the worst case also. Thus it will be useful to see if we can claim some bounds for the semi-random model too, where a random perturbation is applied to an adversarial input. For the sparse case, it is an interesting question how to use a required optimization function (in a general manner) to get more levarage in finding a valid clustering.

References

[1] Dimitris Achlioptas and Frank McSherry, *On spectral learning of mixtures of distributions*, Conference on Learning Theory (COLT) 2005, 458-469.

[2] Noga Alon, *Eigenvalues and expanders*, Combinatorica, **6(2)**, (1986) , 83-96.

[3] Noga Alon and Nabil Kahale, *A spectral technique for coloring random 3-colorable graphs*, SIAM Journal on Computing **26** (1997), n. 6. 1733-1748.

[4] Yossi Azar, Amos Fiat, Anna R. Karlin, Frank McSherry and Jared Saia, *Spectral analysis of data*, Proceedings of the 32^{nd} annual ACM Symposium on Theory of computing (2001), 619-626.

[5] Ravi Boppana, *Eigenvalues and graph bisection: an average case analysis*, Proceedings of the 28^{th} IEEE Symposium on Foundations of Computer Science (1987).

[6] Thang Bui, Soma Chaudhuri, Tom Leighton and Mike Sipser, *Graph bisection algorithms with good average case behavior*, Combinatorica, **7**, 1987, 171-191.

[7] David Cheng, Ravi Kannan, Santosh Vempala and Grant Wang, *A Divide-and- Merge methodology for Clustering*, Proc. of the 24^th ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems (PODS), 196 - 205.

[8] Amin Coja-Oghlan, *A spectral heuristic for bisecting random graphs*, Proceedings of the 16^th Annual ACM-SIAM Symposium on Discrete Algorithms, 2005.

[9] Amin Coja-Oghlan, *An adaptive spectral heuristic for partitioning random graphs*, Automata, Languages and Programming, 33^{rd} International Colloquium, ICALP, Lecture Notes in Computer Science 4051 Springer 2006.

[10] Anirban Dasgupta, John Hopcroft and Frank McSherry, *Spectral analysis of random Graphs with skewed degree distributions*, Proceedings of the 42^{nd} IEEE Symposium on Foundations of Computer Science (2004), 602-610.

[11] Martin Dyer and Alan Frieze, *Fast Solution of Some Random NP-Hard Problems*, Proceedings of the 27^{th} IEEE Symposium on Foundations of Computer Science (1986), 331-336

[12] Uriel Feige and Eran Ofek, *Spectral techniques applied to sparse random graphs*, Random Structures and Algorithms, **27(2)**, 251–275, September 2005.

[13] M Fiedler, *Algebraic connectibility of graphs*, Czechoslovak Mathematical Journal, **23(98)**, (1973), 298-305.

[14] Joel Friedman, Jeff Kahn and Endre Szemeredi, *On the second eigenvalue of random regular graphs*, Proceedings of the 21^{st} annual ACM Symposium on Theory of computing (1989), 587 - 598.

[15] Zoltan Furedi and Janos Komlos, *The eigenvalues of random symmetric matrices*, Combinatorica 1, **3**, (1981), 233–241.

[16] G. Golub, C. Van Loan (1996), *Matrix computations, third edition*, *The Johns Hopkins University Press Ltd., London*.

[17] Ravi Kannan, Santosh Vempala and Adrian Vetta, *On Clusterings : Good, bad and spectral*, Proceedings of the Symposium on Foundations of Computer Science (2000), 497 - 515.

[18] Frank McSherry, *Spectral partitioning of random graphs*, Proceedings of the 42^{nd} IEEE Symposium on Foundations of Computer Science (2001), 529-537.

[19] Alistair Sinclair and Mark Jerrum, *Conductance and the mixing property of markov chains*, the approximation of the permenant resolved, Proc. of the 20^{th} annual ACM Symposium on Theory of computing (1988), 235-244.

[20] Daniel Spielman and Shang-hua Teng, *Spectral Partitioning Works: Planar graphs and finite element meshes*, Proc. of the 37th Annual Symposium on Foundations of Computer Science (FOCS '96), 96 - 105.

[21] Deepak Verma and Marina Meila, *A comparison of spectral clustering algorithms*, TR UW-CSE-03-05-01, Department of Computer Science and Engineering, University of Washington (2005).

[22] Van Vu, *Spectral norm of random matrices*, Proc. of the 36^{th} annual ACM Symposium on Theory of computing (2005), 619-626.

[23] Ying Zhao and George Karypis, *Evaluation of hierarchical clustering algorithms for document datasets*, Proc. of the 11 International Conference on Information and Knowledge Management (2002), 515 - 524.