

**From the holdings of the Department of Special Collections , Princeton University Library,  
Princeton, NJ**

These documents are either in the public domain, or can only be used for educational and research purposes (Fair use) as per U.S. Copyright law (text below). By accessing this file, all users agree that their use falls in the public domain or within fair use as defined by the copyright law. We do not charge any permission or usage fees for the publication of images of material in our collections, including those provided by Princeton University Library, via our website, catalog, or directly from staff. For more information regarding copyright, credit, and citation please review the information on the Copyright, Credit and Citation Guidelines page of our website.

<https://library.princeton.edu/special-collections/policies/copyright-credit-and-citation-guidelines>. If you have any questions, including how to order higher-quality images from the collection, contact Special Collections Public Services via our Ask Us! Form. <https://library.princeton.edu/special-collections/ask-us>

---

Doctoral Dissertations, 2011-2026

Mathematics

---

2017

# Community Detection in the Stochastic Block Model: fundamental limits

Sandon, Colin Peter

Princeton, NJ : Princeton University

---

<http://arks.princeton.edu/ark:/88435/dsp011v53k058n>

*U.S. Copyright law: The copyright law of the United States (Title 17, §108, United States Code) governs the making of photocopies or other reproductions of copyrighted material. Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be used for any purpose other than private study, scholarship, or research. If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of fair use that user may be liable for copyright infringement. This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve a violation of copyright law. <https://www.copyright.gov/title17/title17.pdf>*

# COMMUNITY DETECTION IN THE STOCHASTIC BLOCK MODEL: FUNDAMENTAL LIMITS

COLIN SANDON

A DISSERTATION  
PRESENTED TO THE FACULTY  
OF PRINCETON UNIVERSITY  
IN CANDIDACY FOR THE DEGREE  
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE  
BY THE DEPARTMENT OF  
MATHEMATICS  
ADVISER: EMMANUEL ABBE

JUNE 2017

© Copyright by Colin Sandon, 2017.

All Rights Reserved

# Abstract

In recent years, there have been many developments in the study of the stochastic block model, starting with a paper by Decelle et al. in which they conjecture that in the sparse stochastic block model, efficient community detection is possible if and only if the parameters are above the Kesten-Stigum bound. Massoulié, Mossel et al., and Bordenave et al. succeeded in proving the positive half of this conjecture in the 2-community symmetric case. Also, Mossel et al. proved that community detection is impossible below the KS bound in the 2-community symmetric case. On another note, Abbe et al. and Mossel et al. found a threshold for when it was possible to completely recover the communities in the 2-community symmetric case.

In this thesis, we go beyond the 2-community symmetric case to prove the positive half of the conjecture for the general SBM. In order to do that, we construct a linearized belief propagation algorithm that runs in  $O(n \log n)$  time and prove that it detects communities whenever the parameters of the SBM are above the KS bound. We also prove that unlike in the 2-community symmetric case, in the general SBM there are choices of parameters below the KS threshold for which one can detect communities in exponential time, confirming another conjecture by Decelle et al. This result implies that we cannot prove the impossibility of efficient community detection below the KS bound without proving some major conjectures on what computations can be done efficiently. We also extend the threshold for when it is possible to completely recover the communities to the general SBM.

## Acknowledgements

I would like to thank Emmanuel Abbe for his support. This thesis would not exist without him.

I dedicated this thesis to Bob Chaffee in recognition of his role in my early math education.

# Contents

|                                                                                        |           |
|----------------------------------------------------------------------------------------|-----------|
| Abstract . . . . .                                                                     | iii       |
| Acknowledgements . . . . .                                                             | iv        |
| <b>1 Introduction</b>                                                                  | <b>1</b>  |
| 1.1 Community detection . . . . .                                                      | 1         |
| 1.2 The stochastic block model . . . . .                                               | 1         |
| 1.3 Our results . . . . .                                                              | 1         |
| 1.4 Related work . . . . .                                                             | 2         |
| 1.5 Outline . . . . .                                                                  | 3         |
| <b>2 Results</b>                                                                       | <b>4</b>  |
| 2.1 Terminology . . . . .                                                              | 4         |
| 2.2 Weak recovery, ABP, and TSP . . . . .                                              | 6         |
| 2.2.1 The acyclic belief propagation algorithm . . . . .                               | 7         |
| 2.2.2 Information-theoretic detection, and the typicality sampling algorithm . . . . . | 11        |
| 2.3 Strong recovery . . . . .                                                          | 11        |
| 2.4 Exact recovery and the degree profiling algorithm . . . . .                        | 17        |
| <b>3 The typicality sampling algorithm</b>                                             | <b>20</b> |
| <b>4 Acyclic belief propagation</b>                                                    | <b>23</b> |
| 4.1 ABP intuition . . . . .                                                            | 23        |
| 4.1.1 Amplifying luck . . . . .                                                        | 24        |
| 4.1.2 Nonbacktracking walks . . . . .                                                  | 26        |
| 4.1.3 Compensation for the average value . . . . .                                     | 29        |
| 4.1.4 Vanilla ABP . . . . .                                                            | 31        |
| 4.1.5 ABP for the symmetric SBM . . . . .                                              | 33        |

|          |                                                     |            |
|----------|-----------------------------------------------------|------------|
| 4.1.6    | Spectral view of ABP . . . . .                      | 36         |
| 4.1.7    | ABP for the general SBM . . . . .                   | 38         |
| 4.1.8    | Alternatives . . . . .                              | 41         |
| 4.2      | ABP proof technique . . . . .                       | 42         |
| <b>5</b> | <b>The degree profiling algorithm</b>               | <b>44</b>  |
| <b>6</b> | <b>Proofs</b>                                       | <b>46</b>  |
| 6.1      | Information-theoretic community detection . . . . . | 46         |
| 6.1.1    | Atypicality of a bad clustering . . . . .           | 47         |
| 6.1.2    | Size of the typical set . . . . .                   | 52         |
| 6.1.3    | Sampling estimates . . . . .                        | 59         |
| 6.2      | Achieving the KS threshold . . . . .                | 61         |
| 6.2.1    | Preliminaries . . . . .                             | 66         |
| 6.2.2    | The shard decomposition . . . . .                   | 68         |
| 6.2.3    | Bounding the variance of $W_{m/S}$ . . . . .        | 75         |
| 6.3      | Degree profiling and exact recovery . . . . .       | 90         |
| 6.3.1    | Formal results . . . . .                            | 91         |
| 6.3.2    | Testing degree profiles . . . . .                   | 93         |
| 6.3.3    | Proof of Theorem 11 . . . . .                       | 99         |
| <b>7</b> | <b>Simulations and code</b>                         | <b>102</b> |

# Chapter 1

## Introduction

### 1.1 Community detection

In many situations one wishes to analyze a collection of entities with connections between them that can be represented by a graph. Frequently, these entities have difficult to observe characteristics that influence which edges exist. That suggests that one could deduce information about these characteristics from the graph. More precisely, it suggests that the graph contains multiple communities of vertices that behave similarly, and that being able to determine which community a vertex is in could be useful.

### 1.2 The stochastic block model

A simple model of random graphs with communities is the stochastic block model. In it, each vertex is independently assigned a community according to a fixed probability distribution, and every pair of vertices have an edge between them with a probability dependent on their communities.

### 1.3 Our results

It was previously conjectured that in the stochastic block model, it is possible to efficiently detect communities if and only if the parameters of the model are above the Kesten-Stigum bound.[12] In this thesis, we prove one half of this conjecture by giving the Acyclic Belief Propagation algorithm and proving that it detects communities in  $O(n \log(n))$  time whenever the parameters are above the KS threshold. We also show that there are choices of parameters below the KS threshold

in which community detection is information-theoretically possible by giving an exponential time algorithm, the Typicality Sampling Algorithm, which detects communities on these models. The TSP algorithm can be carried out using memory polynomial in  $n$ , so if the conjecture that efficient detection is only possible above the KS bound is true, then there exists a computation that can be performed in polynomial memory, but not in polynomial time. In other words, the conjecture implies that  $P \neq PSPACE$ . That is believed to be true, but no one knows how to prove it, so it is unlikely that anyone will prove that efficient detection is impossible below the threshold in the near future. We also prove that one can determine communities with accuracy  $1 - o(1)$  when the degrees of vertices in the graph increases with  $n$ , and that it is possible to recover the communities exactly if and only if the parameters of the model satisfy a criterion we will give.

## 1.4 Related work

Decelle et al. conjectured based on tests that they had run and ideas from statistical physics that in the sparse stochastic block model, there is an efficient algorithm for partial recovery of communities if and only if the parameters were above the KS bound [12]. They also conjectured that for any  $k \geq 4$ , there were parameters for the  $k$ -community symmetric SBM that were below the KS bound for which partial recovery was possible in exponential time, while for  $k \leq 3$  detection was impossible below the KS threshold in the  $k$ -community symmetric SBM. Most of the other recent work has attempted to prove or build off this conjecture.

Mossel, Neeman, and Sly proved that it was impossible to detect communities below the KS bound in the 2-community symmetric SBM [16], and gave an algorithm based on nonbacktracking walks that detects communities in the 2-community symmetric SBM whenever the parameters are above the KS bound [18]. Massoulié gave an alternate method of detecting communities in the 2-community symmetric SBM above the KS bound that is based on the second eigenvector of a matrix listing the number of paths of appropriate length between each pair of vertices in the graph [15]. Mossel, Neeman, and Sly also proved that one can find communities with optimal accuracy when the signal to noise ratio is sufficiently high in the 2-community symmetric SBM by using the output of a detection algorithm as a starting point for an appropriate version of belief propagation [17].

Abbe, Bandeira, and Hall[1], as well as Mossel, Neeman, and Sly[19] found a threshold for when exact recovery is possible in the 2-community symmetric SBM. Mossel, Neeman, and Sly[19] also give a tight condition for when strong recovery is possible in this case. Both papers give an algorithm for

recovering communities that starts by using a fairly inaccurate algorithm and then uses information the preliminary classification provides on each vertex’s neighbors to classify it more accurately. They also both prove impossibility results by showing that with those parameters there are likely to be vertices that appear more plausible as members of the opposite community than as members of their actual community. In the general SBM, Bordenave, Lelarge, and Massouli proved that if the parameters are above the KS threshold and a specified matrix has a second eigenvalue of order 1 then the second eigenvector of the graph’s nonbacktracking walk matrix is correlated with the graph’s communities [10]. This covers the papers that are most directly related to this thesis. There are many more papers on the SBM, and a more detailed review of the literature can be found in [6].

## 1.5 Outline

Chapter 2 of this paper starts by formally defining the stochastic block model, community detection, and community recovery. It then gives our typicality sampling algorithm from [5] and [7], our acyclic belief propagation algorithm from [4], and our degree profiling algorithm from [2], along with our theorems concerning their effectiveness. It also gives our theorem from [2] that exact recovery is impossible when the conditions for degree profiling to work are not met. Chapter 3 explains why the TSP algorithm should detect communities if any possible algorithm will, as well as briefly explaining how we will prove that it works for some choices of parameters. Chapter 4 provides intuition for why the ABP algorithm is the way it is, as well as briefly explaining the techniques used to prove that it works. Chapter 5 provides intuition for why the threshold for exact recovery is where it is, as well as summarizing how we will prove that the degree profiling algorithm succeeds at exact recovery whenever it is possible. Chapter 6 formally proves all of our theorems. Chapter 7 contains code for a variant of ABP and data on how well the program performs on test graphs.

# Chapter 2

## Results

### 2.1 Terminology

**Definition 1.** For positive integers  $k, n$ , a probability distribution  $p \in (0, 1)^k$ , and a  $k \times k$  symmetric matrix  $W$  with entries in  $[0, 1]$ , we define  $\text{SBM}(n, p, W)$  as the probability distribution over ordered pairs  $(\sigma, G)$  of an assignment of vertices to one of  $k$  communities and an  $n$ -vertex graph generated by the following procedure. First, each vertex  $v \in V(G)$  is independently assigned a community  $\sigma_v$  under the probability distribution  $p$ . Then, for every  $v \neq v'$ , an edge is drawn in  $G$  between  $v$  and  $v'$  with probability  $W_{\sigma_v, \sigma_{v'}}$ , independently of other edges. We define  $\Omega_i = \{v : \sigma_v = i\}$ . Also, let  $P$  be the  $k \times k$  diagonal matrix such that  $P_{i,i} = p_i$  for each  $i$ .

We call  $\Omega_1, \dots, \Omega_k$  the communities of the resulting graph. Our goal is to determine how accurately one can find the communities of a graph drawn from  $\text{SBM}(n, p, s(n)Q)$  in the asymptotic limit as  $n \rightarrow \infty$  where  $k$ ,  $p$ , and  $Q$  stay constant while  $s(n)$  is a decreasing function of  $n$ . There are a couple of ways to measure the accuracy with which one can find a community. However, before we get to that, we need to consider what it means to find communities with nontrivial accuracy. One formalization of that is the following.

**Definition 2.** An algorithm  $\hat{\sigma} : 2^{\binom{[n]}{2}} \rightarrow [k]^n$  solves max-detection in  $\text{SBM}(n, p, Q)$  if there exists  $\epsilon > 0$  such that

$$\mathbb{P}\{A(\sigma, \hat{\sigma}) \geq \max p_i + \epsilon\} = 1 - o(1), \quad (2.1)$$

where  $(\sigma, G) \sim \text{SBM}(n, p, Q)$  and for  $x, y \in [k]^n$ ,  $A(x, y) = \max_{\pi \in S_k} \frac{1}{n} \sum_{i=1}^n \mathbb{1}(x_i = \pi(y_i))$  denotes

the agreement<sup>1</sup> between  $x$  and  $y$ .

max-detection was the original definition that Decelle et al. used when conjecturing that partial recovery of communities is possible iff the parameters of the model are above the KS threshold. However, it turns out that there are parameters of the SBM above the KS threshold on which max-detection is impossible. The key issue is that in order to solve max-detection an algorithm must sometimes find enough evidence about a vertex's community to overcome the prior probability that it is in the largest community, and this is not always possible, such as in the following case.

**Example 1.** *Consider a 2-community SBM in which each vertex is assigned to community 1 with probability .99 and community 2 with probability .01. Each pair of vertices in community 1 have an edge between them with probability  $2/n$ , while vertices in community 2 never have edges. In a graph drawn from this SBM, every vertex of nonzero degree must be from community 1, while a vertex of degree 0 is still approximately 93% likely to be from community 1. As such, the way to classify the vertices in order to get as many right as possible is to guess that they all came from the largest community and max-detection is impossible. However, we can divide the vertices into a set of vertices that are definitely from community 1, and a set of vertices that have an elevated probability of being from community 2. So, we clearly have nontrivial amounts of information on which vertices are from community 2.*

That suggests that perhaps we should say that an algorithm finds communities with nontrivial accuracy if it can select a subset of a graph's vertices that is disproportionately drawn from certain communities. In other words, we could use the following definition.

**Definition 3.** *Let  $A$  be an algorithm that takes a graph as input and outputs a subset of its vertices.  $A$  separates communities in graphs drawn from  $\text{SBM}(n, p, W)$  if there exists  $\epsilon > 0$  such that the following holds. When  $(\sigma, G)$  is drawn from  $\text{SBM}(n, p, Q/n)$  and  $A$  outputs a subset  $S$  of its vertices, with probability  $1 - o(1)$ , there exist  $i \in [k]$  such that  $|\Omega_i \cap S|/|\Omega_i| - |S|/n > \epsilon$ . Community separation is solvable efficiently if the algorithm runs in polynomial time in  $n$ , and information-theoretically if such an algorithm exists at all.*

Alternately, one could have an algorithm assign every vertex in the graph to a community, in which case you could define its accuracy as follows.

**Definition 4.** *An algorithm  $\hat{\sigma} : 2^{\binom{[n]}{2}} \rightarrow [k]^n$  recovers communities with accuracy  $f(n)$  in  $\text{SBM}(n, p, Q)$*

---

<sup>1</sup>Permutations  $\pi$  need to be considered since only the partition needs to be detected and not the actual labels.

if

$$\mathbb{P} \left\{ \max_{\pi \in S_k} \frac{1}{k} \sum_{i=1}^k \frac{|\{v \in \Omega_i : \pi(\hat{\sigma}_v) = i\}|}{|\Omega_i|} \geq f(n) \right\} = 1 - o(1), \quad (2.2)$$

where  $(\sigma, G) \sim \text{SBM}(n, p, Q)$ . We say that an algorithm solves weak recovery if it recovers communities with accuracy  $1/k + \Omega(1)$ , that an algorithm solves strong recovery if it recovers communities with accuracy  $1 - o(1)$ , and that it solves exact recovery if it recovers communities with accuracy 1.

In other words, an algorithm that assigns each vertex to a community has an accuracy equal to the average, over all communities, of the fraction of the vertices from that community that it classified correctly. Note that an algorithm that assigns vertices to communities in a way that is independent of their actual communities will always have an accuracy of  $1/k$ , while an algorithm that assigns every vertex to the correct community will have accuracy 1.

Also, any algorithm that solves max-detection also solves weak recovery, while an algorithm that separates communities can be converted to an algorithm that solves weak recovery by assigning all of the vertices in the set that it outputs to one community, all other vertices to another community, and not assigning any vertices to the remaining communities. An algorithm that solves weak recovery can usually be converted to an algorithm that solves max-detection or an algorithm that separates communities, but there are exceptions. Throughout the rest of this paper, we will mostly be focusing on the question of whether or not there is an algorithm that recovers communities with some given level of accuracy, although we will sometimes construct algorithms that separate communities in order to prove that weak recovery is possible.

## 2.2 Weak recovery, ABP, and TSP

If  $s(n) = o(1)$ , then most of the vertices in a graph drawn from  $\text{SBM}(n, p, s(n)Q)$  will have degree 0, and there will be no way to determine what communities they are from. So, weak recovery is clearly impossible. Thus,  $\text{SBM}(n, p, Q/n)$  is the lowest degree nontrivial case. Strong recovery is still clearly impossible because a constant fraction of the vertices in every community will have degree 0, so in this case we focus on whether or not weak recovery is possible.

If vertices in different communities have different expected degrees, then we can separate communities simply by picking out the high degree vertices. So, for the rest of this section we will assume that the parameters of the SBM are such that vertices in every community have the same expected degree. Now, let  $\lambda_1, \dots, \lambda_h$  be the distinct eigenvalues of  $PQ$  in order of nonincreasing magnitude.

Our results are in terms of the following notion of SNR:

**Definition 5.** *The signal to noise ratio of  $SBM(n, p, Q/n)$  is defined as*

$$SNR = \lambda_2^2 / \lambda_1.$$

*or as  $\lambda_1$  if it has multiplicity greater than 1.*

This paper shows that efficient weak recovery is possible if  $SNR > 1$ , and gives an efficient algorithm that can detect communities in this case. More precisely, we prove that

**Theorem 1.** *Let  $p \in (0, 1)^k$  with  $\sum p = 1$ ,  $Q$  be a symmetric matrix with nonnegative entries,  $P$  be the diagonal matrix such that  $P_{i,i} = p_i$ , and  $\lambda_1, \dots, \lambda_h$  be the distinct eigenvalues of  $PQ$  in order of nonincreasing magnitude. If  $\lambda_2^2 > \lambda_1$  then there exist constants  $r, c$  and  $m = \Theta(\log(n))$  such that the acyclic belief propagation algorithm with these parameters separates communities in  $SBM(n, p, Q/n)$ . The algorithm can be run in  $O(n \log n)$  time.*

The proof is in Section 6.2. On the flip side, we believe that when the  $SNR < 1$ , there is no efficient algorithm that solves weak recovery, as conjectured in [12]. More formally, we believe that the following holds:

**Conjecture 1.** *Let  $p \in (0, 1)^k$  with  $\sum p = 1$ ,  $Q$  be a symmetric matrix with nonnegative entries,  $P$  be the diagonal matrix such that  $P_{i,i} = p_i$ , and  $\lambda_1, \dots, \lambda_h$  be the distinct eigenvalues of  $PQ$  in order of nonincreasing magnitude. If every column of  $PQ$  has the same sum,  $\lambda_2^2 \leq \lambda_1$ , and either  $\lambda_1$  has multiplicity 1 or  $\lambda_1 \leq 1$  then no polynomial time algorithm solves weak recovery on graphs drawn from  $SBM(n, p, Q/n)$ .*

The conjectural boundary at  $SNR = 1$  between parameters for which efficient weak recovery is possible and parameters for which it is not is referred to as the Kesten-Stigum bound.

### 2.2.1 The acyclic belief propagation algorithm

Here we present two versions of our main algorithm. A simplified version ABP\* that can easily be implemented, and the general version ABP that is used to prove Theorem 1. The general version has extra steps that are necessary for the proof that the algorithm works, but probably unnecessary in practice. The intuition behind the algorithm is discussed in chapter 4. The algorithms below have a message passing implementation and correspond to linearized version of belief propagation in which we originally guess what communities each vertex is likely to be in and then determine what

communities each vertex's neighbors provide evidence for it to be in. Then we use that information to update our beliefs about what evidence each vertex provides about its neighbors' communities and repeat while compensating for the feedback caused by any small cycles in the graph.

ABP<sup>\*</sup>( $G, m, r$ ):

1. For each adjacent  $v$  and  $v'$  in  $G$ , randomly draw  $y_{v,v'}^{(1)}$  from a Gaussian distribution with mean 0 and variance 1. Also, consider  $y_{v,v'}^{(t)}$  as having a value of 0 whenever  $t < 1$ .
2. For each  $1 < t \leq m$ , set

$$z_{v,v'}^{(t-1)} = y_{v,v'}^{(t-1)} - \frac{1}{2|E(G)|} \sum_{(v'',v''') \in E(G)} y_{v'',v'''}^{(t-1)} \quad (2.3)$$

for all adjacent  $v$  and  $v'$ . For each adjacent  $v, v'$  in  $G$  that are not part of a cycle of length  $r$  or less, set

$$y_{v,v'}^{(t)} = \sum_{v'' : (v', v'') \in E(G), v'' \neq v} z_{v', v''}^{(t-1)},$$

and for the other adjacent  $v, v'$  in  $G$ , let the other vertex in the cycle that is adjacent to  $v$  be  $v'''$ , the length of the cycle be  $r'$ , and set

$$y_{v,v'}^{(t)} = \sum_{v'' : (v', v'') \in E(G), v'' \neq v} z_{v', v''}^{(t-1)} - \sum_{v'' : (v, v'') \in E(G), v'' \neq v', v'' \neq v'''} z_{v, v''}^{(t-r')}$$

unless  $t = r'$ , in which case, set  $y_{v,v'}^{(t)} = \sum_{v'' : (v', v'') \in E(G), v'' \neq v} z_{v', v''}^{(t-1)} - z_{v''', v}^{(1)}$ .

3. Set  $y'_v = \sum_{v' : (v', v) \in E(G)} y_{v,v'}^{(m)}$  for every  $v \in G$ . Return  $\{v : y'_v > 0\}$ .

**Remarks.**

- (1) In the  $r = 2$  case, one does not need to find cycles and one can exit step 2. after the second formula. As mentioned above, we rely on a less compact version of the algorithm to prove the theorem, but expect that the above also succeeds at weak recovery as long as  $m > 2 \ln(n) / \ln(\text{SNR}) + \omega(1)$ .
- (2) What the algorithm does if  $(v, v')$  is in multiple cycles of length  $r$  or less is unspecified above, as there is no such edge with probability  $1 - o(1)$  in the sparse SBM. This can be modified for more general settings. The simplest such modification would be to apply this adjustment independently for each such cycle, and thus to set

$$y_{v,v'}^{(t)} = \sum_{v'' : (v', v'') \in E(G), v'' \neq v} z_{v', v''}^{(t-1)} - \sum_{r'=1}^r \sum_{v''' : (v, v''') \in E(G)} C_{v''', v, v'}^{(r')} \sum_{v'' : (v, v'') \in E(G), v'' \neq v', v'' \neq v'''} z_{v, v''}^{(t-r')},$$

where  $C_{v''',v,v'}^{(r')}$  denotes the number of length  $r'$  cycles that contain  $v''', v, v'$  as consecutive vertices, substituting  $z_{v''',v}^{(1)}$  for  $\sum_{v'':(v,v'') \in E(G), v'' \neq v', v'' \neq v'''} z_{v,v''}^{(t-r')}$  when  $r' = t$ . This will not quite count  $r$ -nonbacktracking walks, but we believe that it would give a good enough approximation.

(3) The purpose of setting  $z_{v,v'}^{(t-1)} = y_{v,v'}^{(t-1)} - \frac{1}{2|E(G)|} \sum_{(v'',v''') \in E(G)} y_{v'',v'''}^{(t-1)}$  is to ensure that the average value of the  $y^{(t)}$  is approximately 0, and thus that the eventual division of the vertices into two sets is roughly even. There is an alternate way of doing this in which we simply let  $z_{v,v'}^{(t-1)} = y_{v,v'}^{(t-1)}$  and then compensate for any bias of  $y^{(t)}$  towards positive or negative values at the end. More specifically, we define  $Y$  to be the  $n \times m$  matrix such that for all  $t$  and  $v$ ,  $Y_{v,t} = \sum_{v':(v',v) \in E(G)} y_{v,v'}^{(t)}$ , and  $M$  to be the  $m \times m$  matrix such that  $M_{i,i} = 1$  and  $M_{i,i+1} = -\lambda_1$  for all  $i$ , and all other entries of  $M$  are equal to 0. Then we set  $y' = Y M^{m'} e_m$ , where  $e_m \in \mathbb{R}^m$  denotes the unit vector with 1 in the  $m$ -th entry, and  $m'$  is a suitable integer.

The full version of the algorithm applying to the general SBM is as follows.

$ABP(G, m, r, c, (\lambda_1, \dots, \lambda_h)) :$

1. Initialize:

- (a) Set  $s = 2$  unless  $h > 2$  and  $|\lambda_2| = |\lambda_3|$ , in which case set  $s = 3$ .
- (b) Set  $\gamma = (1 - \lambda_1/\lambda_2^2)/2$ .
- (c) Set  $l = \max((s-1)/\ln((1-\gamma)|\lambda_s|) + s-1, 2(2r+1)(s-1))$ .
- (d) Assign each edge of  $G$  independently with probability  $\gamma$  to a set  $\Gamma$ . Then, remove these edges from  $G$ .
- (e) For every vertex  $v \in G$ , randomly draw  $x_v$  from a Gaussian distribution with mean 0 and variance 1.
- (f) For each adjacent  $v$  and  $v'$ , set  $y_{v,v'}^{(1)} = x_{v'}$ , and  $y_{v,v'}^{(t)} = 0$  for all  $t < 1$ .

2. Propagate:

- (a) For each  $1 \leq t \leq m$ , and each adjacent  $(v, v') \in E(G)$ , set

$$y_{v,v'}^{(t)} = \sum_{v'':(v',v'') \in E(G), v'' \neq v} y_{v',v''}^{(t-1)}$$

unless  $(v, v')$  is part of a cycle of length  $r$  or less. If it is, then let the other vertex in the cycle that is adjacent to  $v$  be  $v'''$ , and the length of the cycle be  $r'^2$ . Set

$$y_{v,v'}^{(t)} = \sum_{v'':(v',v'') \in E(G), v'' \neq v} y_{v',v''}^{(t-1)} - \sum_{v'':(v,v'') \in E(G), v'' \neq v', v'' \neq v'''} y_{v,v''}^{(t-r')}$$

unless  $t = r'$ . In that case, set

$$y_{v,v'}^{(t)} = \sum_{v'':(v',v'') \in E(G), v'' \neq v} y_{v',v''}^{(t-1)} - x_v.$$

(b) Set  $Y$  to be the  $n \times m$  matrix such that for all  $t$  and  $v$ ,

$$Y_{v,t} = \sum_{v':(v',v) \in E(G)} y_{v,v'}^{(t)}$$

(c) For each  $s' < s$ , set  $M_{s'}$  to be the  $m \times m$  matrix such that  $M_{i,i} = 1$  and  $M_{i,i+1} = -(1-\gamma)\lambda_{s'}$  for all  $i$ , and all other entries of  $M_{s'}$  are equal to 0. Also, let  $e_m \in \mathbb{R}^m$  be the vector with an  $m$ -th entry of 1 and all other entries equal to 0. Set

$$y^{(m)} = Y \left( \prod_{s' < s} M_{s'}^{\lceil \frac{m-r-(2r+1)s'}{t} \rceil} \right) e_m.$$

(d) For each  $v$ , set

$$y'_v = \sum_{v':(v,v') \in \Gamma} y_{v,v'}^{(m)}$$

and set  $y''_v$  to the sum of  $y'_{v'}$  over all  $v'$  that have shortest paths to  $v$  of length  $\lfloor \sqrt{\log \log n} \rfloor$ .

3. Assign:

(a) Set  $c' = c \cdot \sqrt{\sum_{v \in G} (y''_v)^2 / n}$ . Create sets of vertices  $S_1$  and  $S_2$  as follows. For each vertex  $v$ , if  $y''_v < -c'$ , assign  $v$  to  $S_1$ . If  $y''_v > c'$ , then assign  $v$  to  $S_2$ . Otherwise, assign  $v$  to  $S_2$  with probability  $1/2 + y''_v/2c'$  and  $S_1$  otherwise.

(b) Return  $S_2$ .

---

<sup>2</sup>What the algorithm does if  $(v, v')$  is in multiple cycles of length  $r$  or less is unspecified as there is no such edge with probability  $1 - o(1)$  in the SBM. One can adapt this for more general models.

### 2.2.2 Information-theoretic detection, and the typicality sampling algorithm

The previous theorem and conjecture cover when we think efficient weak recovery is possible on graphs drawn from  $\text{SBM}(n, p, Q/n)$ ; that leaves the topic of when weak recovery is information-theoretically possible on  $\text{SBM}(n, p, Q/n)$ . Our main result there is simply that it is sometimes information-theoretically possible when we do not think there is an efficient algorithm that solves weak recovery. More formally,

**Theorem 2.** *There exists  $p \in (0, 1)^k$  and a symmetric matrix  $Q$  satisfying the conditions of conjecture 1 such that the typicality sampling algorithm solves weak recovery on graphs drawn from  $\text{SBM}(n, p, Q/n)$ .*

**Definition 6.** *Given a graph  $G$  drawn from  $\text{SBM}(n, p, Q/n)$ , an assignment of its vertices to communities  $\sigma'$ , and  $\delta > 0$ ,  $\sigma'$  is  $\delta$ -typical if for every community  $i$ ,  $|\{v : \sigma'_v = i\}|/n \in [p_i - \delta, p_i + \delta]$  and for every pair of communities  $i, j$ , the number of edges between vertices in  $\{v : \sigma'_v = i\}$  and vertices in  $\{v : \sigma'_v = j\}$  is between  $(1 - \delta)$  times the expected number of edges between vertices in those communities and  $(1 + \delta)$  times the expected number of edges between vertices in those communities. We also let  $T_\delta(G)$  be the set of all  $\delta$ -typical classifications of  $G$ 's vertices.*

In other words, an assignment of  $G$ 's vertices to communities is typical if it assigns approximately the right number of vertices to each community and it has approximately the right number of edges between each pair of alleged communities. The typicality sampling algorithm simply returns a random element of  $T_\delta(G)$  for appropriate  $\delta$ .

## 2.3 Strong recovery

The obvious approach to solving strong recovery in a graph drawn from  $\text{SBM}(n, p, f(n))$  when  $f(n) = \omega(1)$  would be to try to use a weak recovery algorithm in the expectation that as the degree increases its accuracy will also increase, or possibly to split the graph into  $\omega(1)$  graphs with expected degree 1, run a weak recovery algorithm on each of them, and then combine the results. However, both of these tactics require a weak recovery algorithm that distinguishes between every pair of communities with a nontrivial advantage, and ABP does not do that. So, we will use our approach from [2] instead.

The first key observation used to classify graphs' vertices in this case is that if  $v$  is a vertex in a

graph drawn from  $\text{SBM}(n, p, Q/n)$  then for all small  $r$  the expected number of vertices in community  $i$  that are  $r$  edges away from  $v$  is approximately  $e_i \cdot (PQ)^r e_{\sigma_v}$ . So, we define:

**Definition 7.** For any vertex  $v$ , let  $N_r(v)$  be the set of all vertices with shortest path to  $v$  of length  $r$ . If there are multiple graphs that  $v$  could be considered a vertex in, let  $N_{r[G]}(v)$  be the set of all vertices with shortest paths in  $G$  to  $v$  of length  $r$ .

We also refer to the vector with  $i$ -th entry equal to the number of vertices in  $N_r(v)$  that are in community  $i$  as  $N_r(v)$ . One could determine  $e_{\sigma_v}$  given  $(PQ)^r e_{\sigma_v}$ , but using  $N_r(v)$  to approximate that would require knowing how many of the vertices in  $N_r(v)$  are in each community. So, we attempt to get information relating to how many vertices in  $N_r(v)$  are in each community by checking how it connects to  $N_{r'}(v')$  for some vertex  $v'$  and integer  $r'$ . The obvious way to do this would be to compute the cardinality of their intersection. Unfortunately, whether a given vertex in community  $i$  is in  $N_r(v)$  is not independent of whether it is in  $N_{r'}(v')$ , which causes the cardinality of  $|N_r(v) \cap N_{r'}(v')|$  to differ from what one would expect badly enough to disrupt plans to use it for approximations.

In order to get around this, we randomly assign every edge in  $G$  to a set  $E$  with probability  $c$ . We hence define the following.

**Definition 8.** For any vertices  $v, v' \in G$ ,  $r, r' \in \mathbb{Z}$ , and subset of  $G$ 's edges  $E$ , let  $N_{r,r'[E]}(v \cdot v')$  be the number of pairs of vertices  $(v_1, v_2)$  such that  $v_1 \in N_{r[G \setminus E]}(v)$ ,  $v_2 \in N_{r'[G \setminus E]}(v')$ , and  $(v_1, v_2) \in E$ .

Note that  $E$  and  $G \setminus E$  are disjoint; however,  $G$  is sparse enough that even if they were generated independently a given pair of vertices would have an edge between them in both with probability  $O(\frac{1}{n^2})$ . So,  $E$  is approximately independent of  $G \setminus E$ . Thus, for any  $v_1 \in N_{r[G \setminus E]}(v)$  and  $v_2 \in N_{r'[G \setminus E]}(v')$ ,  $(v_1, v_2) \in E$  with a probability of approximately  $cQ_{\sigma_{v_1}, \sigma_{v_2}}/n$ . As a result,

$$\begin{aligned} N_{r,r'[E]}(v \cdot v') &\approx N_{r[G \setminus E]}(v) \cdot \frac{cQ}{n} N_{r'[G \setminus E]}(v') \\ &\approx ((1-c)PQ)^r e_{\sigma_v} \cdot \frac{cQ}{n} ((1-c)PQ)^{r'} e_{\sigma_{v'}} \\ &= c(1-c)^{r+r'} e_{\sigma_v} \cdot Q(PQ)^{r+r'} e_{\sigma_{v'}}/n \end{aligned}$$

Recall that  $\lambda_1, \dots, \lambda_h$  are the distinct eigenvalues of  $PQ$ , ordered so that  $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_h| \geq 0$ . Also define  $h'$  so that  $h' = h$  if  $\lambda_h \neq 0$  and  $h' = h - 1$  if  $\lambda_h = 0$ . If  $W_i$  is the eigenspace of  $PQ$

corresponding to the eigenvalue  $\lambda_i$ , and  $P_{W_i}$  is the projection operator on to  $W_i$ , then

$$\begin{aligned}
N_{r,r'}[E](v \cdot v') &\approx c(1-c)^{r+r'} e_{\sigma_v} \cdot Q(PQ)^{r+r'} e_{\sigma_{v'}} / n \\
&= \frac{c(1-c)^{r+r'}}{n} \left( \sum_i P_{W_i}(e_{\sigma_v}) \right) \cdot Q(PQ)^{r+r'} \left( \sum_j P_{W_j}(e_{\sigma_{v'}}) \right) \\
&= \frac{c(1-c)^{r+r'}}{n} \sum_{i,j} P_{W_i}(e_{\sigma_v}) \cdot Q(PQ)^{r+r'} P_{W_j}(e_{\sigma_{v'}}) \\
&= \frac{c(1-c)^{r+r'}}{n} \sum_{i,j} P_{W_i}(e_{\sigma_v}) \cdot P^{-1}(\lambda_j)^{r+r'+1} P_{W_j}(e_{\sigma_{v'}}) \\
&= \frac{c(1-c)^{r+r'}}{n} \sum_i \lambda_i^{r+r'+1} P_{W_i}(e_{\sigma_v}) \cdot P^{-1} P_{W_i}(e_{\sigma_{v'}})
\end{aligned}$$

where the final equality holds because for all  $i \neq j$ ,

$$\begin{aligned}
\lambda_i P_{W_i}(e_{\sigma_v}) \cdot P^{-1} P_{W_j}(e_{\sigma_{v'}}) &= (PQ P_{W_i}(e_{\sigma_v})) \cdot P^{-1} P_{W_j}(e_{\sigma_{v'}}) \\
&= P_{W_i}(e_{\sigma_v}) \cdot Q P_{W_j}(e_{\sigma_{v'}}) \\
&= P_{W_i}(e_{\sigma_v}) \cdot P^{-1} \lambda_j P_{W_j}(e_{\sigma_{v'}}),
\end{aligned}$$

and since  $\lambda_i \neq \lambda_j$ , this implies that  $P_{W_i}(e_{\sigma_v}) \cdot P^{-1} P_{W_j}(e_{\sigma_{v'}}) = 0$ .

That implies that one can approximately solve for  $P_{W_i} e_{\sigma_v} \cdot P^{-1} P_{W_i} e_{\sigma_{v'}}$  given  $N_{r,r'+j}(v \cdot v')$  for all  $0 \leq j < h'$ . Of course, this requires  $r$  and  $r'$  to be large enough such that

$$\frac{c(1-c)^{r+r'}}{n} \lambda_i^{r+r'+1} P_{W_i}(e_{\sigma_v}) \cdot P^{-1} P_{W_i}(e_{\sigma_{v'}})$$

is large relative to the error terms for all  $i \leq h'$ . At a minimum, that requires that  $|(1-c)\lambda_i|^{r+r'+1} = \omega(n)$ , so

$$r + r' > \log(n) / \log((1-c)|\lambda_{h'}|).$$

On the flip side, one also needs

$$r, r' < \log(n) / \log((1-c)\lambda_1)$$

because otherwise the graph will start running out of vertices before one gets  $r$  steps away from  $v$  or  $r'$  steps away from  $v'$ .

Furthermore, for any  $v$  and  $v'$ ,

$$\begin{aligned} 0 &\leq P_{W_i}(e_{\sigma_v} - e_{\sigma_{v'}}) \cdot P^{-1}P_{W_i}(e_{\sigma_v} - e_{\sigma_{v'}}) \\ &= P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_v} - 2P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_{v'}} + P_{W_i}e_{\sigma_{v'}} \cdot P^{-1}P_{W_i}e_{\sigma_{v'}} \end{aligned}$$

with equality for all  $i$  if and only if  $\sigma_v = \sigma_{v'}$ , so sufficiently good approximations of  $P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_v}$ ,  $P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_{v'}}$  and  $P_{W_i}e_{\sigma_{v'}} \cdot P^{-1}P_{W_i}e_{\sigma_{v'}}$  can be used to determine which pairs of vertices are in the same community as follows.

**The Vertex comparison algorithm.** The inputs are  $(v, v', r, r', E, x, c)$ , where  $v, v'$  are two vertices,  $r, r'$  are positive integers,  $E$  is a subset of  $G$ 's edges,  $x$  is a positive real number, and  $c$  is a real number between 0 and 1.

The algorithm outputs a decision on whether  $v$  and  $v'$  are in the same community or not. It proceeds as follows.

(1) Solve the systems of equations:

$$\begin{aligned} \sum_i ((1-c)\lambda_i)^{r+r'+j+1} z_i(v \cdot v') &= \frac{(1-c)n}{c} N_{r+j, r'[E]}(v \cdot v') \text{ for } 0 \leq j < h' \\ \sum_i ((1-c)\lambda_i)^{r+r'+j+1} z_i(v \cdot v) &= \frac{(1-c)n}{c} N_{r+j, r'[E]}(v \cdot v) \text{ for } 0 \leq j < h' \\ \sum_i ((1-c)\lambda_i)^{r+r'+j+1} z_i(v' \cdot v') &= \frac{(1-c)n}{c} N_{r+j, r'[E]}(v' \cdot v') \text{ for } 0 \leq j < h' \end{aligned}$$

(2) If  $\exists i : z_i(v \cdot v) - 2z_i(v \cdot v') + z_i(v' \cdot v') > 5(2x(\min p_j)^{-1/2} + x^2)$  then conclude that  $v$  and  $v'$  are in different communities. Otherwise, conclude that  $v$  and  $v'$  are in the same community.

One could generate a reasonable classification based solely on this method of comparing vertices (with an appropriate choice of the parameters, as later detailed). However, that would require computing  $N_{r, r'[E]}(v \cdot v)$  for every vertex in the graph with fairly large  $r + r'$ , which would be slow. Instead, we use the fact that for any vertices  $v, v'$ , and  $v''$  with  $\sigma_v = \sigma_{v'} \neq \sigma_{v''}$ ,

$$\begin{aligned} P_{W_i}e_{\sigma_{v'}} \cdot P^{-1}P_{W_i}e_{\sigma_{v'}} - 2P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_{v'}} + P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_v} &= 0 \\ \leq P_{W_i}e_{\sigma_{v''}} \cdot P^{-1}P_{W_i}e_{\sigma_{v''}} - 2P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_{v''}} + P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_v} \end{aligned}$$

for all  $i$ , and the inequality is strict for at least one  $i$ . So, subtracting  $P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_v}$  from both sides gives us that

$$P_{W_i}e_{\sigma_{v'}} \cdot P^{-1}P_{W_i}e_{\sigma_{v'}} - 2P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_{v'}} \leq P_{W_i}e_{\sigma_{v''}} \cdot P^{-1}P_{W_i}e_{\sigma_{v''}} - 2P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_{v''}}$$

for all  $i$ , and the inequality is still strict for at least one  $i$ .

So, given a representative vertex in each community, we can determine which of them a given vertex,  $v$ , is in the same community as without needing to know the value of  $P_{W_i}e_{\sigma_v} \cdot P^{-1}P_{W_i}e_{\sigma_v}$  as follows.

**The Vertex\_classification\_algorithm.** The inputs are  $(v[], v', r, r', E, x, c)$ , where  $v[]$  is a list of vertices,  $v'$  is a vertex,  $r, r'$  are positive integers,  $E$  is a subset of  $G$ 's edges,  $x$  is a positive real number, and  $c$  is a real number between 0 and 1. It is assumed that  $z_i(v[\sigma] \cdot v[\sigma])$  satisfying

$$\sum_i ((1-c)\lambda_i)^{r+r'+j+1} z_i(v[\sigma] \cdot v[\sigma]) = \frac{(1-c)n}{c} N_{r+j, r'[E]}(v[\sigma] \cdot v[\sigma]) \text{ for } 0 \leq j < h'$$

have already been computed for every  $v[\sigma] \in v[]$ .

The algorithm is supposed to output  $\sigma$  such that  $v'$  is in the same community as  $v[\sigma]$ . It works as follows.

- (1) For each  $\sigma$  solve the system of equations

$$\sum_i ((1-c)\lambda_i)^{r+r'+j+1} z_i(v[\sigma] \cdot v') = \frac{(1-c)n}{c} N_{r+j, r'[E]}(v[\sigma] \cdot v') \text{ for } 0 \leq j < h'$$

- (2) If there exists a unique  $\sigma$  such that for all  $\sigma' \neq \sigma$  and all  $i$ ,

$$z_i(v[\sigma] \cdot v[\sigma]) - 2z_i(v[\sigma] \cdot v') \leq z_i(v[\sigma'] \cdot v[\sigma']) - 2z_i(v[\sigma'] \cdot v') + \frac{19}{3} \cdot (2x(\min p_j)^{-1/2} + x^2)$$

then conclude that  $v'$  is in the same community as  $v[\sigma]$ .

- (3) Otherwise, Fail.

This runs fairly quickly if  $r$  is large and  $r'$  is small because the algorithm only requires focusing on  $N_{r'}(v')$  vertices. This leads to the following plan for partial recovery. First, randomly select a set of vertices that is large enough to contain at least one vertex from each community with high

probability. Next, compare all of the selected vertices in an attempt to determine which of them are in the same communities. Then, pick one in each community. After that, use the algorithm above to attempt to determine which community each of the remaining vertices is in. As long as there actually was at least one vertex from each community in the initial set and none of the approximations were particularly bad, this should give a reasonably accurate classification.

**The Unreliable\_graph\_classification\_algorithm.** The inputs are  $(G, c, m, \epsilon, x)$ , where  $G$  is a graph,  $c$  is a real number between 0 and 1,  $m$  is a positive integer,  $\epsilon$  is a real number between 0 and 1, and  $x$  is a positive real number.

The algorithm outputs an alleged list of communities for  $G$ . It works as follows.

- (1) Randomly assign each edge in  $G$  to  $E$  independently with probability  $c$ .
- (2) Randomly select  $m$  vertices in  $G$ ,  $v[0], \dots, v[m-1]$ .
- (3) Set  $r = (1 - \frac{\epsilon}{3}) \log n / \log((1-c)\lambda_1) - h'$  and  $r' = \frac{2\epsilon}{3} \cdot \log n / \log((1-c)\lambda_1)$
- (4) Compute  $N_{r''[G \setminus E]}(v[i])$  for each  $r'' < r + h'$  and  $0 \leq i < m$ .
- (5) Run `Vertex_comparison_algorithm`( $v[i], v[j], r, r', E, x$ ) for every  $i$  and  $j$
- (6) If these give results consistent with some community memberships which indicate that there is at least one vertex in each community in  $v[]$ , randomly select one alleged member of each community  $v'[\sigma]$ . Otherwise, fail.
- (7) For every  $v''$  in the graph, compute  $N_{r''[G \setminus E]}(v'')$  for each  $r'' < r'$ . Then, run `Vertex_classification_algorithm`( $v'[], v'', r, r', E, x$ ) in order to get a hypothesized classification of  $v''$ .
- (8) Return the resulting classification.

The risk that this randomly gives a bad classification due to a bad set of initial vertices can be mitigated as follows. First, repeat the previous classification procedure several times. Next, discard any classification that differs too badly from the majority. Assuming that the procedure gives a good classification more often than not, this should eliminate any really bad classification. Finally, average the remaining classifications together. This last procedure completes the **Sphere comparison** algorithm.

**The Reliable\_graph\_classification\_algorithm (i.e., Sphere comparison).** The inputs are  $(G, c, m, \epsilon, x, T(n))$ , where  $G$  is a graph,  $c$  is a real number between 0 and 1,  $m$  is a positive integer,  $\epsilon$  is a real number between 0 and 1,  $x$  is a positive real number, and  $T$  is a function from the positive integers to itself.

The algorithm outputs an alleged list of communities for  $G$ . It works as follows.

- (1) Run `Unreliable_graph_classification_algorithm`( $G, c, m, \epsilon, x$ )  $T(n)$  times and record the resulting classifications.
- (2) Discard any classification that has greater than

$$4ke^{-\frac{(1-c)x^2\lambda_{h'}^2 \min p_i}{16dk(1+x)}} / (1 - e^{-\frac{(1-c)x^2\lambda_{h'}^2 \min p_i}{16dk(1+x)} \cdot ((\frac{(1-c)\lambda_{h'}^4}{4\lambda_1^2 d}) - 1)})$$

disagreement with more than half of the other classifications. In this step, define the disagreement between two classifications as the minimum disagreement over all bijections between their communities.

- (3) Let  $\{\sigma[i]\}$  be the remaining classifications. For each vertex  $v \in G$ , randomly select some  $i$  and assert that  $\sigma_v$  is the  $j$  that maximizes  $|\{v' : \sigma[1]_{v'} = j\} \cap \{v' : \sigma[i]_{v'} = \sigma[i]_v\}|$ . In other words, assume that  $\sigma[i]$  classifies  $v$  correctly and then translate that to a community of  $\sigma[1]$  by assuming the communities of  $\sigma[i]$  correspond to the communities of  $\sigma[1]$  that they have the greatest overlap with.
- (4) Return the resulting combined classification.

If the parameters of the SBM that the graph is drawn from are sufficiently favorable, then *Reliable\_graph\_classification\_algorithm* with the proper parameters solves weak recovery. More importantly, the algorithm's accuracy approaches 1 as the average vertex degree goes to  $\infty$ . So, we have the following, as proved in [2].

**Theorem 3.** *For any  $k \in \mathbb{Z}$ ,  $p \in (0, 1)^a$  with  $|p| = 1$ , and symmetric matrix  $Q$  with no two rows equal, there exist  $\epsilon(\delta) = O(1/\ln(\delta))$  and constant  $c_1 > 0$  such that for all sufficiently large  $\delta$  there exists an algorithm (**Sphere-comparison**) that recovers communities in graphs drawn from  $\text{SBM}(n, p, \delta Q/n)$  with accuracy at least  $1 - O_\delta(e^{-c_1\delta})$  in  $O_n(n^{1+\epsilon(\delta)})$ .*

**Corollary 1.** *For any  $k \in \mathbb{Z}$ ,  $p \in (0, 1)^a$  with  $|p| = 1$ , symmetric matrix  $Q$  with no two rows equal, and  $f(n) = \omega(1/n)$ , there exists an algorithm (**Sphere-comparison**) that recovers communities in graphs drawn from  $\text{SBM}(n, p, f(n)Q)$  with accuracy  $1 - o(1)$ .*

## 2.4 Exact recovery and the degree profiling algorithm

As long as  $s(n) = o(\log(n)/n)$ , there will be  $\omega(1)$  degree 0 vertices in each community of a graph drawn from  $\text{SBM}(n, p, s(n)Q)$  with high probability. There is no way to determine which of these

vertices are from which community, so exact recovery is impossible. That brings us to the question of when exact recovery is possible on graphs drawn from  $\text{SBM}(n, p, Q \ln(n)/n)$ . We can solve strong recovery on any such graph, so this essentially comes down to whether or not we can determine every vertex's community from its neighbors' communities.

The expected number of neighbors that a vertex in community  $i$  has in each community is given by the  $i$ th column of the matrix  $\ln(n)PQ$ , so one would expect the difficulty of determining a vertex's community based on its neighbors' communities to be a function of how much the columns of  $PQ$  differ from each other. In order to make that precise, we define the Chernoff-Hellinger divergence between two vectors as follows.

**Definition 9.** *Given  $\theta, \theta' \in \mathbb{R}^k$ , the Chernoff-Hellinger divergence between  $\theta$  and  $\theta'$  is given by*

$$D_+(\theta, \theta') = \max_{t \in [0,1]} \sum_{i=1}^k t\theta_i + (1-t)\theta'_i - (\theta_i)^t (\theta'_i)^{1-t}$$

In other words, the CH divergence between two vectors is the maximum, over all choices of weightings, of the difference between the sum of their entrywise weighted arithmetic means and the sum of their entrywise weighted geometric means. That allows us to state the following results

**Theorem 4.** *Let  $p \in (0, 1)^k$  with  $\sum p = 1$  and  $Q$  be a symmetric matrix with positive entries such that the CH divergence between any two different columns of  $PQ$  is at least 1. Then the degree profiling algorithm solves exact recovery on graphs drawn from  $\text{SBM}(n, p, Q \ln(n)/n)$ . Also, the degree profiling algorithm runs in polynomial time.*

**Remark 1.** *If we allow entries in  $Q$  to be 0 then some modifications to the degree profiling algorithm are needed. Other than that, this result still holds, with one exception. Specifically, if there exist  $1 \leq i, j \leq k$  such that  $D_+((PQ)_i, (PQ)_j) = 1$ , and for every pair of corresponding entries in  $(PQ)_i$  and  $(PQ)_j$ , either they are equal or one of them is 0, then no algorithm solves exact recovery on graphs drawn from  $\text{SBM}(n, p, Q \ln(n)/n)$ .*

On the flip side,

**Theorem 5.** *Let  $p \in (0, 1)^k$  with  $\sum p = 1$  and  $Q$  be a symmetric matrix with nonnegative entries such that  $PQ$  has two different columns with a CH divergence of less than 1. Then there is no algorithm that solves exact recovery on graphs drawn from  $\text{SBM}(n, p, Q \ln(n)/n)$ .*

The degree profiling algorithm essentially works by getting a preliminary classification of the vertex's communities, and then reclassifying each vertex based on the preliminary classifications of

its neighbors. More formally, it works as follows.

Degree – profiling( $G, \gamma$ ) :

1. Create two graphs  $G'$  and  $G''$  by copying every vertex in  $G$  to both graphs, and independently assign each edge in  $G$  to  $G'$  with probability  $\gamma$  and to  $G''$  otherwise.
2. Run **Sphere-comparison** on  $G'$  to obtain the preliminary classification  $\sigma' \in [k]^n$  (see Section 2.3.)
3. Determine the edge density between each pair of alleged communities, and use this information and the alleged communities' sizes to attempt to identify the communities up to symmetry.
4. For each node  $v \in [n]$ , determine in which community node  $v$  is most likely to belong to based on its degree profile computed from the preliminary classification  $\sigma'$  (see Section 6.3.2), and call it  $\sigma''_v$
5. For each node  $v \in [n]$ , determine in which community node  $v$  is most likely to belong to based on its degree profile computed from the preliminary classification  $\sigma''$  (see Section 6.3.2).
6. Return the resulting assignment of vertices to communities.

**Remark 2.** *Using the algorithm above to classify the communities of a graph requires knowing the parameters of the SBM the graph is drawn from in order to choose the correct parameters for Sphere-comparison and determine how likely a vertex would be to have its observed neighbors if it were in a given community. However, as shown in [3], it is possible to modify Degree-profiling so that it does not need to know anything about the SBM parameters.*

Also, note that if  $s(n) = \omega(\log(n)/n)$  and no two columns of  $Q$  are equal then we can solve exact recovery on graphs drawn from  $\text{SBM}(n, p, s(n)Q)$  by deleting every edge with the correct probability to convert the graphs to ones drawn from  $\text{SBM}(n, p, cQ \ln(n)/n)$  for some sufficiently large  $c$  and then using the degree profiling algorithm.

## Chapter 3

# The typicality sampling algorithm

Assume there exists an algorithm,  $A$ , that solves weak recovery on  $\text{SBM}(n, p, W)$ , and let  $(G, X)$  be drawn from  $\text{SBM}(n, p, W)$ . Also, for a given assignment of vertices to communities,  $x$ , let  $\bar{x} \in [k]^{n^2}$  be the vector such that for any  $1 \leq v, v' \leq n$ , if  $x_v = x_{v'}$  then  $\bar{x}_{v,v'} = 1/p_{x_v} - 1$  and otherwise  $\bar{x}_{v,v'} = -1$ . Also, for any subset  $S \subseteq [n]$ , let  $\bar{S} \in [k]^{n^2}$  be the vector such that  $\bar{S}_{v,v'}$  is 1 if  $v$  and  $v'$  are both in  $S$  and 0 otherwise. For an arbitrary  $S$  and  $x$ , we have that

$$\begin{aligned}\bar{x} \cdot \bar{S} &= -|S|^2 + \sum_{i=1}^k |\{v \in S : x_v = i\}|^2 / p_i \\ &= \sum_{i=1}^k (|\{v \in S : x_v = i\}| - p_i \cdot |S|)^2 / p_i\end{aligned}$$

So, the fact that  $A$  solves weak recovery implies that there exists  $\epsilon > 0$  such that  $\bar{X} \cdot \overline{A(G)} > \epsilon n^2$  with high probability. That in turn implies that if  $X'$  is drawn from the probability distribution of  $X$  given  $G$ , then

$$\begin{aligned}
\mathbb{E}[\bar{X} \cdot \bar{X}'] &= \sum_{g \in \{0,1\}^{\binom{n}{2}}} \sum_{x, x' \in [k]^n} \mathbb{P}[G = g] \cdot \mathbb{P}[X = x | G = g] \cdot \mathbb{P}[X = x' | G = g] \cdot (\bar{x} \cdot \bar{x}') \\
&= \sum_{g \in \{0,1\}^{\binom{n}{2}}} \mathbb{P}[G = g] \cdot \left( \sum_{x \in [k]^n} \mathbb{P}[X = x | G = g] \bar{x} \right)^2 \\
&\geq \sum_{g \in \{0,1\}^{\binom{n}{2}}} \mathbb{P}[G = g] \cdot \left( \overline{A(g)} \cdot \sum_{x \in [k]^n} \mathbb{P}[X = x | G = g] \bar{x} \right)^2 / n^2 \\
&\geq \frac{1}{n^2} \cdot \mathbb{E}^2[\overline{A(G)} \cdot \bar{X}] \\
&\geq (1 - o(1)) \epsilon^2 n^2
\end{aligned}$$

As long as  $x$  and  $x'$  both assign approximately the right number of vertices to each community, we have that

$$\bar{x} \cdot \bar{x}' = \sum_{1 \leq i, j \leq k} |\{v : x_v = i, x'_v = j\}|^2 / (p_i p_j) - n^2 + o(n^2)$$

so the fact that  $\mathbb{E}[\bar{X} \cdot \bar{X}'] \geq (1 - o(1)) \epsilon^2 n^2$  suggests that  $X$  is likely to be correlated with  $X'$ . That in turn suggests that we should be able to find an assignment of vertices to communities that is correlated with the true classification by randomly selecting a community assignment that is likely to result in the observed graph. More formally,

**Definition 10.** Let  $G$  be a graph drawn from  $\text{SBM}(n, p, W)$ ,  $x$  be an assignment of its vertices to communities, and  $\epsilon > 0$ . Then  $x$  is  $\epsilon$ -typical if for every community  $i$ , we have that

$$(p_i - \epsilon)n \leq |\{v : x_v = i\}| \leq (p_i + \epsilon)n$$

and the number of edges between vertices  $x$  assigns to  $i$  is in  $[(1 - \epsilon)p_i^2 W_{i,i} n^2 / 2, (1 + \epsilon)p_i^2 W_{i,i} n^2 / 2]$  and for every two communities  $i \neq j$ , the number of edges between a vertex that  $x$  assigns to  $i$  and a vertex that  $x$  assigns to  $j$  is in  $[(1 - \epsilon)p_i p_j W_{i,j} n^2, (1 + \epsilon)p_i p_j W_{i,j} n^2]$ . We call the set of all  $\epsilon$ -typical assignments of  $G$ 's vertices to communities  $T_\epsilon(G)$ .

The typicality sampling algorithm simply selects an  $\epsilon$ -typical assignment of the input graph at random for some given  $\epsilon$  and returns it. The simplest approach to proving that it works for graphs drawn from the SBM with a given set of parameters is to observe that the true assignment of vertices to communities is  $\epsilon$ -typical with high probability for any constant  $\epsilon$ , so the TSP returns

something with high probability. Then, one can compute the expected number of vertex classifications with negligible correlation with the true classification that are  $\epsilon$ -typical. If it is  $o(1)$  then the vertex classification returned by the TSP must be correlated with the true classification with high probability. While there are refinements of this technique, this is already enough to show that there are choices of parameters below the KS threshold for which the TSP detects communities.

## Chapter 4

# Acyclic belief propagation

### 4.1 ABP intuition

Recall the parameters:  $k$  and  $n$  are positive integers,  $p \in (0, 1)^k$  with  $\sum p_i = 1$ , and  $Q$  is a  $k \times k$  symmetric matrix with nonnegative entries. Then  $SBM(n, p, Q/n)$  generates  $n$ -vertex graphs by the following procedure. First, each vertex  $v$  is randomly and independently assigned a community  $\sigma_v$  such that the probability that  $\sigma_v = i$  is  $p_i$  for each  $i$ . Then, each pair of vertices  $v$  and  $v'$  have an edge put between them with probability  $Q_{\sigma_v, \sigma_{v'}}/n$ . Also, let  $\Omega_1, \dots, \Omega_k$  be the communities and  $P$  be the  $k \times k$  diagonal matrix such that  $P_{i,i} = p_i$  for each  $i$ . Now, consider the 2-community symmetric stochastic block model. In this case,  $k = 2$ ,  $p = [1/2, 1/2]$ ,  $Q_{i,j}$  is  $a$  if  $i = j$  and  $b$  otherwise for some  $a, b$ . Now, let  $\lambda_1 = \frac{a+b}{2}$  be the average degree of a vertex in a graph drawn from this model, and  $\lambda_2 = \frac{a-b}{2}$  be the other eigenvalue of  $PQ$ . Throughout this section we say  $f$  is approximately  $g$  or  $f \approx g$  when  $|f - g| = o(|f + g|)$  with probability  $1 - o(1)$ .

Our goal is to determine which of  $v$ 's vertices are in each community with an accuracy that is nontrivially better than that attained by random guessing. Obviously, the symmetry between communities ensures that we can never tell whether a given vertex is in community 1 or community 2, so the best we can hope for is to divide the vertices into two sets such that there is a nontrivial difference between the fraction of vertices from community 1 that are assigned to the first set and the fraction of vertices from community 2 that are assigned to the first set.

### 4.1.1 Amplifying luck

Let  $x$  and  $n - x$  be the numbers of vertices in community 1 and community 2 respectively. If the vertices are assigned sets at random, then the expected numbers of vertices from each community in the first set are  $\frac{x}{2}$  and  $\frac{n-x}{2}$ . However, by the central limit theorem, the probability distribution of the actual number of vertices from a given community in the first set is approximately a gaussian distribution with the mean stated previously and a variance of  $\frac{x}{4}$  or  $\frac{n-x}{4}$  as appropriate. That means that the probability distribution of the difference between the fraction of the vertices from community 1 assigned to the first set and the fraction of the vertices from the community 2 assigned to the first set is also approximately a bell curve. This one has a mean of  $\frac{1}{2} - \frac{1}{2} = 0$  and a variance of

$$\frac{x}{4}/x^2 + \frac{n-x}{4}/(n-x)^2 = \frac{1}{4x} + \frac{1}{4(n-x)} \approx \frac{1}{2n} + \frac{1}{2n} = \frac{1}{n}$$

That means that it has a standard deviation of  $\approx 1/\sqrt{n}$ , and the difference between the fraction of the vertices from community 1 assigned to the first set and the fraction of the vertices from community 2 assigned to the first set will typically have a magnitude on the order of  $1/\sqrt{n}$ . Label the sets  $S_1$  and  $S_2$  such that the fraction of the vertices from  $\Omega_1$  that were assigned to  $S_1$  is at least as large as the fraction of the vertices from  $\Omega_2$  that were assigned to  $S_1$ . For the rest of this section, we will consider the difference between these fractions to be fixed.

Consider determining the community of  $v$  using belief propagation, assuming some preliminary guesses about the vertices  $t$  edges away from it, and assuming that the subgraph of  $G$  induced by the vertices within  $t$  edges of  $v$  is a tree. For any vertex  $v'$  such that  $d(v, v') < t$ , let  $C_{v'}$  be the set of the children of  $v'$ . If we believe based on either our prior knowledge or propagation of beliefs up to these vertices that  $v''$  is in community 1 with probability  $\frac{1}{2} + \frac{1}{2}\epsilon_{v''}$  for each  $v'' \in C_{v'}$ , then the algorithm will conclude that  $v'$  is in community 1 with a probability of

$$\frac{\prod_{v'' \in C_{v'}} (\frac{a+b}{2} + \frac{a-b}{2}\epsilon_{v''})}{\prod_{v'' \in C_{v'}} (\frac{a+b}{2} + \frac{a-b}{2}\epsilon_{v''}) + \prod_{v'' \in C_{v'}} (\frac{a+b}{2} - \frac{a-b}{2}\epsilon_{v''})}.$$

If all of the  $\epsilon_{v''}$  are close to 0, then this is approximately equal to

$$\frac{1 + \sum_{v'' \in C_{v'}} \frac{a-b}{a+b}\epsilon_{v''}}{2 + \sum_{v'' \in C_{v'}} \frac{a-b}{a+b}\epsilon_{v''} + \sum_{v'' \in C_{v'}} -\frac{a-b}{a+b}\epsilon_{v''}} = \frac{1}{2} + \frac{a-b}{a+b} \sum_{v'' \in C_{v'}} \frac{1}{2}\epsilon_{v''}.$$

That means that the belief propagation algorithm will ultimately conclude that  $v$  is in community 1 with a probability of approximately  $\frac{1}{2} + \frac{1}{2}(\frac{a-b}{a+b})^t \sum_{v'': d(v, v'')=t} \epsilon_{v''}$ . If there exists  $\epsilon$  such that

$E_{v'' \in \Omega_1}[\epsilon_{v''}] = \epsilon$  and  $E_{v'' \in \Omega_2}[\epsilon_{v''}] = -\epsilon$  (recall that  $\Omega_i = \{v : \sigma_v = i\}$ ), then on average we would expect to assign a probability of approximately  $\frac{1}{2} + \frac{1}{2} \left( \frac{(a-b)^2}{2(a+b)} \right)^t \epsilon$  to  $v$  being in its actual community, which is enhanced as  $t$  increases when  $\text{SNR} > 1$ .

Equivalently, given a vertex  $v$  and a small  $t$ , the expected number of vertices that are  $t$  edges away from  $v$  is approximately  $\left(\frac{a+b}{2}\right)^t$ , and the expected number of these vertices in the same community as  $v$  is approximately  $\left(\frac{a-b}{2}\right)^t$  greater than the expected number of these vertices in the other community. So, if we had some way to independently determine which community a vertex is in with an accuracy of  $\frac{1}{2} + \epsilon$  for small  $\epsilon$ , we could guess that each vertex is in the community that we think that the majority of the vertices  $t$  steps away from it are in to determine its community with an accuracy of roughly  $\frac{1}{2} + \left( \frac{(a-b)^2}{2(a+b)} \right)^{t/2} \epsilon$ .

To obtain initial estimates, we simply guess the vertices' communities at random as described in the previous section, with the expectation that the fractions of the vertices from the two communities assigned to a community will differ by  $\Theta(1/\sqrt{n})$  by the Central Limit Theorem. Once we have even such weak information on which vertex is in which community, we can try to improve our classification of a given vertex by factoring in our knowledge of what communities the nearby vertices are in. For a vertex  $v$  and integer  $t$ , let  $N_t(v)$  be the number of vertices  $t$  edges away from  $v$ ,  $\Delta_t(v)$  be the difference between the number of vertices  $t$  edges away from  $v$  that are in community 1 and the number of vertices  $t$  edges away from  $v$  that are in community 2, and  $\tilde{\Delta}_t(v)$  be the difference between the number of vertices  $t$  edges away from  $v$  that are in  $S_1$  and the number of vertices  $t$  edges away from  $v$  that are in  $S_2$ . For small  $t$ ,

$$E[N_t(v)] \approx \left( \frac{a+b}{2} \right)^t$$

and

$$E[\Delta_t(v)] \approx \left( \frac{a-b}{2} \right)^t \cdot (-1)^{\sigma_v}$$

For any fixed values of  $N_t(v)$  and  $\Delta_t(v)$ , the probability distribution of  $\tilde{\Delta}_t(v)$  is essentially a Gaussian distribution with a mean of  $\Theta(\Delta_t(v)/\sqrt{n})$  and a variance of  $\approx N_t(v)$  because it is the sum of  $N_t(v)$  nearly independent variables that are approximately equally likely to be 1 or  $-1$ . So,  $\tilde{\Delta}_t(v)$  is positive with a probability of  $\frac{1}{2} + \Theta(\Delta_t(v)/\sqrt{|N_t(v)|n})$ . In other words, if  $v$  is in community 1 then  $\tilde{\Delta}_t(v)$  is positive with a probability of

$$\frac{1}{2} - \Theta \left( \left( \frac{a-b}{2} \right)^t \cdot \left( \frac{a+b}{2} \right)^{-t/2} / \sqrt{n} \right)$$

and if  $v$  is in community 2 then  $\tilde{\Delta}_t(v)$  is positive with a probability of

$$\frac{1}{2} + \Theta\left(\left(\frac{a-b}{2}\right)^t \cdot \left(\frac{a+b}{2}\right)^{-t/2} / \sqrt{n}\right)$$

If  $(a-b)^2 \leq 2(a+b)$ , then this is not improving the accuracy of the classification, so this technique is useless. On the other hand, if  $(a-b)^2 > 2(a+b)$ , the classification becomes more accurate as  $t$  increases. However, this formula says that to classify vertices with an accuracy of  $1/2 + \Omega(1)$ , we would need to have  $t$  such that

$$\left(\frac{a-b}{2}\right)^{2t} = \Omega\left(\left(\frac{a+b}{2}\right)^t n\right)$$

However, unless<sup>1</sup>  $a$  or  $b$  is 0, that would imply that

$$\left(\frac{a+b}{2}\right)^{2t} = \omega\left(\left(\frac{a-b}{2}\right)^{2t}\right) = \omega\left(\left(\frac{a+b}{2}\right)^t n\right)$$

which means that  $\left(\frac{a+b}{2}\right)^t = \omega(n)$ . It is obviously impossible for  $N_t(v)$  to be greater than  $n$ , so this  $t$  is too large for the approximation to hold. The problem is, the approximation assumes that each vertex at a distance of  $t-1$  from  $v$  has one edge leading back towards  $v$ , and that the rest of its edges lead towards new vertices. Once a significant fraction of the vertices are less than  $t$  edges away from  $v$ , a significant fraction of the edges incident to vertices  $t-1$  edges away from  $v$  are part of loops and thus do not lead to new vertices.

#### 4.1.2 Nonbacktracking walks

An obvious way to solve the problem caused by running out of vertices would be to simply count the walks of length  $t$  from  $v$  to vertices in  $S_1$  or  $S_2$ . Recall that a *walk* is a series of vertices such that each vertex in the walk is adjacent to the next, and a *path* is a walk with no repeated vertices. The last vertex of such a walk will be adjacent to an average of approximately  $a/2$  vertices in its community outside the walk and  $b/2$  vertices in the other community outside the walk. However, it will also be adjacent to the second to last vertex of the walk, and maybe some of the other vertices in the walk as well. As a result, the number of walks of length  $t$  from  $v$  to vertices in  $S_1$  or  $S_2$  cannot be easily predicted in terms of  $v$ 's community. So, the numbers of such walks are not useful

---

<sup>1</sup>If  $a = 0$  and  $k = 2$  or  $b = 0$ , then classifying vertices based on the sign of  $\tilde{\Delta}_t(v)$  for suitable  $t$  is likely to work, but this is pointlessly complicated because every component of the graph consists of all vertices of one community or vertices of alternating communities.

for classifying vertices.

We could deal with this issue by counting paths<sup>2</sup> of length  $t$  from  $v$  to vertices in  $S_1$  and  $S_2$ . Given a path of length  $t-1$ , the expected number of vertices outside the path in the same community as its last vertex that are adjacent to it is approximately  $a/2$  and the expected number of vertices outside the path in the opposite community as its last vertex that are adjacent to it is approximately  $b/2$ . So, the expected number of paths of length  $t$  from  $v$  is approximately  $(\frac{a+b}{2})^t$  and the expected difference between the number that end in vertices in the same community as  $v$  and the number that end in the other community is approximately  $(\frac{a-b}{2})^t$ . The problem with this is that counting all of these paths is inefficient.

The compromise we use is to count nonbacktracking walks ending at  $v$ , i.e. walks that never repeat the same edge twice in a row. We can efficiently determine how many nonbacktracking walks of length  $t$  there are from vertices in  $S_i$  to  $v$  by using the fact that the number of nonbacktracking walks of length  $t$  starting at a vertex in  $S_i$  and having  $v'$  and  $v$  as their last two vertices is equal to the sum over all  $v'' \neq v$  such that  $v''$  is adjacent to  $v'$  of the number of nonbacktracking walks of length  $t-1$  starting at a vertex in  $S_i$  and having  $v''$  and  $v'$  as their last two vertices. Furthermore, most nonbacktracking walks of a given length that is logarithmic in  $n$  are paths, so it seems reasonable to expect that counting nonbacktracking walks instead of paths in our algorithm will have a negligible effect on the accuracy.

---

<sup>2</sup>This type of approach is considered in [8].

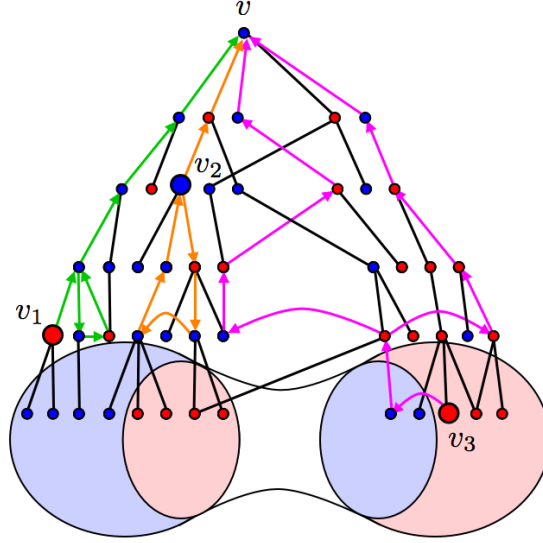


Figure 4.1: The ABP algorithm amplifies the belief of vertex  $v$  by considering all the walks of a given length that end at it. To avoid being disrupted by backtracking or cycling the beliefs on short loops, the algorithm considers only walks that do not repeat the same vertex within  $r$  steps, i.e.,  $r$ -nonbacktracking walks. For example, when  $r = 3$  and when the walks have length 7, the green walk starting at vertex  $v_1$  is discarded, whereas the orange walk starting at the vertex  $v_2$  is counted. Note also that the same vertex can lead to multiple walks, as illustrated with the two magenta walks from  $v_3$ . Since there are approximately equally many such walks between any two vertices, if the majority of the vertices were initially classified as blue, this is likely to classify all of the vertices as blue. We hence need a compensation step to prevent the classification from becoming biased towards one community.

More precisely, that suggests the following approach. Define  $y_{v,v'}^{(t)}$  to be the number of nonbacktracking walks of length  $t$  that start at vertices in  $S_2$  and end in the directed edge  $(v', v)$  minus the number of nonbacktracking walks of length  $t$  that start at vertices in  $S_1$  and end in  $(v', v)$ . Also, define  $y_v^{(t)}$  to be the overall difference between the number of nonbacktracking walks of length  $t$  from vertices in  $S_2$  to  $v$  and the number of nonbacktracking walks of length  $t$  from vertices in  $S_1$  to  $v$ . Their values can be efficiently computed by means of the following procedure:

1. For every  $(v, v') \in E(G)$  :
  - If  $v' \in S_2$ , set  $y_{v,v'}^{(1)} = 1$
  - Otherwise, set  $y_{v,v'}^{(1)} = -1$

2. For every  $1 < t \leq m$  and  $(v, v') \in E(G)$  :

$$\text{Set } y_{v,v'}^{(t)} = \sum_{v'' : (v'', v') \in E(G), v'' \neq v} y_{v'', v'}^{(t-1)}$$

3. For every  $(v, v') \in E(G)$  and  $v \in G$

$$\text{Set } y_v^{(m)} = \sum_{v' : (v, v') \in E(G)} y_{v,v'}^{(m)}$$

One way of viewing this algorithm is that  $y_{v,v'}^{(t)}$  represents our current belief about what community  $v'$  is in, disregarding any information derived from the fact that it is next to  $v$ . We start with fairly unconfident beliefs about the vertices' communities, and then derive more and more confident beliefs about the vertices' communities by taking our beliefs about their neighbors' communities into account.

### 4.1.3 Compensation for the average value

$y_{v,v'}^{(1)}$  has an average value of  $\Theta(1/\sqrt{n})$  for  $v'$  in community 2 and  $-\Theta(1/\sqrt{n})$  for  $v'$  in community 1. Also,  $y_{v,v'}^{(1)}$  has a variance of order 1. For a random  $(v, v') \in E(G)$ ,  $v'$  will have an average of approximately  $a/2$  neighbors other than  $v$  in its community and  $b/2$  neighbors other than  $v$  in the other community. So, by induction on  $t$ , we would expect that  $y_{v,v'}^{(t)}$  would have an average value of  $\Theta((\frac{a-b}{2})^t / \sqrt{n})$  for  $v'$  in community 2 and  $-\Theta((\frac{a-b}{2})^t / \sqrt{n})$  for  $v'$  in community 1. Since  $y_{v',v''}^{(t-1)}$  should be approximately independent for different  $v''$  adjacent to  $v'$ , we would also expect that  $y_{v,v'}^{(t)}$  would have an empirical variance of approximately  $(\frac{a+b}{2})^t$ , and thus a standard deviation of approximately  $\sqrt{(\frac{a+b}{2})^t}$ . So, for  $t$  such that  $(\frac{a-b}{2})^t / \sqrt{n} > \sqrt{(\frac{a+b}{2})^t}$ , we would expect that we could determine the community of  $v'$  from  $y_{v,v'}^{(t)}$  with accuracy  $1/2 + \Omega(1)$ .

The problem with this reasoning is that the average value over all  $(v, v') \in E(G)$  of  $y_{v,v'}^{(1)}$  will not be exactly 0. It will also tend to have an absolute value on the order of  $1/\sqrt{n}$ . That means that the average value over all  $(v, v') \in E(G)$  of  $y_{v,v'}^{(t)}$  will have an absolute value of  $\Theta((\frac{a+b}{2})^t / \sqrt{n})$ . If we hold the average value of  $y_{v,v'}^{(t-1)}$  fixed then that means that  $E[y_{v,v'}^{(t)} | N_1(v')]$  will have an empirical variance of  $\Theta((\frac{a+b}{2})^{2t}/n)$ , and thus that  $y_{v,v'}^{(t)}$  will also have an empirical variance of at least  $\Theta((\frac{a+b}{2})^{2t}/n)$ . This implies that the standard deviation of  $y_{v,v'}^{(t)}$  will always be much greater than the difference between the average value of  $y_{v,v'}^{(t)}$  for  $v'$  in community 1 and the average value of  $y_{v,v'}^{(t)}$  for  $v'$  in community 2, which would render attempts to classify  $v'$  based on  $y_{v,v'}^{(t)}$  ineffective.

**Remark 3.** *The simple way to fix this would be to add a step where we subtract the average value of  $y^{(t)}$  from every element of  $y^{(t)}$  so its sum is 0 for every  $t$  like we did in the version of ABP in*

Section 2.2.1. However, this does not extend easily to the general Stochastic Block Model, and we want a solution that does.

In order to prevent this, we need to stop the average value of  $y_{v,v'}^{(t)}$  from getting too large. It will tend to multiply by roughly  $\frac{a+b}{2}$  each time  $t$  increases by 1, so the average value of  $y_{v,v'}^{(t)} - \frac{a+b}{2} y_{v,v'}^{(t-1)}$  will probably be much smaller than the average value of  $y_{v,v'}^{(t)}$ . So, if we pick some  $1 < i \leq m$  and redefine  $y_{v,v'}^{(i)}$  so that

$$y_{v,v'}^{(i)} = -\frac{a+b}{2} y_{v,v'}^{(i-1)} + \sum_{v'':(v'',v') \in E(G), v'' \neq v} y_{v',v''}^{(i-1)} \quad (4.1)$$

for all  $(v, v') \in E(G)$ , the average value of  $y_{v,v'}^{(i)}$  will be much smaller than it would have been. Since the difference between the average values of  $y_{v,v'}^{(t)}$  over  $v'$  in different communities grows as  $\Theta((\frac{a-b}{2})^t / \sqrt{n})$ , this redefinition will merely change the difference between the average values over  $v'$  in different communities of  $y_{v,v'}^{(i)}$  from  $\Theta((\frac{a-b}{2})^i / \sqrt{n})$  to

$$\Theta \left( \left( \frac{a-b}{2} \right)^i / \sqrt{n} - \frac{a+b}{2} \cdot \left( \frac{a-b}{2} \right)^{i-1} / \sqrt{n} \right) = -\Theta \left( b \left( \frac{a-b}{2} \right)^{i-1} / \sqrt{n} \right).$$

However, the average value of  $y_{v,v'}^{(i)}$  will still be nonzero, and if we continue to set  $y_{v,v'}^{(t)} = \sum_{v'':(v'',v') \in E(G), v'' \neq v} y_{v',v''}^{(t-1)}$  for all  $t > i$ , the average value of  $y_{v,v'}^{(t)}$  would resume increasing in magnitude faster than the average difference between  $y_{v,v'}^{(t)}$  for different communities of  $v'$ . This creates the risk that it would still eventually get too large. So, in order to actually fix the problem, it may be necessary to repeat the step where its magnitude is reduced. More precisely, we may have to choose several indices  $t_0, t_1, \dots, t_{m'}$  and redefine  $y_{v,v'}^{(t_i)}$  for each  $i$  so that

$$y_{v,v'}^{(t_i)} = -\frac{a+b}{2} y_{v,v'}^{(t_i-1)} + \sum_{v'':(v'',v') \in E(G), v'' \neq v} y_{v',v''}^{(t_i-1)} \quad (4.2)$$

for every  $(v, v') \in E(G)$ .

Once we have made these modifications, it will be the case that for sufficiently large  $m$ , the average value for  $v'$  in community 1 of  $y_{v,v'}^{(m)}$  will differ from the average value for  $v'$  in community 2 of  $y_{v,v'}^{(m)}$  by a constant multiple of the standard deviation of  $y_{v,v'}^{(m)}$ . Then, we define  $y_v^{(m)} = \sum_{v':(v,v') \in E(G)} y_{v,v'}^{(m)}$  in order to simplify it to a function of one vertex which is still correlated with the vertex's community. Unfortunately, this does not guarantee that simply dividing the vertices into those with a positive value of  $y_v^{(m)}$  and those with a negative value of  $y_v^{(m)}$  will give a useful partition. It could

be the case that the fraction of  $v$  for which  $y_v^{(m)}$  is positive is the same for both communities, but  $y_v^{(m)}$  is typically more strongly positive or less strongly negative for  $v$  in one community than  $v$  in the other. So, we randomly assign each vertex to a set with a probability that scales linearly with  $y_v^{(m)}$  in order to ensure that having a higher average value of  $y_v^{(m)}$  actually leads to having greater representation in one of the proposed communities.

#### 4.1.4 Vanilla ABP

We present first a simplified version of our algorithm. Our proof relies on a modified version described below, but this version captures the essence of our algorithm while avoiding technicalities required for the proof.

ABP $^*(G, m, m', r, \lambda_1)$ :

1. For each adjacent  $v$  and  $v'$  in  $G$ , randomly draw  $y_{v,v'}^{(1)}$  from a Gaussian distribution with mean 0 and variance 1. Also, consider  $y_{v,v'}^{(t)}$  as having a value of 0 whenever  $t < 1$ .
2. For each  $1 < t \leq m$ , and each adjacent  $v$  and  $v'$  in  $G$ , set

$$y_{v,v'}^{(t)} = \sum_{v'' : (v', v'') \in E(G), v'' \neq v} y_{v', v''}^{(t-1)}$$

unless  $(v, v')$  is part of a cycle of length  $r$  or less.<sup>3</sup> If it is, then let the other vertex in the cycle that is adjacent to  $v$  be  $v'''$ , and the length of the cycle be  $r'$ , and set

$$y_{v,v'}^{(t)} = \sum_{v'' : (v', v'') \in E(G), v'' \neq v} y_{v', v''}^{(t-1)} - \sum_{v'' : (v, v'') \in E(G), v'' \neq v', v'' \neq v'''} y_{v, v''}^{(t-r')}$$

unless  $t = r'$ , in which case, set

$$y_{v,v'}^{(t)} = \sum_{v'' : (v', v'') \in E(G), v'' \neq v} y_{v', v''}^{(t-1)} - y_{v''', v}^{(1)}.$$

3. Set  $Y$  to be the  $n \times m$  matrix such that for all  $t$  and  $v$ ,

$$Y_{v,t} = \sum_{v' : (v', v) \in E(G)} y_{v,v'}^{(t)}$$

---

<sup>3</sup>See Remark (2) in Section 2.2.1 for how to modify the algorithm in the case of multiple cycles.

4. Set  $M$  to be the  $m \times m$  matrix such that  $M_{i,i} = 1$  and  $M_{i,i+1} = -\lambda_1$  for all  $i$ , and all other entries of  $M$  are 0. Also, let  $e_m \in \mathbb{R}^m$  be the vector with an  $m$ -th entry of 1 and all other entries equal to 0. Set

$$y' = Y M^{m'} e_m.$$

5. Return  $(\{v : y'_v > 0\}, \{v : y'_v \leq 0\})$ .

**Remark 4.** In the  $r = 2$  case, one does not need to find cycles and one can exit step 2. after the second line. In general when running this algorithm, one should use variables that are accurate to within a factor of less than  $(\lambda_2/\lambda_1)^m$ . As mentioned above, we rely on a less compact version of the algorithm to prove the theorem, but expect that the above also succeeds at detection as long as  $m > 2 \ln(n)/\ln(\text{SNR}) + \omega(m')$  and  $m' > m \ln(\lambda_1^2/\lambda_2^2)/(\ln(n) - \omega(1))$ .

**Remark 5.** This version of ABP is mostly the same as the one in Section 2.2.1. However, the previous version compensates for biases in  $y^{(1)}$  by subtracting the average value of  $y^{(t)}$  from each entry in  $y^{(t)}$  for every  $t$  in order to prevent it from ever becoming significantly biased. This version compensates for biases by a variant of the method explained in the previous subsection instead. More specifically, it uses the method explained in the previous subsection except that it calculates all of the  $y^{(t)}$  without using any form of compensation and then adjusts the results in order to essentially apply the compensation retroactively.

**Implementation details.** Note that  $ABP^*$  has several differences from the algorithm outlined in previous sections. First of all, we initialize the  $y_{v,v'}^{(0)}$  using a random value that is drawn from a normal distribution because a probability distribution that is a multidimensional normal distribution is easier to analyse than a probability distribution that is evenly distributed over the vertices of an  $n$ -dimensional hypercube. Secondly, we require that our walks never repeat the same vertex within  $r$  steps for some  $r$ , rather than merely requiring that they not backtrack. This allows us to use the expected number of walks between vertices in our analysis without worrying about the tiny probability that there is a dense tangle in the graph with a huge number of nonbacktracking walks between its vertices. Making this modification to the algorithm requires adding an extra part to the recursion step where walks that just repeated a vertex are cancelled out, specifically the second half of step 3 of the algorithm above. The resulting algorithm is called the *acyclic belief propagation* algorithm because it counts walks that do not contain any small cycles. Thirdly, we move all of the recursion steps that compensate for the average value to the end of the algorithm. This is possible

because the operation that takes  $y^{(t-1)}$  as input and outputs a list that has a value of

$$-\frac{a+b}{2}y_{v,v'}^{(t-1)} + \sum_{v'':(v'',v') \in E(G), v'' \neq v} y_{v',v''}^{(t-1)}$$

for each  $(v, v') \in E$  commutes with the one that simply outputs a list that has a value of

$\sum_{v'':(v'',v') \in E(G), v'' \neq v} y_{v',v''}^{(t-1)}$  for each  $(v, v') \in E$ . The algorithm generates  $y^{(m)}$  by applying these two operations in some sequence to  $y^{(1)}$ , so we can calculate it by applying the later operation  $m - m'$  times and then applying the former operation  $m'$  times. Actually, the algorithm takes this one step farther by applying the later operation  $m$  times and then calculating how the result would have changed if it had applied the former the appropriate number of times, but it still has the same result.

The full Acyclic Belief Propagation algorithm also has a few differences from  $ABP^*$  that make it easier to prove that it works. In particular, we randomly select a small fraction of the graph's edges at the beginning of the algorithm. Then, we require one specific step of each nonbacktracking walk to use one of the selected edges, and all of their other steps to use edges that have not been selected. Since the selected edges are nearly independent of the rest of the graph, this allows us to more easily prove that the values of  $y_{v',v''}^{(t-1)}$  for  $v'$  adjacent to  $v$  will not become dependent in a way that disrupts the algorithm.

#### 4.1.5 ABP for the symmetric SBM

**Theorem 6.** *Let  $a$  and  $b$  be positive real numbers such that  $(a - b)^2 > 2(a + b)$ , and  $S$  be the 2-community symmetric stochastic block model with these parameters. There exist constants  $\epsilon, l, r, c > 0$  and  $m = \Theta(\log(n))$  such that when the basic 2-community symmetric acyclic belief propagation algorithm is run on these parameters and a random  $G \in S$ , the expected difference between the fraction of vertices from community 1 that are in  $S_1$  and the fraction of vertices from community 2 that are in  $S_1$  is at least  $\epsilon$ .*

The basic 2-community symmetric acyclic belief propagation algorithm is as follows.

$2CS - ABP^*(G, m, r, l, c, \lambda_1)$  :

1. Find all cycles of length  $r$  or less in  $G$ .
2. For every vertex  $v \in G$ , randomly assign  $x_v$  according to a Normal distribution with mean 0 and variance 1. For each adjacent  $v$  and  $v'$ , let  $y_{v,v'}^{(1)} = x_{v'}$ , and  $y_{v,v'}^{(t)} = 0$  for all  $t \leq 0$ .

3. For each  $1 \leq t \leq m$ , and each adjacent  $(v, v') \in E(G)$ , set

$$y_{v,v'}^{(t)} = \sum_{v'':(v',v'') \in E(G), v'' \neq v} y_{v',v''}^{(t-1)}$$

unless  $v, v'$  is part of a cycle of length  $r$  or less. If it is, then let the other vertex in the cycle that is adjacent to  $v$  be  $v'''$ , and the length of the cycle be  $r'$ . Set

$$y_{v,v'}^{(t)} = \sum_{v'':(v',v'') \in E(G), v'' \neq v} y_{v',v''}^{(t-1)} - \sum_{v'':(v,v'') \in E(G), v'' \neq v', v'' \neq v'''} y_{v,v''}^{(t-r')}$$

unless  $t = r'$ . In that case, set

$$y_{v,v'}^{(t)} = \sum_{v'':(v',v'') \in E(G), v'' \neq v} y_{v',v''}^{(t-1)} - x_v.$$

4. Set  $Y$  to be the  $n \times m$  matrix such that for all  $t$  and  $v$ ,

$$Y_{v,t} = \sum_{v':(v',v) \in E(G)} y_{v',v}^{(t)}$$

5. Set  $M$  to be the  $m \times m$  matrix such that  $M_{i,i} = 1$  and  $M_{i,i+1} = -\lambda_1$  for all  $i$ , and all other entries of  $M$  are equal to 0. Also, let  $e_m \in \mathbb{R}^m$  be the vector with an  $m$ -th entry of 1 and all other entries equal to 0. Set

$$y^{(m)} = Y M^{\lceil \frac{m-3r-1}{l} \rceil} e_m.$$

6. Set  $c' = c \cdot \sqrt{\sum_{v \in G} (y_v^{(m)})^2 / n}$ . Create sets of vertices  $S_1$  and  $S_2$  as follows. For each vertex  $v$ , if  $y_v^{(m)} < -c'$ , assign  $v$  to  $S_1$ . If  $y_v^{(m)} > c'$ , then assign  $v$  to  $S_2$ . Otherwise, assign  $v$  to  $S_2$  with probability  $1/2 + y_v^{(m)}/2c'$  and  $S_1$  otherwise. Return  $(S_1, S_2)$ .

We believe that the difference between the fraction of vertices from community 1 that this algorithm puts in  $S_1$  and the fraction of vertices from community 2 that this algorithm puts in  $S_1$  will be at least  $\epsilon$  with probability  $1 - o(1)$ . However, in order to prove that we can detect communities reliably we use the following slightly modified form of the algorithm.

**Theorem 7.** *Let  $a$  and  $b$  be positive real numbers such that  $(a - b)^2 > 2(a + b)$ , and  $S$  be the 2-community symmetric stochastic block model with these parameters. There exist constants  $\epsilon, l, r, c, \gamma > 0$  and  $m = \Theta(\log(n))$  such that when the 2-community symmetric acyclic belief propagation algorithm is run on these parameters and a random  $G \in S$ , the difference between the fraction*

of vertices from community 1 that are in  $S_1$  and the fraction of vertices from community 2 that are in  $S_1$  is at least  $\epsilon$  with probability  $1 - o(1)$ .

The 2-community symmetric acyclic belief propagation algorithm is as follows.

$2CS - ABP(G, m, r, l, c, \lambda_1, \gamma) :$

1. Randomly and independently select each edge in  $G$  with probability  $\gamma$ . Put all of the selected edges in a set  $\Gamma$ , and remove them from the graph.
2. For every vertex  $v \in G$ , randomly assign  $x_v$  according to a Normal distribution with mean 0 and variance 1. For each adjacent  $v$  and  $v'$ , let  $y_{v,v'}^{(1)} = x_{v'}$ , and  $y_{v,v'}^{(t)} = 0$  for all  $t \leq 0$ .
3. For each  $1 \leq t \leq m$ , and each adjacent  $(v, v') \in E(G)$ , set

$$y_{v,v'}^{(t)} = \sum_{v'' : (v', v'') \in E(G), v'' \neq v} y_{v',v''}^{(t-1)}$$

unless  $v, v'$  is part of a cycle of length  $r$  or less. If it is, then let the other vertex in the cycle that is adjacent to  $v$  be  $v'''$ , and the length of the cycle be  $r'$ . Set

$$y_{v,v'}^{(t)} = \sum_{v'' : (v', v'') \in E(G), v'' \neq v} y_{v',v''}^{(t-1)} - \sum_{v'' : (v, v'') \in E(G), v'' \neq v', v'' \neq v'''} y_{v,v''}^{(t-r')}$$

unless  $t = r'$ . In that case, set

$$y_{v,v'}^{(r')} = \sum_{v'' : (v', v'') \in E(G), v'' \neq v} y_{v',v''}^{(r'-1)} - x_v.$$

4. Set  $Y$  to be the  $n \times m$  matrix such that for all  $t$  and  $v$ ,

$$Y_{v,t} = \sum_{v' : (v', v) \in E(G)} y_{v,v'}^{(t)}$$

5. Set  $M$  to be the  $m \times m$  matrix such that  $M_{i,i} = 1$  and  $M_{i,i+1} = -(1 - \gamma)\lambda_1$  for all  $i$ , and all other entries of  $M$  are equal to 0. Also, let  $e_m \in \mathbb{R}^m$  be the vector with an  $m$ -th entry of 1 and all other entries equal to 0. Set

$$y^{(m)} = Y M^{\lceil \frac{m-3r-1}{l} \rceil} e_m.$$

6. For each  $v \in G$ , set

$$y'_v = \sum_{v':(v,v') \in \Gamma} y_{v'}^{(m)}$$

7. For each  $v \in G$ , set  $y''_v$  to be the sum of  $y'_{v'}$  for all vertices  $v'$  that are exactly  $\lfloor \sqrt{\ln \ln n} \rfloor$  edges away from  $v$ .

8. Set  $c' = c \cdot \sqrt{\sum_{v \in G} (y''_v)^2 / n}$ . Create sets of vertices  $S_1$  and  $S_2$  as follows. For each vertex  $v$ , if  $y''_v < -c'$ , assign  $v$  to  $S_1$ . If  $y''_v > c'$ , then assign  $v$  to  $S_2$ . Otherwise, assign  $v$  to  $S_2$  with probability  $1/2 + y''_v/2c'$  and  $S_1$  otherwise. Return  $(S_1, S_2)$ .

**Remark 6.** Note that the last theorem is simply Theorem 1 in the 2 community symmetric case with some parts of the initialization step replaced by an assumption that parameters are chosen correctly. Also, the one before it follows from a slightly simplified version of the same proof.

**Remark 7.** For the  $k$ -community symmetric stochastic block model, the above is largely unchanged. The key differences are that  $\lambda_1 = \frac{a+(k-1)b}{k}$ ,  $\lambda_2 = \frac{a-b}{k}$ , the requirement for the algorithm to work is that  $(a-b)^2 > k(a+(k-1)b)$ , and if the requirements are met the  $kCS-ABP^*$  algorithm distinguishes between every pair of communities with expected accuracy at least  $\epsilon$ .

**Remark 8.** In this algorithm, the initialization step and computing  $y^{(t)}$  for a given  $t$  both run in  $O(n)$  time.  $y^{(m)}$  can be computed in  $O(n \log n)$  time if it is done the efficient way,  $y'$  takes  $O(n)$  time to compute, and computing  $y''$  takes  $O(n \log n)$  time. Determining  $S_1$  and  $S_2$  from  $y''$  can also be done in  $O(n)$  time, so this whole algorithm can be run in  $O(n \log n)$  time.

#### 4.1.6 Spectral view of ABP

An alternative perspective on this algorithm is the following. Assume for the moment that there are exactly  $n/2$  vertices in each community, and let  $M$  be the expected adjacency matrix, the matrix such that  $M_{v,v'}$  is  $a/n$  if  $v$  and  $v'$  are in the same community and  $b/n$  if they are not. This matrix has an eigenvector whose entries are all 1 with eigenvalue  $\frac{a+b}{2}$ , an eigenvector whose entries are  $\pm 1$  with their signs determined by the relevant vertices' communities that has eigenvalue  $\frac{a-b}{2}$ , and all of its other eigenvalues are 0.

Now, let  $M'$  be the graph's actual adjacency matrix. The result above suggests that the second eigenvector of  $M'$  may have entries that are correlated with the vertices' communities. The problem with this reasoning is that while  $M'$  has an expected value of  $M$ ,  $(M')^2$  has an expected value of roughly  $M^2 + \frac{a+b}{2}I$  because for every  $i, j$ ,  $M'_{i,j} = 1 \iff M'_{j,i} = 1$ , with the result that

$E[\sum_j M'_{i,j} \cdot M'_{j,i}]$  is very different from  $\sum_j E[M'_{i,j}] \cdot E[M'_{j,i}]$ . In other words, the square of the adjacency matrix counts walks of length 2 and has an expected value that is significantly different from the square of the expected adjacency matrix due to backtracking.

In order to avoid this issue, we use the graph's nonbacktracking walk matrix  $W$ , which was defined in [14] as a matrix over the vector space with an orthonormal basis consisting of a vector for each directed edge in the graph.  $W_{(v_1,v_2),(v'_1,v'_2)}$  is defined to be 1 if  $v'_2 = v_1$  and  $v_2 \neq v'_1$  and 0 otherwise. In other words, it has a 1 for every case where one directed edge leads to another that is not the same edge in the other direction.

Now, let  $w \in R^{2|E(G)|}$  be the vector whose entries are all 1, and  $w' \in R^{2|E(G)|}$  be the vector such that  $w'_{(v_0,v_1)}$  is 1 if  $v_0$  is in community 1 and  $-1$  if  $v_0$  is in community 2. As mentioned before, for a small  $t$  and a random  $(v, v') \in E(G)$ , there will be an average of approximately  $(\frac{a+b}{2})^t$  directed edges  $t$  edges in front of  $(v, v')$ , and approximately  $(\frac{a-b}{2})^t$  more of these edges will have ending vertices in the same community as  $v'$  than in the other community on average. So,  $w \cdot W^t w \approx 2|E(G)|(\frac{a+b}{2})^t$  and  $w' \cdot W^t w' \approx 2|E(G)|(\frac{a-b}{2})^t$ . That strongly suggests that  $W$  has eigenvectors that are correlated with  $w$  and  $w'$  that have eigenvalues of approximately  $\frac{a+b}{2}$  and  $\frac{a-b}{2}$  respectively. It also seems plausible that  $W$ 's other eigenvalues have relatively small magnitudes.

If this is true, then one can gain information on which vertices of  $G$  are in each community from the second eigenvector of  $W$ . One could simply calculate the second eigenvector of  $W$  directly. However, it is significantly faster to pick a random vector  $w''$  and then compute  $W^m w''$  for some suitable  $m$ . The resulting vector will be approximately a linear combination of  $W$ 's main eigenvectors. Unfortunately, it will be much closer to being a multiple of its first eigenvector than its second. If we multiply  $(W - \frac{a+b}{2}I)$  by the resulting vector, such as in (4.1), the component of the vector that is proportional to the first eigenvector will be mostly cancelled out. However, since its eigenvalue is not exactly  $\frac{a+b}{2}$ , it will not be cancelled out completely, and might still be too large. Luckily, if we instead multiply  $(W - \frac{a+b}{2}I)^{m'}$  by the resulting vector for suitable  $m'$ , such as in (4.2), the component of the vector that is proportional to the first eigenvalue will be essentially cancelled out, leaving a vector that is approximately a multiple of  $W$ 's second eigenvector, and thus correlated with  $G$ 's communities. We believe that this would succeed in detecting communities in the SBM, but in order to make it easier to prove that our algorithm works we actually use  $r$ -nonbacktracking walks. This corresponds to using the graph's  $r$ -nonbacktracking walk matrix, which is defined as follows.

**Definition 11.** For any  $r$ , the graph's  $r$ -nonbacktracking walk matrix,  $W^{(r)}$ , is a matrix over the

vector space with an orthonormal basis consisting of a vector for each directed path of length  $r - 1$  on the graph.  $W_{(v_1, v_2, \dots, v_r), (v'_1, v'_2, \dots, v'_r)}^{(r)}$  is 1 if  $v'_{i+1} = v_i$  for each  $1 \leq i < r$  and  $v'_1 \neq v_r$ , otherwise it is 0. In other words,  $W^{(r)}$  maps a path of length  $r - 1$  to the sum of all paths resulting from adding another element to the end of the path and deleting its first element.

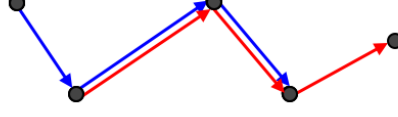


Figure 4.2: These two paths (blue and red) lead to an entry of 1 in the  $W^{(4)}$  matrix.

Performing these calculations is essentially what the acyclic belief propagation algorithm does. From this perspective, the algorithm roughly translates to:

1. Choose  $y^{(1)}$  randomly such that each element is independently drawn from a Normal distribution.
2. For each  $1 < t \leq m$ , let  $y^{(t)} = W^{(r)} y^{(t-1)}$ .
3. Change  $y^{(m)}$  to  $(W^{(r)} - \frac{a+b}{2} I)^{m'} y^{(m-m')}$ , where  $m' = \lceil \frac{m-3r-1}{l} \rceil$ .
4. Set<sup>4</sup>  $y'_v = \sum_{v': (v', v) \in E(G)} y_{v, v'}^{(m)}$  for every  $v \in G$ . Return  $(\{v : y'_v > 0\}, \{v : y'_v \leq 0\})$ .

Even though  $r = 2$  might suffice to achieve the KS threshold in the SBM, the use of larger  $r$  might help for other graph models, e.g., having more short cycles.

#### 4.1.7 ABP for the general SBM

Now, consider a graph  $G$  drawn from  $SBM(n, p, Q/n)$  with arbitrary  $p$  and  $Q$ . Also, let  $\lambda_1, \lambda_2, \dots, \lambda_h$  be the distinct eigenvalues of  $PQ$  in order of nonincreasing magnitude. If the parameters are such that vertices from different communities have different expected degrees, then one can detect communities by simply dividing its vertices into those with above-average degrees and those with below-average degrees. So, assume that the expected degree of a vertex is independent of its community. Detecting communities in the general case runs into some obstacles that do not apply in the 2-community symmetric case. First of all, it is much less clear that assigning vertices to sets randomly is a useful start. Also, even if we did have reasonable preliminary guesses of which community

<sup>4</sup>Proceed as in step 6 of 2CS-ABP\* for the proof.

each vertex was in, it is not obvious how to determine a vertex's community based on the alleged communities of the vertices a fixed distance from it.

For the moment, assume that for each vertex  $v$ , we have a vector  $x_v$  such that we believe  $v$  is in community  $i$  with probability  $p_i + x_v \cdot e_i$  for each  $i$ , where all elements of  $x_v$  are small. Furthermore, assume that  $x_v$  is generated independently of  $v$ 's neighbors. The correct belief about the probability that  $v$  is in each community once its neighbors are taken into account is

$$p + x_v + \frac{1}{\lambda_1} \sum_{v': (v, v') \in E[G]} PQx_{v'}$$

up to nonlinear terms in the  $x$ 's. So, given  $m$  small enough that the set of vertices within  $m$  edges of  $v$  is a tree, the correct belief about what community  $v$  is in once all of the vertices within  $m$  edges of  $v$  are taken into account is

$$p + \sum_{0 \leq m' \leq m} \lambda_1^{-m'} \sum_{v': d(v, v') = m'} (PQ)^{m'} x_{v'}$$

up to nonlinear terms in the  $x$ 's. So, the logical belief about the probability that  $v$  is in each community based only on the preliminary guesses concerning the vertices  $m$  edges away from  $v$  is

$$p + \lambda_1^{-m} \sum_{v': d(v, v') = m} (PQ)^m x_{v'}.$$

Conveniently, this expression is linear, so if  $w$  is an eigenvector of  $PQ$  with eigenvalue  $\lambda_i$  for  $i \neq 1$ , then

$$\begin{aligned} E[w \cdot P^{-1} e_{\sigma_v}] &\approx w \cdot P^{-1} \left( p + \lambda_1^{-m} \sum_{v': d(v, v') = m} (PQ)^m x_{v'} \right) \\ &= \lambda_1^{-m} \lambda_i^m \sum_{v': d(v, v') = m} w \cdot P^{-1} x_{v'}. \end{aligned}$$

In particular, this means that we only need an initial estimate for  $w \cdot P^{-1} e_{\sigma_v}$ , for every vertex in the graph, rather than needing a full set of beliefs about the vertices' communities. Any random guesses we make will probably have correlation  $\pm \Omega(1/\sqrt{n})$  with  $w \cdot P^{-1} e_{\sigma_v}$ , so we can use them as a starting point.

Unfortunately, just like in the two-community symmetric case, the graph will run out of vertices before  $m$  becomes large enough to amplify our beliefs enough. However, switching from a sum over

all vertices  $v'$  that are  $m$  edges away from  $v$  to a sum over all nonbacktracking walks of length  $m$  ending in a vertex  $v'$  fixes this problem the same way it does in the two-community symmetric case. Likewise, we can still compute this sum by randomly dividing  $G$ 's vertices between two sets  $S_1$  and  $S_2$  and then using the following algorithm.

- 
1. For every  $(v, v') \in E(G)$  :
    - If  $v' \in S_2$ , set  $y_{v,v'}^{(1)} = 1$
    - Otherwise, set  $y_{v,v'}^{(1)} = -1$
  2. For every  $1 < t \leq m$  and  $(v, v') \in E(G)$  :
    - Set  $y_{v,v'}^{(t)} = \sum_{v'' : (v'', v') \in E(G), v'' \neq v} y_{v,v''}^{(t-1)}$
  3. For every  $(v, v') \in E(G)$  and  $v \in G$ 
    - Set  $y_v^{(m)} = \sum_{v' : (v, v') \in E(G)} y_{v,v'}^{(m)}$
- 

However, needing to compensate for the average value is a special case of a considerably more complicated phenomenon. The average value over all  $(v, v') \in E(G)$  of  $y_{v,v'}^{(1)} \cdot w_{\sigma_v}$  will typically have a magnitude of  $\Theta(1/\sqrt{n})$ , and for general  $t$  the average value over all  $(v, v') \in E(G)$  of  $y_{v,v'}^{(t)} \cdot w_{\sigma_v}$  will typically have a magnitude of  $\Theta(|\lambda_i^t|/\sqrt{n})$ . Now, let  $w'$  be an eigenvector of  $PQ$  with an eigenvalue of  $\lambda_{i'}$  which has greater magnitude than  $\lambda_i$ . Then the average value over all  $(v, v') \in E(G)$  of  $y_{v,v'}^{(t)} \cdot w'_{\sigma_v}$  will typically have a magnitude of  $\Theta(|\lambda_{i'}^t|/\sqrt{n})$ . Since this grows faster than  $y_{v,v'}^{(t)} \cdot w_{\sigma_v}$  does, it will eventually become large enough to disrupt efforts to estimate  $w_{\sigma_v}$  using  $y_{v,v'}^{(t)}$  the same way the average value of  $y^{(t)}$  did in the 2-community symmetric case. In fact, the issue with the average value is just the subcase of this when  $i' = 1$ . So, in order to deal with this, we need to compensate for each eigenvalue,  $\lambda_{i'}$ , of  $PQ$  with magnitude greater than  $\lambda_i$  by choosing several indices  $t_{0,i'}, t_{1,i'}, \dots, t_{m',i'}$  and redefining  $y_{v,v'}^{(t_{j,i'})}$  for each  $j$  so that

$$y_{v,v'}^{(t_{j,i'})} = -\lambda_{i'} y_{v,v'}^{(t_{j,i'}-1)} + \sum_{v'' : (v'', v') \in E(G), v'' \neq v} y_{v',v''}^{(t_{j,i'}-1)}$$

for every  $(v, v') \in E(G)$ . Assuming that this is done,  $y_{v,v'}^{(1)}$  has a variance of approximately 1, and then  $y_{v,v'}^{(t)}$  has a variance of roughly  $\lambda_1^t$ . It becomes possible to determine which community  $v$  is in with accuracy nontrivially greater than that obtained by guessing randomly based on  $y_{v,v'}^{(t)}$  when the expected difference between its values for  $v$  in different communities is within a constant factor of its standard deviation. In other words,  $t$  needs to be large enough that  $|\lambda_i^t|/\sqrt{n}$  is significant relative

to  $\sqrt{\lambda_1^t}$ . If  $\lambda_1 \geq \lambda_i^2$  then this will never happen, so the algorithm requires that

$$\lambda_i^2 > \lambda_1.$$

The general acyclic belief propagation algorithm is almost the same as the 2-community symmetric version. However, it takes a list of eigenvectors as input instead of just  $\lambda_1$ . Also, the step compensating for larger eigenvalues is changed from “Set  $y^{(m)} = Y M^{\lceil \frac{m-3r-1}{t} \rceil} e_m$  where  $M$  is the matrix such that  $M_{i,i} = 1$  for all  $i$ ,  $M_{i,i+1} = -(1-\gamma)\lambda_1$  for all  $i$  and all other entries of  $M$  are 0” to “Set  $y^{(m)} = Y \prod_{s' < s} M_{s'}^{\lceil \frac{m-r-(2r+1)s'}{t} \rceil} e_m$  where  $M_{s'}$  is the matrix such that  $(M_{s'})_{i,i} = 1$  for all  $i$ ,  $(M_{s'})_{i,i+1} = -(1-\gamma)\lambda_{s'}$  for all  $i$  and all other entries of  $M_{s'}$  are 0” Its effectiveness is described by the following theorem.

**Theorem 8.** *Let  $p \in (0,1)^k$  with  $\sum p = 1$ ,  $Q$  be a symmetric matrix with nonnegative entries,  $P$  be the diagonal matrix such that  $P_{i,i} = p_i$ , and  $\lambda_1, \dots, \lambda_h$  be the eigenvalues of  $PQ$  in order of nonincreasing magnitude. If  $\lambda_2^2 > \lambda_1$  then there exist constants  $\epsilon, r, c$  and  $m = \Theta(\log(n))$  such that when the acyclic belief propagation algorithm is run on these parameters and a random  $G \in SBM(n, p, Q/n)$ , with probability  $1 - o(1)$  there exist  $\sigma$  and  $\sigma'$  such that the difference between the fraction of vertices from community  $\sigma$  that are in  $S_1$  and the fraction of vertices from community  $\sigma'$  that are in  $S_1$  is at least  $\epsilon$ . The algorithm can be run in  $O(n \log n)$  time.*

**Remark 9.** *Let  $s = 3$  if  $|\lambda_2| = |\lambda_3|$  and  $s = 2$  otherwise. This theorem could alternately have stated that there exists  $\epsilon$  such that for any two communities  $\sigma$  and  $\sigma'$  such that there exists an eigenvector  $w$  of  $PQ$  with eigenvalue  $\lambda_s$  such that  $w_\sigma \neq w_{\sigma'}$ , the expected difference between the fraction of vertices from community  $\sigma$  that are in  $S_1$  and the fraction of vertices from community  $\sigma'$  that are in  $S_1$  is at least  $\epsilon$ .*

#### 4.1.8 Alternatives

There are also a couple of other variants of these ideas that may be useful for community detection. For instance, if we pick  $r \approx \ln(\ln(n))$  and then define  $\Sigma$  to be the  $n \times n$  symmetric matrix such that  $\Sigma_{v,v'}$  is the number of nonbacktracking walks of length  $r$  between  $v$  and  $v'$ , we suspect that  $\Sigma$ 's eigenvector of second largest magnitude will have entries that are correlated with the corresponding vertices' communities. Like in the standard case, we expect that we could get an approximation of this eigenvector by taking a random vector  $w$  and then computing  $(\Sigma - \lambda'_1 I)^{m'} \Sigma^{m-m'} w$  for suitable  $m$  and  $m'$  where  $\lambda'_1$  is an estimate of  $\Sigma$ 's largest eigenvalue.

We can compute  $\Sigma$  as follows. First, let  $\Sigma^{(t)}$  be the  $n \times n$  matrix such that  $\Sigma_{v,v'}^{(t)}$  is the number of nonbacktracking walks of length  $t$  between  $v$  and  $v'$ . Then  $\Sigma^{(0)} = I$ ,  $\Sigma^{(1)}$  is the graph's adjacency matrix, and  $\Sigma_{v,v'}^2$  is equal to the number of shared neighbors  $v$  and  $v'$  have for all  $v$  and  $v'$ . For every  $t > 2$ , we have that  $\Sigma^{(t)} = \Sigma^{(1)} \cdot \Sigma^{(t-1)} - D \cdot \Sigma^{(t-2)}$ , where  $D$  is the diagonal matrix such that  $D_{v,v}$  is one less than the degree of  $v$  for all  $v$ . This can be used to efficiently compute  $\Sigma = \Sigma^{(r)}$ .

Also, instead of prohibiting repeating a vertex within  $r$  steps, we could address the issue of tangles by dividing  $G$ 's edges between sets  $E_0, \dots, E_{m'}$  for suitable  $m'$  such that most of the edges are assigned to  $E_0$  and the rest are assigned to one of the others at random. Then we count nonbacktracking walks with the restriction that edge  $r$  of the walk must be from  $E_1$ , edge  $2r$  must be from  $E_2$  and so on, while all other edges must be from  $E_0$  for suitable  $r$ . The periodic prohibitions on using edges from  $E_0$  would force the walk to leave any tangle it had been in, while the fact that most of the edges are chosen from  $E_0$  prevents the restriction from reducing the number of walks too severely.

## 4.2 ABP proof technique

In order to prove that ABP works, we use the following key ideas. First, we note that for each  $v$  and  $t$ ,  $Y_{v,t}$  is equal to the sum over all  $r$ -nonbacktracking walks of length  $t$  ending at  $v$  of the values of  $x$  corresponding to their starting vertices. So, we can view each series of  $m+1$  vertices as making some contribution to  $y_v^{(m)}$ , where serieses that do not correspond to  $r$ -nonbacktracking walks make a contribution of 0. Next, we define  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  as a weighted sum of the contribution to  $y_v^{(m)}$  made by  $(v_0, \dots, v_m)$  and the contributions made by some of the subserieses of  $(v_0, \dots, v_m)$  and show that with appropriate values of  $(c_0, \dots, c_m)$ , it will be the case that

$$y_v^{(m)} = \sum_{v_0, \dots, v_m \in G: v_m = v} x_{v_0} \cdot W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$$

In order to prove that  $y_v^{(m)}$  behaves the way we need it to, we need to show that the average value over all  $v$  in a given community of  $y_v^{(m)}$  depends on the community in a certain way, and that the variance of  $y_v^{(m)}$  within a community is not excessively large. Doing that requires us to be able to bound the expected values of expressions of the forms  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  and  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m)) \cdot W_{(c_0, \dots, c_m)[r]}((v'_0, \dots, v'_m))$ . We reduce the later case to the former by

showing that if we add new vertices  $u_1, \dots, u_r$  to the graph that are adjacent to everything then

$$\begin{aligned} & W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m)) \cdot W_{(c_0, \dots, c_m)[r]}((v'_0, \dots, v'_m)) \\ &= W_{(c_0, \dots, c_m, 0, \dots, 0, c_m, c_{m-1}, \dots, c_0)[r]}((v_0, \dots, v_m, u_1, \dots, u_r, v'_m, v'_{m-1}, \dots, v'_0)) \end{aligned}$$

It is easy to calculate the expected value of  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  when  $v_0, \dots, v_m$  are all distinct, and to show that the total contribution of these terms to the expected value of  $y_v^{(m)}$  has the desired value. That leaves the task of showing that the total contribution of terms for which  $v_0, \dots, v_m$  repeats vertices is not too large.

This bound is more complicated to prove. First, we prove a bound on the expected value of any expression of the form  $W_{(c_0, \dots, c_{m'})[r]}((v_0, \dots, v_{m'}))$  which satisfies a certain property about how the indices of nonzero elements of  $(c_0, \dots, c_{m'})$  relate to the indices of elements of  $(v_0, \dots, v_{m'})$  that appear more than once on the list. Then, we show that any expression of the form  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  can be expressed as a sum of expressions satisfying the previous properties. Next, we argue that there are very few sequences of vertices  $(v_0, \dots, v_{m'})$  that repeat in relatively complicated ways while the expected value of  $W_{(c_0, \dots, c_{m'})[r]}((v_0, \dots, v_{m'}))$  if  $(v_0, \dots, v_{m'})$  repeats vertices in a simple way is relatively small. Finally, we combine all of this in order to prove the desired bounds.

## Chapter 5

# The degree profiling algorithm

With the tools to perform almost exact recovery developed in the previous section, we are in a good position to complete the full recovery of the clusters. Note that as discussed in the introduction, while the exact recovery requirement may be viewed as a strong requirement, it benefits from a sharp phase transition as opposed to almost exact recovery, and hence can be used to gain insight on algorithms (with the threshold as a clear benchmark) and on relevant information measures for community detection.

Our analysis of the possibility or impossibility of complete recovery relies on the fact that the probability distribution of the numbers of neighbors a given vertex has in each community is essentially a multivariable poisson distribution. We hence investigate a hypothesis problem (see Section 6.3.2), where a node in the SBM graph with known<sup>1</sup> clusters is taken and re-classified based on its degree profile, i.e., on the number of neighbors it has in each community. Depending on the parameters of the SBM, the error probability of the optimal test depends on the degree of overlap between two multivariable poisson distributions, which we show is either  $o(\frac{1}{n})$  or  $\omega(\frac{1}{n})$  depending on how the communities' corresponding columns in  $PQ$  compare. This is where the CH-divergence emerges as the exponent for the test error probability. This gives in particular a new sharp estimate for the tail event of Poisson measures.

If the error probability is  $\omega(\frac{1}{n})$  then any method of distinguishing between vertices in those two communities must fail with probability  $\omega(\frac{1}{n})$ , so any possible algorithm attempting to distinguish between them must misclassify at least one vertex with probability  $1 - o(1)$ . On the other hand, if the degree of overlap between all communities we are trying to distinguish between is  $o(1/n)$  then

---

<sup>1</sup>Note that the clusters are not exactly known in our case, since the algorithm of previous section give an  $o(n)$  fraction of wrong labels to be mitigated.

with probability  $1 - o(1)$  one could correctly classify any vertex in the graph if one knew what community each of its neighbors was in. There exists  $\delta$  such that attempting to classify a vertex based on classifications of its neighbors that are wrong with probability  $x$  results in a probability of misclassifying the vertex that is only  $n^{\delta x}$  times as high as it would be if they were all classified correctly. Based on this, the obvious approach to exact recovery would be to use a partial recovery algorithm to create a preliminary classification and then attempt to determine which family of communities each vertex is in based on its neighbors' alleged communities. However, the standard partial recovery algorithm has a constant error rate, so this procedure's output would have an error rate  $n^c$  times as large as if each vertex were being classified based on its neighbors' true communities for some  $c > 0$ . If the degrees of overlap are only barely below  $1/n$  then this would increase the error rate enough that this procedure would misclassify at least one vertex with high probability.

Instead, we go through three successively more accurate classifications as follows. Given a partial reconstruction of the communities with an error rate that is a sufficiently low constant, one can classify vertices based on their neighbors' alleged communities with an accuracy of  $1 - O(n^{-c})$  for some constant  $c > 0$ . Then one can use this classification of a vertex's neighbors to determine which family of communities it is in with an accuracy of  $1 - o(\frac{1}{n} \cdot n^{\delta c' n^{-c}}) = 1 - o(1/n)$ . Therefore, the resulting classification is correct with probability  $1 - o(1)$ . Overall, the algorithm for exact recovery works as follows.

**The Degree-profiling algorithm.** The inputs are  $(G, \gamma)$ , where  $G$  is a graph, and  $\gamma \in [0, 1]$  (see Theorem 11 for how to set  $\gamma$ ).

The algorithm outputs an assignment of each vertex to a community. It does the following:

- (1) Define the graph  $g'$  on the vertex set  $[n]$  by selecting each edge in  $g$  independently with probability  $\gamma$ , and define the graph  $g''$  that contains the edges in  $g$  that are not in  $g'$ .
- (2) Run **Sphere-comparison** on  $g'$  to obtain the preliminary classification  $\sigma' \in [k]^n$  (see Section 2.3.)
- (3) Determine the edge density between each pair of alleged communities, and use this information and the alleged communities' sizes to attempt to identify the communities up to symmetry.
- (4) For each node  $v \in [n]$ , determine which community node  $v$  is most likely to belong to based on its degree profile computed from the preliminary classification  $\sigma'$  (see Section 6.3.2), and call it  $\sigma''_v$ .
- (5) For each node  $v \in [n]$ , determine which community node  $v$  is most likely to belong to based on its degree profile computed from the preliminary classification  $\sigma''$  (see Section 6.3.2).

## Chapter 6

# Proofs

### 6.1 Information-theoretic community detection

For this section, we will focus on the symmetric stochastic block model, which is defined as follows.

**Definition 12.** Given  $\alpha, \beta > 0$  and a positive integer  $k$ ,  $\text{SSBM}(n, k, \alpha, \beta) = \text{SBM}(n, p, W)$  where  $p \in [0, 1]^k$  is defined so that  $p_i = 1/k$  for all  $i$  and  $W$  is the  $k \times k$  matrix such that  $W_{i,i} = \alpha$  for all  $i$  and  $W_{i,j} = \beta$  for all  $i \neq j$ .

In other words,  $\text{SSBM}(n, k, \alpha, \beta)$  is equally likely to assign a vertex to any community, vertices in the same community have an edge between them with probability  $\alpha$ , and vertices in different communities have an edge between them with probability  $\beta$ .

For  $x \in [k]^n$  and  $\varepsilon > 0$ , define the set of bad clusterings with respect to  $x$  as

$$B_\varepsilon(x) = \{y \in [n]^k : \frac{1}{n}d_*(x, y) > 1 - \frac{1}{k} - \varepsilon\}, \quad (6.1)$$

where  $d_*(x, y) = \min_{\pi \in S_k} d_H(x, \pi(y))$ ,  $d_H$  is the Hamming distance and  $\pi(y)$  denotes the application of  $\pi$  to each component of  $y$ .

Recall that

$$\text{Bal}(n, k, \varepsilon) = \{x \in [k]^n : \forall i \in [k], |\{u \in [n] : x_u = i\}|/n \in [1/k - \varepsilon, 1/k + \varepsilon]\}, \quad (6.2)$$

and

$$T_\delta(G) = \{x \in \text{Bal}(n, k, \delta) : \\ \sum_{i=1}^k |\{G_{u,v} : (u, v) \in \binom{[n]}{2} \text{ s.t. } x_u = i, x_v = i\}| \geq \frac{an}{2k}(1 - \delta), \\ \sum_{i,j \in [k], i < j} |\{G_{u,v} : (u, v) \in \binom{[n]}{2} \text{ s.t. } x_u = i, x_v = j\}| \leq \frac{bn(k-1)}{2k}(1 + \delta)\}.$$

### 6.1.1 Atypicality of a bad clustering

**Lemma 1.** *Let  $\varepsilon > 0$ ,  $x, y \in \text{Bal}(n, k, \delta)$  such that  $y \in B_\varepsilon(x)$ , and  $(\sigma, G) \sim \text{SSBM}(n, k, a/n, b/n)$ . Then,*

$$\mathbb{P}\{y \in T_\delta(G) | \sigma = x\} \leq \exp\left(-\frac{n}{k}A(\varepsilon, \delta)\right)$$

where  $A(\varepsilon, \delta)$  is continuous at  $(\varepsilon, \delta) = (0, 0)$  and

$$A(0, 0) = \frac{a + b(k-1)}{2} \ln \frac{k}{(a + (k-1)b)} + \frac{a}{2} \ln a + \frac{b(k-1)}{2} \ln b.$$

Recall that  $d := \frac{a + (k-1)b}{k}$ .

**Corollary 2.** *Detection is solvable in  $\text{SSBM}(n, k, a/n, b/n)$  if*

$$\frac{1}{2 \ln k} \left( \frac{a \ln a + (k-1)b \ln b}{k} - d \ln d \right) > 1.$$

Note that for  $a = 0$ , the above bound for  $b$  is  $\Theta(\ln k/k)$  times the bound given by the KS threshold. Further, this improves on the KS threshold for  $k \geq 5$ . However, for  $b = 0$ , the above bound gives  $a > 2k$ , which is worse than the KS threshold  $a > k$ . For the same reasons, it is loose for  $k = 2$  and  $a = 0$ . In the next section, we improve the bound to capture the right behaviour at the extremal regimes.

*Proof of the Corollary 2.* Let  $(\sigma, G) \sim \text{SSBM}(n, k, a/n, b/n)$  where  $k, a, b$  satisfy

$$\frac{1}{2 \ln k} \left( \frac{a \ln a + (k-1)b \ln b}{k} - d \ln d \right) > 1. \tag{6.3}$$

Let  $\hat{\sigma}_\delta(G)$  be uniformly drawn in  $T_\delta(G)$ . We have

$$\mathbb{P}\{\hat{\sigma}_\delta(G) \in B_\epsilon(\sigma)\} = \mathbb{P}\{\hat{\sigma}_\delta(G) \in B_\epsilon(\sigma) \wedge \sigma \in \text{Bal}(n, k, \delta)\} + o(1) \quad (6.4)$$

$$= \mathbb{E} \frac{|T_\delta(G) \cap B_\epsilon(\sigma)| \mathbb{1}(\sigma \in \text{Bal}(n, k, \delta))}{|T_\delta(G)|} + o(1) \quad (6.5)$$

$$\leq \mathbb{E}|T_\delta(G) \cap B_\epsilon(\sigma)| \mathbb{1}(\sigma \in \text{Bal}(n, k, \delta)) + o(1) \quad (6.6)$$

where we use the fact that  $|T_\delta(G)| \geq 1$  with high probability, since  $\sigma \in T_\delta(G)$  with high probability, and  $\sigma \in \text{Bal}(n, k, \delta)$  with high probability. Moreover,

$$\mathbb{P}\{\hat{\sigma}_\delta(G) \in B_\epsilon(\sigma)\} \leq \mathbb{E} \sum_{y \in [k]^n} \mathbb{1}(y \in T_\delta(G), y \in B_\epsilon(\sigma), \sigma \in \text{Bal}(n, k, \delta)) + o(1) \quad (6.7)$$

$$\leq \sum_{y \in [k]^n} \sum_{x \in \text{Bal}(n, k, \delta)} \mathbb{P}\{y \in T_\delta(G), y \in B_\epsilon(\sigma) | \sigma = x\} \mathbb{P}\{\sigma = x\} + o(1) \quad (6.8)$$

$$= \sum_{y \in [k]^n} \sum_{x \in \text{Bal}(n, k, \delta): y \in B_\epsilon(x)} \mathbb{P}\{y \in T_\delta(G) | \sigma = x\} \mathbb{P}\{\sigma = x\} + o(1). \quad (6.9)$$

Letting  $\tau$  be the bound obtained from Lemma 1 on  $\mathbb{P}\{y \in T_\delta(G) | \sigma = x\}$ , we have

$$\mathbb{P}\{\hat{\sigma}_\delta(G) \in B_\epsilon(\sigma)\} \leq k^n \tau + o(1). \quad (6.10)$$

We can now take  $\delta > 0$  such that for a strictly positive  $\varepsilon$ , (6.3) implies that (6.10) vanishes.  $\square$

*Proof of Lemma 1.* Assume that  $a \geq b$  (the other case is treated similarly). Let  $(\sigma, G) \sim \text{SSBM}(n, k, a, b)$  and  $x, y \in \text{Bal}(n, k, \delta)$  such that  $y \in B_\epsilon(x)$ . Denote by  $T_\delta^{(in)}(G)$  and  $T_\delta^{(out)}(G)$  the sets of clusterings having typical fraction of edges inside and across clusters respectively. We have

$$\mathbb{P}\{y \in T_\delta(G) | \sigma = x\} = \mathbb{P}\{y \in T_\delta^{(in)}(G) | \sigma = x\} \mathbb{P}\{y \in T_\delta^{(out)}(G) | \sigma = x\}, \quad (6.11)$$

with

$$\begin{aligned} & \mathbb{P}\{y \in T_\delta^{(in)}(G) | \sigma = x\} \\ &= \mathbb{P}\left\{ \sum_{i=1}^k |\{G_{u,v} : (u,v) \in \binom{[n]}{2} \text{ s.t. } y_u = i, y_v = i\}| \geq \frac{an}{2k}(1-\delta) | \sigma = x \right\} \\ &\leq \mathbb{P}\left\{ \text{Bin}\left(\frac{n(n-1)}{2k^2}(1+\xi), \frac{a}{n}\right) + \text{Bin}\left(\frac{(k-1)n(n-1)}{2k^2}(1+\xi), \frac{b}{n}\right) \geq \frac{an}{2k}(1-\delta) \right\} \end{aligned}$$

$$\begin{aligned}
& \mathbb{P}\{y \in T_\delta^{(out)}(G) | \sigma = x\} \\
&= \mathbb{P}\left\{ \sum_{i,j \in [k], i < j} |\{G_{u,v} : (u,v) \in \binom{[n]}{2} \text{ s.t. } y_u = i, y_v = j\}| \leq \frac{bn(k-1)}{2k}(1+\delta) | \sigma = x \right\} \\
&\leq \mathbb{P}\left\{ \text{Bin}\left(\frac{n^2(k-1)}{2k^2}(1-\xi), \frac{a}{n}\right) + \text{Bin}\left(\frac{n^2(k-1)^2}{2k^2}(1-\xi), \frac{b}{n}\right) \leq \frac{bn(k-1)}{2k}(1+\delta) \right\},
\end{aligned}$$

where  $\text{Bin}(N, p)$  denotes a Binomial random variable with  $N$  trials of bias  $p$ , all Binomial random variables are independent, and  $\xi \leq 2k^2(\varepsilon + \delta)$  (the latter gives a loose bound on the offset of the trial counts in the Binomials due to the fact that  $x, y$  are not exactly but approximately balanced).

Since the dependence in  $\varepsilon, \delta$  is continuous, we next proceed without the  $\xi$  terms to prove the lemma.

We have

$$\begin{aligned}
& \mathbb{P}\left\{ \text{Bin}\left(\frac{n(n-1)}{2k^2}, \frac{a}{n}\right) + \text{Bin}\left(\frac{(k-1)n(n-1)}{2k^2}, \frac{b}{n}\right) \geq \frac{an}{2k} \right\} \\
&\leq O(n^2) \max_{cn \in [0, \frac{n^2}{2k^2}]} \mathbb{P}\left\{ \text{Bin}\left(\frac{n^2}{2k^2}, \frac{a}{n}\right) = cn \right\} \mathbb{P}\left\{ \text{Bin}\left(\frac{(k-1)n^2}{2k^2}, \frac{b}{n}\right) = \frac{an}{2k} - cn \right\}
\end{aligned}$$

and from (6.16),

$$\begin{aligned}
& \mathbb{P}\left\{ \text{Bin}\left(\frac{n^2}{2k^2}, \frac{a}{n}\right) = cn \right\} = \exp\left(-\frac{n}{k} \left(\frac{a}{2k} + kc \ln \frac{2k^2 c}{ea}\right) + o(n)\right), \\
& \mathbb{P}\left\{ \text{Bin}\left(\frac{(k-1)n^2}{2k^2}, \frac{b}{n}\right) = \frac{an}{2k} - cn \right\} \\
&= \exp\left(-\frac{n}{k} \left(\frac{(k-1)b}{2k} + (a/2 - ck) \ln \frac{2k(a/2 - ck)}{eb(k-1)}\right) + o(n)\right)
\end{aligned}$$

and

$$\begin{aligned}
& \mathbb{P}\left\{ \text{Bin}\left(\frac{n^2}{2k^2}, \frac{a}{n}\right) = cn \right\} \mathbb{P}\left\{ \text{Bin}\left(\frac{(k-1)n^2}{2k^2}, \frac{b}{n}\right) = \frac{an}{2k} - cn \right\} \\
&= \exp\left(-\frac{n}{k} \left(\frac{a + (k-1)b}{2k} + \frac{a}{2} \ln \frac{2k(a/2 - ck)}{eb(k-1)} + kc \ln \frac{bc(k-1)k}{a(a/2 - ck)}\right) + o(n)\right) \\
&= \exp\left(-\frac{n}{k} \left(\frac{a + (k-1)b}{2k} + \frac{a}{2} \ln \frac{2k}{eb(k-1)} + \left(\frac{a}{2} - ck\right) \ln \left(\frac{a}{2} - ck\right) + kc \ln \frac{bc(k-1)k}{a}\right) + o(n)\right).
\end{aligned}$$

Note that the function

$$c \rightarrow (a/2 - ck) \ln(a/2 - ck) + kc \ln \frac{bc(k-1)k}{a}$$

has a single minimum at  $\frac{a^2}{2k(a+(k-1)b)}$ . Plugging in this value for  $c$  gives

$$\mathbb{P} \left\{ \text{Bin} \left( \frac{n^2}{2k^2}, \frac{a}{n} \right) = cn \right\} \mathbb{P} \left\{ \text{Bin} \left( \frac{(k-1)n^2}{2k^2}, \frac{b}{n} \right) = \frac{an}{2k} - cn \right\} \quad (6.12)$$

$$\leq \exp \left( -\frac{n}{k} \left( \frac{a+(k-1)b}{2k} + \frac{a}{2} \ln \frac{ak}{e(a+(k-1)b)} \right) \right). \quad (6.13)$$

In addition,

$$\begin{aligned} & \mathbb{P} \left\{ \text{Bin} \left( \frac{n^2(k-1)}{2k^2}, \frac{a}{n} \right) + \text{Bin} \left( \frac{n^2(k-1)^2}{2k^2}, \frac{b}{n} \right) \leq \frac{bn(k-1)}{2k} \right\}, \\ & \leq O(n^2) \\ & \cdot \max_{cn \in [0, \frac{n^2(k-1)}{2k^2}]} \mathbb{P} \left\{ \text{Bin} \left( \frac{n^2(k-1)}{2k^2}, \frac{a}{n} \right) = cn \right\} \mathbb{P} \left\{ \text{Bin} \left( \frac{n^2(k-1)^2}{2k^2}, \frac{b}{n} \right) = \frac{bn(k-1)}{2k} - cn \right\} \end{aligned}$$

and from (6.16),

$$\begin{aligned} & \mathbb{P} \left\{ \text{Bin} \left( \frac{n^2(k-1)}{2k^2}, \frac{a}{n} \right) = cn \right\} = \exp \left( -\frac{n}{k} \left( \frac{a(k-1)}{2k} + kc \ln \frac{2k^2c}{ea(k-1)} \right) + o(n) \right), \\ & \mathbb{P} \left\{ \text{Bin} \left( \frac{n^2(k-1)^2}{2k^2}, \frac{b}{n} \right) = \frac{bn(k-1)}{2k} - cn \right\} \\ & = \exp \left( -\frac{n}{k} \left( \frac{(k-1)^2b}{2k} + (b(k-1)/2 - ck) \ln \frac{(b(k-1) - 2ck)k}{eb(k-1)^2} \right) + o(n) \right) \end{aligned}$$

and

$$\begin{aligned} & \mathbb{P} \left\{ \text{Bin} \left( \frac{n^2(k-1)}{2k^2}, \frac{a}{n} \right) = cn \right\} \mathbb{P} \left\{ \text{Bin} \left( \frac{n^2(k-1)^2}{2k^2}, \frac{b}{n} \right) = \frac{bn(k-1)}{2k} - cn \right\} \\ & = \exp \left( -\frac{n}{k} \left[ \left( \frac{a(k-1)}{2k} + kc \ln \frac{2k^2c}{ea(k-1)} + \frac{(k-1)^2b}{2k} \right. \right. \right. \\ & \quad \left. \left. \left. + (b(k-1)/2 - ck) \ln \frac{(b(k-1) - 2ck)k}{eb(k-1)^2} \right) \right] + o(n) \right) \\ & = \exp \left( -\frac{n}{k} \left[ \left( \frac{(k-1)(a+(k-1)b)}{2k} + \frac{b(k-1)}{2} \ln \frac{(b(k-1) - 2ck)k}{eb(k-1)^2} \right. \right. \right. \\ & \quad \left. \left. \left. + kc \ln \frac{2cb(k-1)k}{a(b(k-1) - 2ck)} \right) \right] + o(n) \right) \\ & = \exp \left( -\frac{n}{k} \left[ \left( \frac{(k-1)(a+(k-1)b)}{2k} + \frac{b(k-1)}{2} \ln \frac{2k}{eb(k-1)^2} \right. \right. \right. \\ & \quad \left. \left. \left. + (b(k-1)/2 - ck) \ln(b(k-1)/2 - ck) + kc \ln \frac{cb(k-1)k}{a} \right) \right] + o(n) \right). \end{aligned}$$

Note that the function

$$c \rightarrow (b(k-1)/2 - ck) \ln(b(k-1)/2 - ck) + kc \ln \frac{cb(k-1)k}{a}$$

has a single minimum at  $\frac{(k-1)ab}{2k(a+(k-1)b)}$ , thus

$$\mathbb{P} \left\{ \text{Bin} \left( \frac{n^2}{2k^2}, \frac{a}{n} \right) = cn \right\} \mathbb{P} \left\{ \text{Bin} \left( \frac{(k-1)n^2}{2k^2}, \frac{b}{n} \right) = \frac{an}{2k} - cn \right\} \quad (6.14)$$

$$\leq \exp \left( -\frac{n}{k} \left( \frac{(k-1)(a+(k-1)b)}{2k} + \frac{b(k-1)}{2} \ln \frac{bk}{e(a+(k-1)b)} \right) + o(n) \right). \quad (6.15)$$

Combining (6.13) and (6.15), the result follows from algebraic manipulations.  $\square$

**Lemma 2** (Binomial estimates). *Let  $N$  be a positive integer and  $t, s > 0$ . Then,*

$$\mathbb{P}\{\text{Bin}(N^2, s/N) = \lfloor tN \rfloor\} = \exp \left( - \left( s - t + t \ln \frac{t}{s} \right) N + O(\ln N) \right), \quad (6.16)$$

$$\mathbb{P}\{\text{Bin}(N^2, s/N) \leq tN\} = \exp \left( - \left( s - t + t \ln \frac{t}{s} \right) N + O(\ln N) \right), \quad \text{if } s > t. \quad (6.17)$$

*Proof.* Stirling's approximation gives

$$N! = e^{N \ln N - N + O(\ln N)} \quad (6.18)$$

thus

$$\binom{N^2}{\lfloor tN \rfloor} = \exp \left( tN \ln N + tN \ln \frac{e}{t} + O(\ln N) \right) \quad (6.19)$$

and

$$\mathbb{P}\{\text{Bin}(N^2, s/N) = \lfloor tN \rfloor\} \quad (6.20)$$

$$= \binom{N^2}{\lfloor tN \rfloor} \left( \frac{s}{N} \right)^{\lfloor tN \rfloor} \left( 1 - \frac{s}{N} \right)^{N^2 - \lfloor tN \rfloor} \quad (6.21)$$

$$= \exp \left( tN \ln N + tN \ln \frac{e}{t} + O(\ln N) \right) \exp \left( -tN \ln N + tN \ln s - sN + O(\ln N) \right) \quad (6.22)$$

$$= \exp \left( \left( t \ln \frac{es}{t} - s \right) N + O(\ln N) \right), \quad (6.23)$$

and the second statement follows from the fact that the Binomial distribution is unimodal.  $\square$

### 6.1.2 Size of the typical set

The goal of this section is to lower bound the size of the typical set  $T_\delta(G)$ . Recall that the expected node degree in  $\text{SSBM}(n, k, a/n, b/n)$  is

$$d := \frac{a + (k-1)b}{k}.$$

**Theorem 9.** *Let  $T_\delta(G)$  denote the typical set of clusterings for  $G$  drawn under  $\text{SSBM}(n, k, a/n, b/n)$ .*

*Then, for any  $\varepsilon > 0$ ,*

$$\mathbb{P}\{|T_\delta(G)| < \max(k^{(\psi-\varepsilon)n}, 2^{(e^{-a/k}(1-(1-e^{-b/k})^{k-1})-\varepsilon)n})\} = o(1),$$

where

$$\begin{aligned} \psi := & \frac{\tau}{d} \left(1 - \frac{\tau}{2}\right) \\ & + \frac{1}{\ln(k)} \left( \frac{a}{a + (k-1)b} \ln \left( \frac{a + (k-1)b}{a} \right) + \frac{(k-1)b}{a + (k-1)b} \ln \left( \frac{a + (k-1)b}{b} \right) \right) \\ & \cdot \left( \frac{\tau^2}{2d} + (d - \tau)e^{-(d-\tau)} \right), \end{aligned}$$

and  $\tau = \tau_d$  is the unique solution in  $(0, 1)$  of

$$\tau e^{-\tau} = d e^{-d} \tag{6.24}$$

or equivalently  $\tau = \sum_{j=1}^{+\infty} \frac{j^{j-1}}{j!} (d e^{-d})^j$ .

As a first step to proving this theorem, we prove one half of the inequality.

**Lemma 3.** *Let  $T_\delta(G)$  denote the typical set of clusterings for  $G$  drawn under  $\text{SSBM}(n, k, a/n, b/n)$ .*

*Then, for any  $\varepsilon > 0$ ,*

$$\mathbb{P}\{|T_\delta(G)| < 2^{(e^{-a/k}(1-(1-e^{-b/k})^{k-1})-\varepsilon)n}\} = o(1).$$

*Proof.* Construct a clustering of  $G$ ,  $\sigma'$ , as follows. Start with  $\sigma' = \sigma$ . Then, go through the vertices of  $G$  in a random order, and for each  $v$  with no neighbor  $v'$  such that  $\sigma'_v = \sigma'_{v'}$ , change  $\sigma'_v$  to a random element of  $[k] \setminus \{\sigma'_{v'} : (v, v') \in E(G)\}$ .

Initially,  $\sigma' = \sigma$ , so the probability distributions of  $(G, \sigma)$  and  $(G, \sigma')$  are identical. For a given ordering of the vertices, if these probability distributions are identical immediately before  $\sigma'_v$  is changed, then for given values of  $G$  and  $\sigma' \setminus \sigma'_v$ , it is equally likely that  $\sigma'_v$  had each value

that it may be assigned in this step. So, the probability distribution is unchanged. Therefore, the probability distribution of  $(G, \sigma')$  is identical to the probability distribution of  $(G, \sigma)$  at any point in this algorithm. In particular,  $\sigma' \in T_\delta(G)$  with probability  $1 - o(1)$ .

Now, for each  $v \in G$ , let  $z_v$  be 1 if there is more than one option for  $\sigma'_v$  when it is time for the algorithm to assign it and 0 otherwise. A vertex has no neighbors in its own community with probability  $e^{-a/k} + o(1)$ , and no neighbors in any given other community with probability  $e^{-b/k} + o(1)$ . So, a vertex has no neighbors in its own community and at least one other community with probability  $e^{-a/k}(1 - (1 - e^{-b/k})^{k-1}) + o(1)$ . Thus, for each  $v$ ,

$$\mathbb{E}(z_v) = e^{-a/k}(1 - (1 - e^{-b/k})^{k-1}) + o(1).$$

Also, given two different vertices, they both have no neighbors in their own community and at least one other with probability  $e^{-2a/k}(1 - (1 - e^{-b/k})^{k-1})^2 + o(1)$ . Given any  $v$  and  $v'$ , assume without loss of generality that  $v$  is before  $v'$  in the random ordering. With probability  $1 - o(1)$ ,  $v'$  will not be in the last  $\sqrt{n}$  vertices, and at the time  $v'$  comes up in the ordering,  $e^{-a/k}(1 - (1 - e^{-b/k})^{k-1}) + o(1)$  of the remaining vertices will have no neighbors claimed to be in their communities and at least one other. So, symmetry between the vertices implies that the correlation between  $z_v$  and  $z_{v'}$  is  $o(1)$ . That means that with probability  $1 - o(1)$ , we have

$$\sum_{v \in G} z_v = e^{-a/k}(1 - (1 - e^{-b/k})^{k-1})n + o(n)$$

For fixed values of  $(G, \sigma)$  and any given  $\sigma'_0, z_0$ , we have that  $\mathbb{P}\{\sigma' = \sigma'_0, \sum_{v \in G} z_v \geq z_0\} \leq 2^{-z_0}$  because every time the algorithm has more than one option for  $\sigma'_v$ , there is at most a  $1/2$  chance that it sets  $\sigma'_v$  to  $(\sigma'_0)_v$ , and if  $\sum z_v \geq z_0$  then there are at least  $z_0$  times when it had such a choice. So, for a fixed graph  $G$ ,

$$\mathbb{P}\left\{\sigma' \in T_\delta(G), \sum_{v \in G} z_v > e^{-a/k}(1 - (1 - e^{-b/k})^{k-1})n - \epsilon n/2\right\} \quad (6.25)$$

$$\leq |T_\delta(G)| \cdot 2^{e^{-a/k}(1 - (1 - e^{-b/k})^{k-1})n - \epsilon n/2}. \quad (6.26)$$

We already know that with probability  $1 - o(1)$  the probability in question is  $1 - o(1)$ , so the conclusion holds.  $\square$

To prove the rest of this theorem, we need some topological properties of the SBM graph, analog

to the Erdős-Rényi case [13].

**Lemma 4.** *Let  $T_j$  denote the number of isolated  $j$ -trees (i.e., trees on  $j$  vertices) in  $\text{SSBM}(n, k, a/n, b/n)$  and  $M_j$  the number of edges contained in those trees. Then, for any  $\varepsilon > 0$ ,*

$$\begin{aligned}\mathbb{P}\{T_j/n \notin [\tau_j/d - \varepsilon, \tau_j/d + \varepsilon]\} &= o(1), \\ \mathbb{P}\{M_j/n \notin [(j-1)\tau_j/d - \varepsilon, (j-1)\tau_j/d + \varepsilon]\} &= o(1),\end{aligned}$$

where

$$\tau_j = \frac{j^{j-2}(de^{-d})^j}{j!}.$$

**Lemma 5.** *Let  $T$  denote the number of isolated trees in  $\text{SSBM}(n, k, a/n, b/n)$  and  $M$  the number of edges contained in those trees. Then, for any  $\varepsilon > 0$ ,*

$$\begin{aligned}\mathbb{P}\left\{T/n \notin \left[\frac{\tau}{d}\left(1 - \frac{\tau}{2}\right) - \varepsilon, \frac{\tau}{d}\left(1 - \frac{\tau}{2}\right) + \varepsilon\right]\right\} &= o(1), \\ \mathbb{P}\left\{M/n \notin \left[\frac{\tau^2}{2d} - \varepsilon, \frac{\tau^2}{2d} + \varepsilon\right]\right\} &= o(1),\end{aligned}$$

where  $\tau$  is defined in (6.24).

**Lemma 6.** [9] *Let  $Q$  denote the number of nodes that are in the giant component in  $\text{SSBM}(n, k, a/n, b/n)$ , i.e., nodes that are in a linear size component of  $\text{SSBM}(n, k, a/n, b/n)$ . Then, for any  $\varepsilon > 0$ ,*

$$\mathbb{P}\{Q/n \notin [\beta - \varepsilon, \beta + \varepsilon]\} = o(1),$$

where  $\beta = 1 - \tau/d$  and  $\tau$  is defined in (6.24).

**Lemma 7.** *Let  $F$  denote the number of edges that are in planted trees of the giant in  $\text{SSBM}(n, k, a/n, b/n)$ , i.e., trees formed by nodes that have a single edge connecting them to a linear size component in  $\text{SSBM}(n, k, a/n, b/n)$ . Then, for any  $\varepsilon > 0$ ,*

$$\mathbb{P}\{F/n \notin [\phi - \varepsilon, \phi + \varepsilon]\} = o(1),$$

$$\text{where } \phi := (d - \tau)e^{-(d-\tau)},$$

and  $\tau$  is defined in (6.24).

*Proof of Theorem 9.* Let  $G \sim \text{SSBM}(n, k, a/n, b/n)$ , and let  $T$  be the number of isolated trees in  $G$ ,

$M$  the number of edges in those trees, and  $F$  the number of edges in the planted trees of the largest connected component of  $G$  (i.e., the giant). We have from Lemma 4, Lemma 5 and Lemma 7 that, for  $\varepsilon > 0$ , with high probability on  $G$ ,

$$\begin{aligned} T &\in \left[ \frac{\tau}{d} \left( 1 - \frac{\tau}{2} \right) - \varepsilon, \frac{\tau}{d} \left( 1 - \frac{\tau}{2} \right) + \varepsilon \right], \\ R &\in \left[ \frac{\tau^2}{2d} + (d - \tau)e^{-(d-\tau)} - \varepsilon, \frac{\tau^2}{2d} + (d - \tau)e^{-(d-\tau)} + \varepsilon \right]. \end{aligned}$$

Assume that  $T$  and  $R$  take typical values as above. We now build a typical vertex labelling on these trees:

- Pick an arbitrary node in each isolated tree, denote these by  $\{v_1, \dots, v_T\}$ , and denote the set of edges contained in these trees by  $\{E_1, \dots, E_M\}$ ;
- Pick the root node for each planted tree that is in the giant, denote these by  $\{w_1, \dots, w_k\}$ , where  $k$  is the number of planted trees (this number is not relevant in the following computations) and denote by  $\{E_{M+1}, \dots, E_{M+F}\}$  the number of edges contained in those planted trees;
- Assign the labels  $U_1^T := (U_{v_1}, \dots, U_{v_T})$  uniformly at random in  $[k]$ , and set  $\hat{X}_{w_1} = \sigma_{w_1}, \dots, \hat{X}_{w_k} = \sigma_{w_k}$ , i.e., assign the latter labels to their true community assignments. Then broadcast each of these labels in their corresponding trees by forwarding the labels on each edge with an independent  $k$ -ary symmetric channel of flip probability  $\frac{b}{a+(k-1)b}$ . This means that the variables  $Z_1, \dots, Z_{M+F}$  are drawn i.i.d. from the distribution

$$\nu := \left( \frac{a}{a + (k-1)b}, \frac{b}{a + (k-1)b}, \dots, \frac{b}{a + (k-1)b} \right), \quad (6.27)$$

on  $\mathbb{F}_k := \{0, 1, \dots, k-1\}$ , and that for each edge  $e$  in the trees, the input bit is forwarded by adding to it the  $Z_e$  variable modulo  $k$ ;

- Assign any other label (that is not contained in the trees) to their true community assignments. Define  $R := M + F$ ,  $Z_1^R := (Z_1, \dots, Z_R)$ , and denote by  $\hat{X}(U_1^T, Z_1^R)$  the previously defined assignment.

Note that the above gives the induced label-distribution on trees in  $\text{SSBM}(n, k, a/n, b/n)$ . Thus, with high probability on  $G$ , as  $T$  and  $M$  grow with  $n$ , this assignment is typical with high probability

on  $U_1^T, Z_1^R$ , i.e.,

$$\mathbb{P}_{U_1^T, Z_1^R} \{ \hat{X}(U_1^T, Z_1^R) \in T_\delta(G) \} = 1 - o(1). \quad (6.28)$$

Denote by  $\text{Emp}(Z_1^R)$  the empirical distribution on  $\mathbb{F}_k$  of the  $R$ -vector  $Z_1^R$ , and by  $\mathcal{B}_\varepsilon(\nu)$  the  $l_1$ -ball around  $\mu$  of radius  $\varepsilon$ . Denote by  $\eta$  the uniform distribution on  $[k]$ . Then by Sanov's Theorem, for any  $\varepsilon > 0$ ,

$$\mathbb{P}_{Z_1^R} \{ \text{Emp}(U_1^T) \in \mathcal{B}_\varepsilon(\eta), \text{Emp}(Z_1^R) \in \mathcal{B}_\varepsilon(\nu) \} \rightarrow 1, \quad (6.29)$$

as  $T, R$  diverge with  $n$ . Define now the set of realizations of  $Z_1^R$  that have a typical likelihood by

$$A_\varepsilon(\nu) := \{ z_1^R \in \mathbb{F}_k^R : k^{-R(H(\nu)+\varepsilon)} \leq \mathbb{P}\{Z_1^R = z_1^R\} \leq k^{-R(H(\nu)-\varepsilon)} \}$$

where  $H$  is the entropy with the logarithm in base  $k$ . For convenience of notation, define also  $A_\varepsilon(\eta) = \mathcal{B}_\varepsilon(\eta)$ . Again, for all  $\varepsilon > 0$

$$\mathbb{P}\{U_1^T \in A_\varepsilon(\eta), Z_1^R \in A_\varepsilon(\nu)\} \rightarrow 1.$$

Therefore,

$$\mathbb{P}_{U_1^T, Z_1^R} \{ \hat{X}(U_1^T, Z_1^R) \in T_\delta(G) \} \quad (6.30)$$

$$= \mathbb{P}_{U_1^T, Z_1^R} \{ \hat{X}(U_1^T, Z_1^R) \in T_\delta(G), U_1^T \in A_\varepsilon(\eta), Z_1^R \in A_\varepsilon(\nu) \} + o(1) \quad (6.31)$$

$$= \sum_{u_1^T \in A_\varepsilon(\eta), z_1^R \in A_\varepsilon(\nu)} \mathbb{1}(\hat{X}(u_1^T, z_1^R) \in T_\delta(G)) \mathbb{P}\{U_1^T = u_1^T\} \mathbb{P}\{Z_1^R = z_1^R\} + o(1) \quad (6.32)$$

$$\leq \sum_{u_1^T \in A_\varepsilon(\eta), z_1^R \in A_\varepsilon(\nu)} \mathbb{1}(\hat{X}(u_1^T, z_1^R) \in T_\delta(G)) k^{-T} k^{-R(H(\nu)-\varepsilon)} + o(1). \quad (6.33)$$

We have

$$\sum_{u_1^T \in A_\varepsilon(\eta), z_1^R \in A_\varepsilon(\nu)} \mathbb{1}(\hat{X}(u_1^T, z_1^R) \in T_\delta(G)) \leq |T_\delta(G)|,$$

since the left hand side counts a subset of the typical clusterings. We thus obtain from (6.28) and

(6.33) that with high probability on  $G$ ,

$$|T_\delta(G)| \geq (1 - o(1))k^{T+R(H(\nu)-\varepsilon)}.$$

Thus, with high probability on  $G$ ,

$$|T_\delta(G)| \geq (1 - o(1))k^{n(\psi-\varepsilon)+o(n)},$$

where

$$\psi = \frac{\tau}{d} \left(1 - \frac{\tau}{2}\right) + H(\nu) \left(\frac{\tau^2}{2d} + (d - \tau)e^{-(d-\tau)}\right),$$

which proves the half of the claim not covered by lemma 3.  $\square$

*Proof of Lemma 4.* Let  $T_j$  denote the number of isolated  $j$ -trees (i.e., trees on  $j$  vertices) in  $\text{SBM}(n, k, a, b)$ , i.e.,

$$T_j = \sum_{S \in \mathcal{T}_j(n)} \mathbb{1}_{\text{iso}}(S)$$

where  $\mathcal{T}_j(n)$  denotes the set of all  $j$ -trees on the vertex set  $[n]$ , and where  $\mathbb{1}_{\text{iso}}(S)$  is equal to 1 if  $S$  is an isolated tree, and 0 otherwise. Since the number of different trees on  $j$  vertices is given by  $j^{j-2}$  [11], and since there is an edge on a designated pair of vertices with probability  $d/n$ , we have

$$\begin{aligned} \mathbb{E}[T_j] &= \binom{n}{j} j^{j-2} \left(\frac{d}{n}\right)^{j-1} \left(1 - \frac{d}{n}\right)^{j(n-j)} \\ &= \frac{n}{d} (de^{-d})^j \frac{j^{j-2}}{j!} + O(1) \end{aligned}$$

and

$$\text{Var } T_j = \frac{1}{2} \frac{n}{d} (de^{-d})^{2j} \left(\frac{j^{j-2}}{j!}\right)^2 + O(1) = O(n).$$

Thus, by Chebyshev's inequality, for any  $\varepsilon > 0$ ,

$$\mathbb{P}\{T_j/n \notin [\tau_j/d - \varepsilon, \tau_j/d + \varepsilon]\} = O(1/n),$$

where

$$\tau_j = \frac{j^{j-2}(de^{-d})^j}{j!}.$$

and since each tree contains  $j - 1$  edges,

$$\mathbb{P}\{M_j/n \notin [(j-1)\tau_j/d - \varepsilon, (j-1)\tau_j/d + \varepsilon]\} = O(1/n).$$

□

*Proof of Lemma 5.* Let  $T$  denote the number of isolated trees in  $\text{SBM}(n, k, a, b)$ , i.e.,

$$T = \sum_{j=1}^n T_j,$$

where  $T_j$  is the number of isolated  $j$ -trees. Hence,

$$\mathbb{E}[T] = \sum_{j=1}^n \mathbb{E}[T_j] = \frac{n}{d} \sum_{j=1}^n \frac{j^{j-2}(de^{-d})^j}{j!} + O(1).$$

Since

$$\sum_{j=n}^{\infty} \frac{j^{j-2}(de^{-d})^j}{j!} = O(n^{-3/2}),$$

we have

$$\lim_{n \rightarrow \infty} \frac{1}{n} \mathbb{E}[T] = \frac{1}{d} \sum_{j=1}^{\infty} \frac{j^{j-2}(de^{-d})^j}{j!}.$$

Finally, since  $\text{Var}(T) = O(n)$ , the result for  $T$  follows from Chebyshev's inequality and the fact that (see [20])

$$\sum_{j=1}^{\infty} \frac{j^{j-2}(de^{-d})^j}{j!} = \tau - \tau^2/2 \tag{6.34}$$

when

$$\tau = \sum_{j=1}^{\infty} \frac{j^{j-1}(de^{-d})^j}{j!}.$$

For  $M$ , the result follows from similar arguments and the fact that

$$\mathbb{E}[M] = \sum_{j=1}^n (j-1) \mathbb{E}[T_j] = \frac{n}{d} \sum_{j=1}^n (j-1) \frac{j^{j-2} (de^{-d})^j}{j!} + O(1) = \frac{n}{d} \frac{\tau^2}{2} + O(1),$$

which uses again (6.34).  $\square$

*Proof of Lemma 7.* From Lemma 6, the giant component has with high probability a relative size in  $[\beta - \varepsilon, \beta + \varepsilon]$ . The probability that a node is connected to a giant component by a single edge is thus given by

$$\beta n(d/n)(1 - d/n)^{\beta n} + o(1) = \beta de^{-\beta d} + o(1), \quad (6.35)$$

and the expected number of such nodes is

$$n\beta de^{-\beta d} + o(n). \quad (6.36)$$

Now, let  $v$  and  $v'$  be random vertices. Conditioning on  $v$  being connected to the giant component by a single edge, the giant component still has relative size in  $[\beta - \varepsilon, \beta + \varepsilon]$  for any  $\varepsilon$  with probability  $1 - o(1)$ , so the probability that  $v'$  is connected to the giant component by a single edge is also  $\beta de^{-\beta d} + o(1)$ . That means that the expected value of the square of the number of nodes that are connected to the giant component by a single edge is

$$(n\beta de^{-\beta d})^2 + o(n^2)$$

hence the variance in the number of such nodes is  $o(n^2)$  and the lemma follows.  $\square$

### 6.1.3 Sampling estimates

**Theorem 10.** Let  $d := \frac{a+(k-1)b}{k}$ , assume  $d > 1$ , and let  $\tau = \tau_d$  be the unique solution in  $(0, 1)$  of  $\tau e^{-\tau} = de^{-d}$ , i.e.,  $\tau = \sum_{j=1}^{+\infty} \frac{j^{j-1}}{j!} (de^{-d})^j$ . The Typicality Sampling Algorithm detects<sup>1</sup> communities in  $\text{SBM}(n, k, a, b)$  if

$$\frac{a \ln a + (k-1)b \ln b}{k} - \frac{a + (k-1)b}{k} \ln \frac{a + (k-1)b}{k} \quad (6.37)$$

$$> \min \left( \frac{1 - \tau}{1 - \tau k / (a + (k-1)b)} 2 \ln(k), 2 \ln(k) - 2 \ln(2) e^{-a/k} (1 - (1 - e^{-b/k})^{k-1}) \right). \quad (6.38)$$

---

<sup>1</sup>Setting  $\delta > 0$  small enough gives the existence of  $\varepsilon > 0$  for detection.

*Proof.* Let  $t = \max(k^{(\psi-\varepsilon)n}, 2^{(e^{-a/k}(1-(1-e^{-b/k})^{k-1})-\varepsilon)n})$ . We have

$$\mathbb{P}\{\hat{\sigma}(G) \in B_\varepsilon(\sigma)\} \quad (6.39)$$

$$= \mathbb{E} \frac{|T_\delta(G) \cap B_\varepsilon(\sigma)|}{|T_\delta(G)|} \quad (6.40)$$

$$\leq \mathbb{E} \frac{|T_\delta(G) \cap B_\varepsilon(\sigma)|}{|T_\delta(G)|} \mathbb{1}(|T_\delta(G)| \geq t) \mathbb{1}(\sigma \in \text{Bal}(n, k, \delta)) + \mathbb{E} \mathbb{1}(|T_\delta(G)| < t) + \mathbb{E} \mathbb{1}(\sigma \notin \text{Bal}(n, k, \delta)) \quad (6.41)$$

$$\leq (1/t) \mathbb{E}|T_\delta(G) \cap B_\varepsilon(\sigma)| \mathbb{1}(|T_\delta(G)| \geq t) \mathbb{1}(\sigma \in \text{Bal}(n, k, \delta)) + \mathbb{P}\{|T_\delta(G)| < t\} + \mathbb{P}(\sigma \notin \text{Bal}(n, k, \delta)) \quad (6.42)$$

$$\leq (1/t) \cdot \mathbb{E}|T_\delta(G) \cap B_\varepsilon(\sigma)| \mathbb{1}(\sigma \in \text{Bal}(n, k, \delta)) + o(1) \quad (6.43)$$

$$\leq (1/t) k^n e^{-A(\varepsilon, \delta)n/k} + o(1). \quad (6.44)$$

Using the bound  $t \geq 2^{(e^{-a/k}(1-(1-e^{-b/k})^{k-1})-\varepsilon)n}$  and the bound on  $A(\varepsilon, \delta)$  from Lemma 1, the exponent in (6.44) can be made to vanish if

$$\begin{aligned} & \frac{1}{k} \left( \frac{a+b(k-1)}{2} \ln \frac{k}{(a+(k-1)b)} + \frac{a}{2} \ln a + \frac{b(k-1)}{2} \ln b \right) + e^{-a/k} (1 - (1 - e^{-b/k})^{k-1}) \ln(2) \\ & > \ln(k). \end{aligned} \quad (6.45)$$

Alternately, plugging in the values of  $\psi$  from Theorem 9 and  $A(\varepsilon, \delta)$  from Lemma 1, the exponent in (6.44) is vanishing if

$$\frac{1}{k \ln(k)} \left( \frac{a+b(k-1)}{2} \ln \frac{k}{(a+(k-1)b)} + \frac{a}{2} \ln a + \frac{b(k-1)}{2} \ln b \right) \quad (6.46)$$

$$+ \frac{\tau}{d} \left( 1 - \frac{\tau}{2} \right) \quad (6.47)$$

$$+ \frac{1}{\ln(k)} \left( \frac{a}{a+(k-1)b} \ln \left( \frac{a+(k-1)b}{a} \right) + \frac{(k-1)b}{a+(k-1)b} \ln \left( \frac{a+(k-1)b}{b} \right) \right) \quad (6.48)$$

$$\cdot \left( \frac{\tau^2}{2d} + (d-\tau)e^{-(d-\tau)} \right) \quad (6.49)$$

$$> 1. \quad (6.50)$$

Since  $\tau$  is the solution in  $(0, 1)$  of  $\tau e^{-\tau} = d e^{-d}$ , we have

$$\frac{\tau^2}{2d} + (d-\tau)e^{-(d-\tau)} = \tau \left( 1 - \frac{\tau}{2d} \right),$$

and this simplifies to

$$\frac{1}{2 \ln(k)} \left( -d \ln d + \frac{a \ln a + b(k-1) \ln b}{k} \right) \quad (6.51)$$

$$+ \frac{1}{2 \ln(k)} \left( d \ln d + d \ln k - \frac{a \ln a + b(k-1) \ln b}{k} \right) \cdot \frac{2\tau}{d} \left( 1 - \frac{\tau}{2d} \right) \quad (6.52)$$

$$> 1 - \frac{\tau}{d} \left( 1 - \frac{\tau}{2} \right). \quad (6.53)$$

Algebraic manipulations lead to the bound in the theorem.  $\square$

**Corollary 3.** *Detection is solvable*

$$\text{in SSBM}(n, k, 0, b/n) \quad \text{if} \quad b > \frac{2k \ln k}{(k-1) \ln \frac{k}{k-1}} f(\tau, b(k-1)/k), \quad (6.54)$$

$$\text{in SSBM}(n, k, a/n, b/n) \quad \text{if} \quad a > a_k(b), \quad \text{where } a_k(0) = k. \quad (6.55)$$

*Proof.* Note that dropping the term in (6.48) above and ignoring the contribution of (6.45) leads to the weaker bound

$$\frac{1}{2 \ln k} \left( \frac{a \ln a + (k-1)b \ln b}{k} - d \ln d \right) > 1 - \frac{\tau}{d} \left( 1 - \frac{\tau}{2} \right),$$

which implies Corollary 3 part 2 since  $\frac{\tau}{d} \left( 1 - \frac{\tau}{2} \right)$  tends to  $1/2$  as  $\tau$  tends to 1 when  $b$  tends to 0 and  $d$  tends to  $a/k$ . Part 1 follows from algebraic manipulations.  $\square$

## 6.2 Achieving the KS threshold

The proof works for the general model  $\text{SBM}(n, p, Q/n)$ . Recall that  $P = \text{diag}(p)$ , and  $\lambda_1, \dots, \lambda_h$  are the distinct eigenvalues of  $PQ$  in order of decreasing magnitude.

**Definition 13.** *For any  $r$  and  $m$ , an  $r$ -nonbacktracking walk of length  $m$  is a sequence of vertices  $v_0, v_1, \dots, v_m$  such that  $v_i \neq v_j$  whenever  $|i - j| \leq r$  and  $v_i$  is adjacent to  $v_{i+1}$  for all  $i$ .*

**Definition 14.** *Given  $r, m > 0$  and vertices  $v$  and  $v'$ , let  $W_{m[r]}(v, v')$  be the number of  $r$ -nonbacktracking walks of length  $m$  from  $v$  to  $v'$ .*

**Definition 15.** *Given  $r, m > 0$ , graph  $G$ , an assignment of a real number  $x_v$  to every vertex  $v$ , a*

multiset of real numbers  $S$ , and a vertex  $v$ , let

$$W_{m/S[r]}(x, v) = \sum_{T \subseteq S} \prod_{y \in T} (-y) \sum_{v' \in G} x_{v'} W_{m-|T|[r]}(v', v). \quad (6.56)$$

In other words,  $W_{m/\emptyset[r]}(x, v)$  is the sum over all  $r$ -nonbacktracking walks of length  $m$  ending at  $v$  of the values of  $x$  at their initial vertices, and for  $S \neq \emptyset$  and  $y \in S$ , we have that  $W_{m/S[r]}(x, v) = W_{m/(S \setminus \{y\})[r]}(x, v) - y \cdot W_{m-1/(S \setminus \{y\})[r]}(x, v)$ . Note that the ABP algorithm sets  $Y_{v,t}$  equal to  $W_{t/\emptyset[r]}(x, v)$  for all  $0 < t \leq m$  and  $v \in G$  in step 2b. Then in step 2c, it sets  $y_v^{(m)}$  equal to  $W_{m/S[r]}(x, v)$ , where  $S$  is the multiset containing  $\lceil \frac{m-r-(2r+1)s'}{l} \rceil$  copies of  $\lambda_{s'}$  for every  $s' < s$ .

Further intuition on (6.56) will be provided with Definition 17. The plan is to select the  $x_v$  independently according to a Normal distribution, and then compute  $W_{m/S[r]}(x, v)$  for appropriate  $S$ ,  $m$ , and  $r$ . The assignment of  $x$  will inevitably have slightly different average values in different communities, and under the right conditions, these differences will be amplified to the point of allowing differentiation between communities with asymptotically nonzero advantage.

*Proof of Theorem 1:* When ABP splits  $\Gamma$  off of  $G$ , the remaining graph is still drawn from the SBM, albeit with connectivity  $(1 - \gamma)Q$ . The formula for  $\gamma$  ensures that if  $\lambda_2^2 > \lambda_1$  then  $((1 - \gamma)\lambda_2)^2 > ((1 - \gamma)\lambda_1)$ . Now, let  $m'' = \lfloor \sqrt{\log \log n} \rfloor + 1$ . For each  $v \in G$  and  $t > 0$ , let  $N_t(v)$  be the set of all vertices  $t$  edges away from  $v$ ,  $N_{\leq t}(v)$  be the subgraph of  $G$  induced by the vertices within  $t$  edges of  $v$ , and  $T_v = \{v' : \exists v'' \in N_{m''}(v) \mid (v', v'') \in \Gamma\}$ . Unless the original graph has a cycle in  $N_{\leq m''}(v) \cup T_v$ , we have that

$$y_v'' = \sum_{v' \in T_v} y_{v'}^{(m)}$$

Now, let  $w$  be an eigenvector of  $PQ$  with eigenvalue  $\lambda_i$  and magnitude 1. Also, for every  $v \in G$  and  $t \geq 0$ , let  $W_t(v) = \sum_{v' \in N_t(v)} w_{\sigma_{v'}} / p_{\sigma_{v'}}$ . Given a fixed value of  $N_{\leq t}(v)$ ,  $v' \in N_t(v)$ , a fixed value of  $\sigma_{v'}$ , and a community  $j$ , the expected number of vertices not in  $N_{\leq t}(v)$  that are in community  $j$  and adjacent to  $v'$  is  $\left(1 - \frac{|N_{\leq t}(v)|}{n}\right) (1 - \gamma) e_j \cdot PQ e_{\sigma_{v'}}$ . So, we can show by induction on  $t$  that for any  $t \geq 0$ ,  $n' > 0$ , and  $z \in \mathbb{R}$ , we have

$$E[W_{m''}(v) | W_{m''-t}(v) = z, |N_{m''-t}(v)| = n'] = (1 - \gamma)^t \lambda_i^t z + O\left(\frac{((1 - \gamma)\lambda_1)^{2t} (n')^2}{n}\right).$$

Similarly, for any  $t \geq 0$ ,  $0 < n' \leq \ln^{2m''-2t}(n)$ , and  $z \in \mathbb{R}$ , we have

$$\text{Var}[W_{m''}(v) | W_{m''-t}(v) = z, |N_{m''-t}(v)| = n'] = O\left(\sum_{t'=1}^t n' (1 - \gamma)^{t+t'} \lambda_1^{t-t'} \lambda_i^{2t'}\right)$$

Now, given nonintersecting graphs  $N'$  and  $N''$  with  $|N'|, |N''| = O(\ln^{2m''}(n))$ ,  $v \in N'$ , and  $v' \in N''$ , whether or not the subgraph of  $G$  induced by  $V(N')$  is  $N'$  is independent of whether or not the subgraph of  $G$  induced by  $V(N'')$  is  $N''$ . Also, there are no edges between these two graphs with probability  $1 - O(|N'| \cdot |N''|/n)$  regardless of what form these graphs take. Furthermore, for any  $v'' \notin N' \cup N''$ , regardless of the value of  $G \setminus \{v''\}$  there are no edges from  $v''$  to  $V(N')$  with probability  $1 - \lambda_1 |N'|/n + O(|N'|^2/n^2)$ , there are no edges from  $v''$  to  $V(N'')$  with probability  $1 - \lambda_1 |N''|/n + O(|N''|^2/n^2)$ , and there are no edges from  $v''$  to either with probability  $1 - \lambda_1(|N'| + |N''|)/n + O((|N'| + |N''|)^2/n^2)$ . So, we have that

$$\begin{aligned} & |P[N_{m''}(v) = N', N_{m''}(v') = N''] - P[N_{m''}(v) = N'] \cdot P[N_{m''}(v') = N'']| \\ &= O(|N'| \cdot |N''|/n) P[N_{m''}(v) = N', N_{m''}(v') = N''] \end{aligned}$$

Furthermore, the probability that  $N_{m''}(v)$  and  $N_{m''}(v')$  intersect is  $O(\lambda_1^{2m''}/n)$ . Also,  $|W_{m''}(v)| \leq n \cdot \max_i |w_i/p_i|$  for all  $v$ , and with probability  $1 - o(n^{-20})$ , every vertex in  $G$  has degree less than  $\ln^2(n)$ , so  $W_{m''}(v) = O(\ln^{2m''}(n))$  for all  $v$ . So, for  $v \neq v'$ , the correlation between  $W_{m''}(v)$  and  $W_{m''}(v')$  is  $o(1/\sqrt{n})$ , and the correlation between  $W_{m''}^2(v)$  and  $W_{m''}^2(v')$  is also  $o(1/\sqrt{n})$ . Therefore, with probability  $1 - o(1)$ , the average value of  $W_{m''}(v)$  over all  $v$  in community  $j$  is  $(1 - \gamma)^{m''} \lambda_i^{m''} (w_j/p_j + o(1))$  for all  $j$ . Also, there exists constant  $\delta$  such that  $\sum_{v \in G} (W_{m''}(v))^2 \leq n\delta \sum_{t=1}^{m''} (1 - \gamma)^{m''+t} \lambda_1^{m''-t} \lambda_i^{2t}$  with probability  $1 - o(1)$ .

On another note, for each  $v$ , we have that  $y_v^{(m)} = W_{m/\{c_i\}}(x, v)$ , so by lemma 19,

$$\text{Var}[W_{m/\{c_i\}}(x, v)] = O\left(\frac{1}{n} \prod_{j=1}^m ((1 - \gamma)\lambda_s - c_j)^2\right)$$

and the empirical variance of  $\{y_v^{(m)}\}$  is  $O\left(\frac{1}{n} \prod_{j=1}^m ((1 - \gamma)\lambda_s - c_j)^2\right)$  with probability  $1 - o(1)$ . Now, let  $\bar{y}$  be the vector such that  $\bar{y}_i = \sum_{v \in \Omega_i} y_v^{(m)}/n$  for all  $i$ . By Lemma 17, we know that with probability  $1 - o(1)$ , the component of  $\bar{y}$  on  $PQ$ 's eigenspace of eigenvalue  $\lambda_s$  has magnitude  $\Omega(\frac{1}{\sqrt{n}} \prod |(1 - \gamma)\lambda_s - c_j|)$ , while all of its components on  $PQ$ 's other eigenspaces have magnitudes of  $O(\frac{1}{\log(n)\sqrt{n}} \prod |(1 - \gamma)\lambda_s - c_j|)$ .

Now, let  $y_v''' = \sum_{v' \in T_v} y_{v'}^{(m)}$  for all  $v$ , and recall that  $y_v'' = y_v'''$  unless the original graph has a

cycle in  $N_{\leq m''}(v) \cup T_v$ . Observe that

$$\begin{aligned} \sum_{v \in G} w_{\sigma_v} y_v''' / p_{\sigma_v} &= \sum_{v, v', v'' \in G: v' \in N_{m''}(v), (v', v'') \in \Gamma} w_{\sigma_v} y_{v''}^{(m)} / p_{\sigma_v} \\ &= \sum_{(v', v'') \in \Gamma} W_{m''}(v') \cdot y_{v''}^{(m)} \end{aligned}$$

For each  $v, v' \in G$  without an edge between them,

$$P[(v, v') \in \Gamma | \sigma_v = j, \sigma_{v'} = j'] = (1 + O(1/n)) \gamma Q_{j, j'} / n$$

So, there exists  $\delta_w > 0$  such that for a fixed  $G$ ,  $\sigma$ , and  $x$ , with probability  $1 - o(1)$  it will be the case that

$$E \left[ \sum_{v \in G} w_{\sigma_v} y_v''' / p_{\sigma_v} \right] = \gamma (1 - \gamma)^{m''} \lambda_i^{m''+1} (w \cdot P^{-1} \bar{y} + o(\|\bar{y}\|_2 / \sqrt{n})) n$$

and

$$\begin{aligned} \text{Var} \left[ \sum_{v \in G} w_{\sigma_v} y_v''' / p_{\sigma_v} \right] &\leq (\delta_w / n) \sum_{v', v''} (W_{m''}(v') \cdot y_{v''}^{(m)})^2 \\ &\leq n \delta_w^2 \sum_{t=1}^{m''} (1 - \gamma)^{m''+t} \lambda_1^{m''-t} \lambda_i^{2t} \cdot \prod_{j=1}^m ((1 - \gamma) \lambda_s - c_j)^2 \\ &= o \left( \left( \gamma (1 - \gamma)^{m''} \lambda_s^{m''+1} \cdot \sqrt{n} \prod_{j=1}^m ((1 - \gamma) \lambda_s - c_j)^2 \right)^2 \right). \end{aligned}$$

This means that with probability  $1 - o(1)$ , we have that

$$\text{Var} \left[ \sum_{v \in G} w_{\sigma_v} y_v''' / p_{\sigma_v} \right] = o(\gamma^2 (1 - \gamma)^{2m''} \lambda_s^{2m''+2} \|\bar{y}\|_2^2 n^2).$$

In particular, if we choose an orthonormal eigenbasis for  $PQ$ ,  $w_1, \dots, w_k$ , then with probability  $1 - o(1)$  this will hold for all  $w_i$  in the basis, which implies that with probability  $1 - o(1)$ , we have that  $\sum_{v \in \Omega_j} y_v''' / n = \gamma (1 - \gamma)^{m''} ((PQ)^{m''+1} \bar{y})_j + o(\gamma (1 - \gamma)^{m''} \lambda_s^{m''+1} \|\bar{y}\|_2)$  for all  $j \in [k]$ . Also, for fixed  $G$  and  $x$  and with probability  $1 - o(1)$  we have

$$\begin{aligned}
\sum_{v \in G} (y_v''')^2 &= \sum_{v, v_2, v_3, v'_2, v'_3 \in G: v_2, v'_2 \in N_{m''}(v), (v_2, v_3) \in \Gamma, (v'_2, v'_3) \in \Gamma} y_{v_3}^{(m)} \cdot y_{v'_3}^{(m)} \\
&= \sum_{v, v_2, v_3, v'_2, v'_3 \in G: v_2, v'_2 \in N_{m''}(v), (v_2, v_3) \neq (v'_2, v'_3) \in \Gamma} y_{v_3}^{(m)} \cdot y_{v'_3}^{(m)} \\
&+ \sum_{v, v_2, v_3 \in G: v_2 \in N_{m''}(v), (v_2, v_3) \in \Gamma} (y_{v_3}^{(m)})^2 \\
&= (1 + o(1)) \sum_{v \in G} \left( \sum_{v_2 \in N_{m''}(v)} \gamma(Q\bar{y})_{\sigma_{v_2}} \right)^2 \\
&+ (1 + o(1)) \gamma(1 - \gamma)^{m''} \lambda_1^{m''+1} \sum_{v \in G} (y_v^{(m)})^2 \\
&\leq (1 + o(1)) n \sum_{t=1}^{m''} \gamma^2 (1 - \gamma)^{m''+t} \lambda_1^{m''-t} \lambda_s^{2t+2} \|\bar{y}\|_2^2 / (\min p_i) + O(\gamma(1 - \gamma)^{m''} \lambda_1^{m''+1} \|\bar{y}\|_2^2 n) \\
&= (1 + o(1)) n \cdot \frac{(1 - \gamma) \lambda_s^2}{\min p_i [(1 - \gamma) \lambda_s^2 - \lambda_1]} \gamma^2 (1 - \gamma)^{2m''} \lambda_s^{2m''+2} \|\bar{y}\|_2^2
\end{aligned}$$

With probability  $1 - o(1)$ , there are only  $O(\ln(n))$  vertices  $v \in G$  such that  $y_v'' \neq y_v'''$ . With probability  $1 - o(1)$ , no such vertex has more than  $\ln^{2m''+2}(n)$  vertices within  $m'' + 1$  edges of it and no such vertex has more than one cycle within  $m'' + 1$  edges of it. Assuming this holds, these differences are small enough that the results above still hold if we substitute  $y_v''$  for  $y_v'''$ . Among other things, this means that we can choose  $\sigma_1$  and  $\sigma_2$  such that the average value of  $y_v''$  for  $v \in \Omega_{\sigma_1}$  differs from the average value of  $y_v''$  for  $v \in \Omega_{\sigma_2}$  by at least  $(1 - o(1)) \gamma(1 - \gamma)^{m''} \lambda_s^{m''+1} \|\bar{y}\|_2$ .

Now, let  $a$  be the difference between the average value of  $y_v''$  for  $v \in \Omega_{\sigma_1}$  and the average value of  $y_v''$  for  $v \in \Omega_{\sigma_2}$ . Also, let  $b$  be the average value of  $(y_v'')^2$ . There exists constant  $\delta'$  such that  $b \leq \delta' a^2$  with probability  $1 - o(1)$ . The average value of  $1/2 + y_v'' / (C \sqrt{\sum (y_{v'}'')^2 / n})$  for  $v \in \Omega_{\sigma_1}$  differs from its average value for  $v \in \Omega_{\sigma_2}$  by  $\frac{a}{C\sqrt{b}}$ . For any  $v$ ,

$$\min(|y_v'' / (C \sqrt{\sum (y_{v'}'')^2 / n})| - 1/2, 0) \leq 2(y_v'' / (C\sqrt{b}))^2$$

So, the average value of  $\min(|y_v'' / (C \sqrt{\sum (y_{v'}'')^2 / n})| - 1/2, 0)$  for  $v$  in any community is at most  $2/(C^2 \min p_i)$ . So, the probability that a vertex from community  $\sigma_1$  is assigned to group 2 differs from the probability that a vertex from community  $\sigma_2$  is assigned to group 2 by at least  $\frac{a}{C\sqrt{b}} - 4/(C^2 \min p_i)$ . For a sufficiently large constant  $C$ , and small constant  $\epsilon > 0$  this will be greater than  $\epsilon$  with probability  $1 - o(1)$ , as desired.

ABP initializes in  $O(n)$  time. Computing the  $y^{(t)}$  takes  $O(n \log n)$  expected time. The eigenvalue

compensation step can be carried out in  $O(n \log n)$  time if  $\left(\prod_{s' < s} M_{s'}^{\lceil \frac{m-r-(2r+1)s'}{l} \rceil}\right) e_m$  is computed first and then  $Y$  is multiplied by it, and computing  $y''$  takes  $o(n \log n)$  time. The assignment step also takes  $O(n)$  time, so the entire algorithm can be run in  $O(n \log n)$  time.  $\square$

### 6.2.1 Preliminaries

Our first step in proving the lemmas used above is to reexpress  $W_{m/S[r]}(x, v)$  as a sum over  $r$ -nonbacktracking walks of an expression that we will eventually be able to bound. Then we will establish some properties of  $r$ -nonbacktracking walks that we will need.

**Definition 16.** For any  $r \geq 1$  and series of vertices  $v_0, v_1, \dots, v_m$ , let  $W_r((v_0, \dots, v_m))$  be 1 if  $v_0, \dots, v_m$  is an  $r$ -nonbacktracking walk, and 0 otherwise.

**Definition 17.** For any  $r \geq 1$ , series of vertices  $v_0, \dots, v_m$ , and series of real numbers  $c_0, \dots, c_m$ , let  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  be the sum, over all subserieses  $i_0, \dots, i_{m'}$  of  $0, \dots, m$  of

$$\left( \prod_{i \notin (i_0, \dots, i_{m'})} (-c_i/n) \right) \cdot W_r((v_{i_0}, v_{i_1}, \dots, v_{i_{m'}})).$$

Note that for any  $c_0, \dots, c_m$  consisting of the elements of  $S$  and 0's with  $c_0 = c_m = 0$ ,

$$W_{m/S[r]}(x, v) = \sum_{v_0, \dots, v_m \in G: v_m = v} [x_{v_0} \cdot W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))].$$

**Definition 18.** Given a series of vertices  $v_0, \dots, v_m$ , its walk graph is the graph consisting of every vertex in the series with an edge between each pair of vertices that are adjacent in the series. Note that the walk graph has a number of vertices equal to the number of distinct vertices in  $v_0, \dots, v_m$ , a number of edges equal to the cardinality of  $\{\{v_i, v_{i+1}\} : 0 \leq i < m\}$ , and every vertex in it other than  $v_0$  or  $v_m$  must have degree at least 2.

**Lemma 8.** Consider the complete graph  $K_n$ . For any  $m, b, w, e \geq 0$ , there are at most  $2n^{m-w+1}(m+1)^{4(w-e)}2^{2b(w-e)}$  walks of length  $m$  in  $K_n$  that have  $m+1-w$  distinct vertices,  $m-e$  distinct edges, no vertex repeated more than  $b$  times, and no edge repeated twice in a row.

*Proof.* Choose such a walk, and let  $H$  be its walk graph. Also, let the distinct vertices in the walk be  $v_1, v_2, \dots, v_{m+1-w}$ , in order of first appearance. There are at most  $n^{m-w+1}$  possible choices of which vertices these are. For any  $i$  such that  $v_i$  is not adjacent to  $v_{i+1}$  in the walk, the first edge in the walk after  $v_i$  and the first edge leading to  $v_{i+1}$  must both have not appeared previously in the walk.

Otherwise, the edge between  $v_i$  and  $v_{i+1}$  must not have appeared previously. So, there are at most  $m - e - (m - w) = w - e$  indices  $i$  such that  $v_{i+1}$  is not adjacent to  $v_i$ . There are at most  $(m + 1)^{w-e}$  choices of which indices have this property, and  $m + 1$  choices of which vertex the first edge leading to each such  $v_{i+1}$  has on its other side. That accounts for  $m - w$  of  $H$ 's edges, so there are at most  $(m + 1)^{2(w-e)}$  possibilities of what the others are. Thus, there are at most  $n^{m-w+1}(m + 1)^{4(w-e)}$  possible graphs  $H$ .

Now, let  $v'_0, \dots, v'_m$  be the complete walk in question. Given the values of  $v'_i$  and  $v'_{i-1}$ , the only possible values of  $v'_{i+1}$  are the vertices that are adjacent to  $v'_i$  other than  $v'_{i-1}$ . Furthermore, since no vertex appears in the walk more than  $b$  times, that means that the number of possible walks for a given  $H$  is at most

$$\begin{aligned}
& 2 \prod_{v \in H: \text{degree}(v) > 1} (\text{degree}(v) - 1)^b \\
& \leq 2 \prod_{v \in H: \text{degree}(v) > 1} 2^{b(\text{degree}(v) - 2)} \\
& \leq 2^{1+b \sum_{v \in H: \text{degree}(v) > 1} (\text{degree}(v) - 2)} \\
& \leq 2^{1+2b(w-e)}.
\end{aligned}$$

□

**Definition 19.** Given a sequence of vertices  $v_0, \dots, v_m$ , a *fresh segment* is a consecutive subsequence  $v_a, v_{a+1}, \dots, v_b$  such that no interior vertex of the segment has degree greater than 2 in the walk graph of  $v_0, \dots, v_m$ , no interior vertex of the segment is equal to  $v_m$ , and the walk  $v_0, \dots, v_a$  does not traverse any of the segment's edges. Call the segment *nonrepeated* if none of its edges occur in the walk  $v_b, \dots, v_m$  and *repeated* otherwise.

**Lemma 9** (Walk decomposition lemma). Let  $v_0, \dots, v_m$  define a walk in  $K_n$  that has  $m + 1 - w$  distinct vertices and  $m - e$  distinct edges. There exists a set of at most  $3(w - e) + 1$  fresh segments of the walk that are disjoint except at end vertices, and such that every edge in the walk is in one of these segments. Also, every end vertex of one of these segments is either  $v_0$ ,  $v_m$ , or a vertex that is repeated in the walk.

*Proof.* Let  $G'$  be the walk graph of  $v_1, \dots, v_m$ . Let  $S$  consist of  $v_1, v_m$ , and all vertices in  $G'$  of degree greater than 2. The total degree of all vertices in  $S$  is at most  $6(w - e) + 2$ , so  $G'$  consists of these vertices and at most  $3(w - e) + 1$  paths between them that do not contain any vertex from  $S$ . Once the walk goes onto one of these paths, the facts that it is nonbacktracking,  $v_m$  is not in the path,

and no vertex on the path has degree greater than 2 forces it to continue to the end of the path. The walk must traverse each of these paths for the first time at some point. Therefore, the set of serieses of vertices corresponding to the first traversals of these paths is the desired set of fresh segments.  $\square$

**Definition 20.** *Given a sequence of vertices  $v_0, \dots, v_m$ , let its standard decomposition be the collection of fresh segments generated by the above construction for  $v_0, \dots, v_m$ .*

**Definition 21.** *Given a collection of fresh segments, let its measures be  $d, d', t, t'$ , where  $d$  is the number of nonrepeated fresh segments in the collection,  $t$  is the sum of their lengths,  $d'$  is the number of repeated fresh segments in the collection, and  $t'$  is the sum of their lengths. Also, let the measures of a series of vertices denote the measures of its standard decomposition.*

### 6.2.2 The shard decomposition

The next step in the proof is to establish an upper bound on  $|E[W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))]|$  when  $(c_0, \dots, c_m), (v_0, \dots, v_m)$  satisfies some special properties. Then we will show that  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  can always be expressed as a linear combination of shards, expressions of the form  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  that have the aforementioned properties. After that, we will show that if  $(c_0, \dots, c_m)$  is chosen correctly, the magnitudes of the expected values of many of the shards of  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  must be fairly small.

**Lemma 10.** *Let  $r \geq 1$  and  $c_0, \dots, c_m$  be a series of real numbers with  $c_0 = c_m = 0$ . Also let  $v_0, \dots, v_m$  be a series of vertices with standard decomposition  $v_{a_1}, \dots, v_{b_1}, v_{a_2}, \dots, v_{b_2}, \dots, v_{a_d}, \dots, v_{b_d}$ , ordered so that  $b_i \leq a_{i+1}$  for each  $i$ . Assume that for any  $i, j$  such that  $v_i$  occurs elsewhere in the series and  $|i - j| \leq r$ ,  $c_j = 0$ . Then,*

$$|E[W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))]| \leq n^{-\sum_{i=1}^d b_i - a_i} (\min p_i)^{-d} \prod_{i=1}^d \sum_{j=1}^k \prod_{i'=a_i}^{b_i-1} |\lambda_j - c_{i'}|.$$

*Proof.* For any  $i$  such that  $c_i \neq 0$ ,  $i$  is at least  $r$  away from every element of the series that is repeated. So, deleting  $v_i$  for all  $i$  in any subset of  $\{i : c_i \neq 0\}$  has no effect on whether or not  $v_1, \dots, v_m$  is  $r$ -nonbacktracking. If  $v_1, \dots, v_m$  is not  $r$ -nonbacktracking, then the expected value of  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  is 0, and the conclusion holds. Now, consider the case where  $v_1, \dots, v_m$  is  $r$ -nonbacktracking. Since any subsequence of  $v_0, \dots, v_m$  resulting from deleting  $v_i$  for which  $c_i \neq 0$  is also  $r$ -nonbacktracking, the restriction that a vertex cannot repeat within  $r$  steps is irrelevant and  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m)) = W_{(c_0, \dots, c_m)[0]}((v_0, \dots, v_m))$ . Also,  $v_{a_i}$  and  $v_{b_i}$  are all repeated vertices

except possibly  $v_0$  and  $v_m$ . So,  $c_{a_i} = 0$  and  $c_{b_i} = 0$  for all  $i$ . That implies that

$$\begin{aligned} W_{(c_0, \dots, c_m)[0]}((v_0, \dots, v_m)) &= W_{(c_0, \dots, c_{a_1})[0]}((v_0, \dots, v_{a_1})) \cdot W_{(c_{a_1}, \dots, c_{b_1})[0]}((v_{a_1}, \dots, v_{b_1})) \\ &\quad \cdot W_{(c_{b_1}, \dots, c_{a_2})[0]}((v_{b_1}, \dots, v_{a_2})) \cdot W_{(c_{a_2}, \dots, c_{b_2})[0]}((v_{a_2}, \dots, v_{b_2})) \\ &\quad \cdot \dots \cdot W_{(c_{b_d}, \dots, c_m)[0]}((v_{b_d}, \dots, v_m)). \end{aligned}$$

If  $v_i$  is not part of one of the fresh segments, then  $v_i$  is a repetition of a vertex that is in one of them, so  $c_i = 0$ . Thus,

$$W_{(c_0, \dots, c_{a_1})[0]}((v_0, \dots, v_{a_1})) \cdot W_{(c_{b_1}, \dots, c_{a_2})[0]}((v_{b_1}, \dots, v_{a_2})) \cdot \dots \cdot W_{(c_{b_d}, \dots, c_m)[0]}((v_{b_d}, \dots, v_m))$$

is either 0 or 1. If it is 0 then there is some  $v_i, v_{i+1}$  that are not part of any of the fresh segments and do not have an edge between them. By the previous lemma, there must exist  $1 \leq i \leq d$  and  $a_i \leq j \leq b_i - 1$  such that  $\{v_i, v_{i+1}\} = \{v_j, v_{j+1}\}$ , and since the vertices are repeated,  $c_j = c_{j+1} = 0$ . So, the lack of an edge between them means that  $W_{(c_{a_i}, \dots, c_{b_i})[0]}((v_{a_i}, \dots, v_{b_i})) = 0$ . Either way,

$$W_{(c_0, \dots, c_m)[0]}((v_0, \dots, v_m)) = \prod_{i=1}^d W_{(c_{a_i}, \dots, c_{b_i})[0]}((v_{a_i}, \dots, v_{b_i})).$$

The fresh segments in a standard decomposition only intersect at their endpoints, so for a fixed assignment of communities to the endpoints,

$$\begin{aligned} |\mathbb{E}[W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))]| &= \prod_{i=1}^d |\mathbb{E}[W_{(c_{a_i}, \dots, c_{b_i})[0]}((v_{a_i}, \dots, v_{b_i}))]| \\ &= n^{-\sum b_i - a_i} \prod_{i=1}^d |e_{\sigma_{v_{a_i}}} P^{-1} \prod_{i'=a_i}^{b_i-1} (PQ - c_{i'} I) e_{\sigma_{v_{b_i}}}| \\ &\leq n^{-\sum b_i - a_i} (\min p_i)^{-d} \prod_{i=1}^d \sum_{j=1}^k \prod_{i'=a_i}^{b_i-1} |\lambda_j - c_{i'}|. \end{aligned}$$

□

Of course, for general  $c_0, \dots, c_m$  and  $v_0, \dots, v_m$  the condition that  $c_i = 0$  whenever there is a repeated vertex  $v_j$  with  $|i - j| \leq r$  may not be satisfied. We can deal with that using the fact that

for arbitrary  $c_0, \dots, c_m, v_0, \dots, v_m, r$ , and  $i$ ,

$$\begin{aligned} & W_{(c_0, \dots, c_{i-1}, c_i, c_{i+1}, \dots, c_m)[r]}((v_0, \dots, v_{i-1}, v_i, v_{i+1}, \dots, v_m)) \\ &= W_{(c_0, \dots, c_{i-1}, 0, c_{i+1}, \dots, c_m)[r]}((v_0, \dots, v_{i-1}, v_i, v_{i+1}, \dots, v_m)) \\ &\quad - \frac{c_i}{n} W_{(c_0, \dots, c_{i-1}, c_{i+1}, \dots, c_m)[r]}((v_0, \dots, v_{i-1}, v_{i+1}, \dots, v_m)). \end{aligned}$$

So, for any  $c_0, \dots, c_m$  with  $c_0 = c_m = 0$ ,  $v_0, \dots, v_m$ , and  $r$ , we can express  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  as a linear combination of expressions that the above lemma applies to by means of the following algorithm.

---

path-sum-conversion( $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$ ):

1. If there exist  $i$  and  $j$  such that  $v_i = v_j$ ,  $c_i \neq 0$ , and  $c_j = 0$ , return

$$\begin{aligned} & \text{path-sum-conversion}(W_{(c_0, \dots, c_{i-1}, 0, c_{i+1}, \dots, c_m)[r]}((v_0, \dots, v_{i-1}, v_i, v_{i+1}, \dots, v_m))) \\ & - \frac{c_i}{n} \text{path-sum-conversion}(W_{(c_0, \dots, c_{i-1}, c_{i+1}, \dots, c_m)[r]}((v_0, \dots, v_{i-1}, v_{i+1}, \dots, v_m))) \end{aligned}$$

2. Otherwise, if there exist  $j \neq j'$  such that  $v_j = v_{j'}$ , and  $i$  such that  $0 < |i - j| \leq r$  and  $c_i \neq 0$ , return

$$\begin{aligned} & \text{path-sum-conversion}(W_{(c_0, \dots, c_{i-1}, 0, c_{i+1}, \dots, c_m)[r]}((v_0, \dots, v_{i-1}, v_i, v_{i+1}, \dots, v_m))) \\ & - \frac{c_i}{n} \text{path-sum-conversion}(W_{(c_0, \dots, c_{i-1}, c_{i+1}, \dots, c_m)[r]}((v_0, \dots, v_{i-1}, v_{i+1}, \dots, v_m))) \end{aligned}$$

3. Otherwise, if there exist  $i \neq j$  such that  $v_i = v_j$ ,  $c_i \neq 0$ , and  $c_j \neq 0$ , return

$$\begin{aligned} & \text{path-sum-conversion}(W_{(c_0, \dots, c_{i-1}, 0, c_{i+1}, \dots, c_m)[r]}((v_0, \dots, v_{i-1}, v_i, v_{i+1}, \dots, v_m))) \\ & - \frac{c_i}{n} \text{path-sum-conversion}(W_{(c_0, \dots, c_{i-1}, c_{i+1}, \dots, c_m)[r]}((v_0, \dots, v_{i-1}, v_{i+1}, \dots, v_m))) \end{aligned}$$

4. Otherwise, if there exist  $i \neq j$  such that  $|i - j| \leq r$ , and  $v_i = v_j$ , return 0.

5. Otherwise, return  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$
- 

If there is more than one  $i$  satisfying the conditions of the case under consideration, the algorithm should choose one according to some rule, such as always using the smallest. Also, note that this algorithm exists as a proof technique allowing us to replace any expression of the form  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  with a sum of expressions of the form  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  that the previous lemma applies to. We would never actually run it.

**Lemma 11.** *Path-sum-conversion( $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$ ) always terminates. Furthermore, if there are no  $i, j$  such that  $c_i \neq 0$ ,  $c_j \neq 0$ , and  $|i - j| \leq r$ , then for any  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  that appears in the expression it outputs and any  $v_i$  that was deleted in the process of going from*

$W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  to  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$ , at least one of the following holds. There exists  $i'$  such that  $v'_{i'} = v_i$ , or there exists  $j$  such that  $|i - j| \leq r$ ,  $v_j$  was not deleted in the process of going from  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  to  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$ , and  $v_j$  appears more than once in the list  $(v'_0, \dots, v'_{m'})$ . Also, if  $c_0 = c_m = 0$ , then for any  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  that appears in the expression it outputs,  $c'_0 = c'_{m'} = 0$ .

*Proof.* The fact that the algorithm terminates follows immediately from the fact that when it recurses, each iteration of the algorithm has one fewer nonzero element of  $(c_0, \dots, c_m)$  than its parent iteration, so it goes down at most  $m$  layers. Now, assume that there are no  $i, j$  such that  $c_i \neq 0$ ,  $c_j \neq 0$ , and  $|i - j| \leq r$ . The algorithm only ever deletes  $c_i$  that have nonzero values, so this will continue to hold for any  $(c'_0, \dots, c'_{m'})$  that  $(c_0, \dots, c_m)$  is converted to. Likewise, if  $c_0 = c_m = 0$ , then the first and last elements of any  $(c'_0, \dots, c'_{m'})$  that  $(c_0, \dots, c_m)$  is converted to will always be 0. Also, note that if  $c_i = 0$ , then  $v_i$  will never be deleted.

Now, if the algorithm deletes  $v_i$  as an instance of case 1, then there exists  $j$  such that  $v_j = v_i$  and  $c_j = 0$ .  $v_j$  will never be deleted, and will thus be in any expression resulting from this deletion. If the algorithm deletes  $v_i$  as an instance of case 2, then there exist  $j \neq j'$  with  $v_j = v_{j'}$  and  $0 < |i - j| \leq r$ . We know that  $c_j = 0$  because  $c_i \neq 0$  and  $i$  and  $j$  are within  $r$  of each other.  $c_{j'} = 0$  as well, because if it did not,  $i = j', j = j$  would have satisfied the conditions for case 1. So, neither  $v_j$  nor  $v_{j'}$  will ever be deleted, and thus  $v_j$  will appear more than once in the paths for any expression resulting from this deletion. If the algorithm deletes  $v_i$  as an instance of case 3, then it must be the case that for every  $i'$  such that  $v_i = v_{i'}$ , there do not exist  $j \neq j'$  such that  $0 < |i' - j| \leq r$  and  $v_j = v_{j'}$ . There is no way for any operation the algorithm performs to result in such  $j$  and  $j'$  coming into existence, so if at any future point there is only one element of  $(v'_0, \dots, v'_{m'})$  that is equal to  $v_i$ , there is no case under which that element will be deleted. So, any  $(v'_0, \dots, v'_{m'})$  resulting from this deletion will contain at least one element equal to  $v_i$ .  $\square$

**Definition 22.**  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  is a level  $x$  shard of  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  if  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  is one of the expressions in the sum resulting from  $\text{path-sum-conversion}(W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m)))$  and exactly  $x$  of the vertices that were deleted in the process of converting  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  to  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  are repetitions of vertices in  $(v'_0, \dots, v'_{m'})$ . For a given  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$ ,  $\text{Shar}(W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m)))$  denotes the set of all of its shards, and  $\text{Shar}_x(W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m)))$  denotes the set of its level  $x$  shards.

**Definition 23.** Let an eigenvalue approximation be an ordered tuple  $\lambda'_1, \dots, \lambda'_{s-1}$  such that  $|\lambda_i - \lambda'_i| \leq |\lambda_i - \lambda'_{i'}|$  for all  $i$  and  $i'$ . Call such a tuple  $\Lambda$ -bounded if  $|\lambda'_i| \leq \Lambda$  and  $|\lambda_i| \leq \Lambda$  for all  $i$ . Also, let the

error of such an approximation be  $\max_i |\lambda_i - \lambda'_i|$ .

**Definition 24.** Given integers  $r, l$ , a tuple  $\lambda'_1, \dots, \lambda'_{s-1}$ , and a sequence  $c_0, \dots, c_m$ , let the sequence be an  $l[r]$ -cycle of  $(\lambda'_1, \dots, \lambda'_{s-1})$  if and only if  $(2r+1)(s-1) \leq l$ ,  $c_i = \lambda'_j$  for all  $i$  and  $j$  such that  $i < m-r$  and the remainder of  $i$  when divided by  $l$  is  $(2r+1)j$ , and  $c_i = 0$  whenever there is no  $j$  such that the above holds. Note that the first and last elements of an  $l[r]$  cycle are always 0, and that if  $(2r+1)(s-1) \leq l$  there exists a unique  $l[r]$ -cycle of length  $m$  for every  $m$ .

**Lemma 12.** Let  $\lambda'_1, \dots, \lambda'_{s-1}$  be a  $\Lambda$ -bounded eigenvalue approximation and  $c_0, \dots, c_m$  be an  $l[r]$ -cycle of this approximation for some  $l, r > 0$ . Then, let  $v_0, \dots, v_m$  be a series of vertices, and let  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  be a level  $x$  shard of  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$ . Finally, let  $d, d', t, t'$  be the measures of  $(v'_0, \dots, v'_{m'})$  and  $t'' = \lceil \frac{t-d(l+2r+1)}{l} \rceil$ . Then

$$\begin{aligned} & |\mathbb{E}(W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'})))| \\ & \leq n^{-t-t'} k^{d+d'} (\min p_i)^{-d-d'} \lambda_1^{t' + \max(t-(2x+d)(s-1)-l \cdot \max(t''-2x, 0), 0)} \\ & \quad \cdot (\lambda_1 + \Lambda)^{t - \max(t-(2x+d)(s-1)-l \cdot \max(t''-2x, 0), 0) - l \cdot \max(t''-2x, 0)} \\ & \quad \cdot \left( \max_{1 \leq j \leq k} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{\max(t''-2x, 0)}. \end{aligned}$$

*Proof.* Let  $v'_{a_1}, \dots, v'_{b_1}, v'_{a_2}, \dots, v'_{b_2}, \dots, v'_{a_d}, \dots, v'_{b_d}$  be the nonrepeated fresh segments in the standard decomposition of  $(v'_0, \dots, v'_{m'})$ , and  $v'_{a'_1}, \dots, v'_{b'_1}, v'_{a'_2}, \dots, v'_{b'_2}, \dots, v'_{a'_d}, \dots, v'_{b'_d}$  be the repeated fresh segments in its standard decomposition. For arbitrary  $1 \leq i \leq d'$  and  $a'_i \leq i' \leq b'_i$ , we have that  $v'_{i'}$  either is a repeated vertex or is within one step of one. So,  $c'_{i'} = 0$ . We know from a previous lemma that

$$\begin{aligned} & |\mathbb{E}[W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))]| \\ & \leq n^{-t-t'} (\min p_i)^{-d-d'} \left( \prod_{i=1}^d \sum_{j=1}^k \prod_{i'=a_i}^{b_i-1} |\lambda_j - c'_{i'}| \right) \left( \prod_{i=1}^{d'} \sum_{j=1}^k \prod_{i'=a'_i}^{b'_i-1} |\lambda_j - c'_{i'}| \right) \\ & \leq n^{-t-t'} k^d (\min p_i)^{-d-d'} \left( \prod_{i=1}^d \max_{1 \leq j \leq k} \prod_{i'=a_i}^{b_i-1} |\lambda_j - c'_{i'}| \right) \left( \prod_{i=1}^{d'} \sum_{j=1}^k \prod_{i'=a'_i}^{b'_i-1} |\lambda_j| \right) \\ & \leq n^{-t-t'} k^{d+d'} (\min p_i)^{-d-d'} \left( \prod_{i=1}^d \max_{1 \leq j \leq k} \prod_{i'=a_i}^{b_i-1} |\lambda_j - c'_{i'}| \right) \left( \prod_{i=1}^{d'} \prod_{i'=a'_i}^{b'_i-1} |\lambda_1| \right) \\ & \leq n^{-t-t'} k^{d+d'} (\min p_i)^{-d-d'} \lambda_1^{t'} \left( \prod_{i=1}^d \max_{1 \leq j \leq k} \prod_{i'=a_i}^{b_i-1} |\lambda_j - c'_{i'}| \right). \end{aligned}$$

Now, for each  $0 \leq i \leq m'$ , define  $f(i)$  such that  $v_{f(i)}$  corresponds to  $v'_i$ . Also, for each  $0 \leq i \leq m$ , define  $g(i)$  so that  $v'_{g(i)}$  corresponds to  $v_i$ , or  $v_{i+1}$  if  $v_i$  has been deleted. Also, for each  $1 \leq i \leq d$ , let  $y_i$  be the largest integer such that  $g(f(a_i + r + 1) + l \cdot y_i) \leq b_i - r$ , or 0 if  $a_i + 2r \geq b_i$ . Note that for any  $i' \geq i$ ,  $i' - i \geq g(i') - g(i)$ . As a result,

$$\begin{aligned} \sum_{i=1}^d y_i &= \frac{1}{l} \sum_{i=1}^d f(a_i + r + 1) + l \cdot y_i - f(a_i + r + 1) \\ &\geq \frac{1}{l} \sum_{i=1}^d g(f(a_i + r + 1) + l \cdot y_i) - g(f(a_i + r + 1)) \\ &\geq \frac{1}{l} \sum_{i=1}^d b_i - r - l - (a_i + r + 1) = \frac{t - d(2r + l + 1)}{l}, \end{aligned}$$

because  $|g(x) - g(x')| \leq |x - x'|$  for all  $x$  and  $x'$  and  $g(f(a_i + r + 1) + l \cdot (y_i + 1)) > b_i - r$ . So,  $\sum_{i=1}^d y_i \geq t''$ .

For every  $i$ , either  $a_i = 0$  or  $v'_{a_i}$  is repeated somewhere else in  $(v'_0, \dots, v'_{m'})$ . Likewise, either  $b_i = m'$  or  $v'_{b_i}$  is repeated somewhere else in  $(v'_0, \dots, v'_{m'})$ . The first  $r + 1$  and last  $r + 1$  of the  $c_i$  are 0, and whenever there exists  $j$  such that  $|i - j| \leq r$  and  $v'_j$  is a repeated vertex  $c'_i$  is 0, so  $c'_j = 0$  whenever  $a_i \leq j \leq a_i + r$  or  $b_i - r \leq j \leq b_i$  for some  $i$ . Also, for any  $1 \leq i \leq d$  and  $1 \leq j \leq y_i$ , it is the case that  $g(f(a_i + r + 1) + l \cdot j) - g(f(a_i + r + 1) + l \cdot (j - 1)) = l$  unless one of the vertices  $v_{f(a_i + r + 1) + l \cdot (j - 1)}, \dots, v_{f(a_i + r + 1) + l \cdot j - 1}$  was deleted and  $\{c'_{g(f(a_i + r + 1) + l \cdot (j - 1))}, \dots, c'_{g(f(a_i + r + 1) + l \cdot j - 1)}\}$  consists of one copy of  $\lambda'_{h''}$  for each  $h''$  and  $l - (s - 1)$  copies of 0 unless one of the vertices  $v_{f(a_i + r + 1) + l \cdot (j - 1)}, \dots, v_{f(a_i + r + 1) + l \cdot j - 1}$  was deleted or one of  $c_{f(a_i + r + 1) + l \cdot (j - 1)}, \dots, c_{f(a_i + r + 1) + l \cdot j - 1}$  was set to 0 as a result of the corresponding vertex being within distance  $r$  of a vertex that was repeated somewhere else in the walk. The entirety of the block is at least  $r$  away from any vertex that is repeated in  $(v'_0, \dots, v'_{m'})$ , so the second case is only possible if the other copy of that vertex was subsequently deleted. So, at most  $x$  blocks fall under each of these two cases, which means that there are at least  $\max(t'' - 2x, 0)$  uncorrupted blocks. Each corrupted block has at most one  $c'_i$  with a value of  $\lambda'_{h''}$  for each  $h''$  and there is at most one  $c'_i$  with a value of  $\lambda_{h''}$  between  $c'_{g(f(a_i + r + 1) + l \cdot y_i)}$

and  $c'_{b_i}$  for each  $i$  and  $h''$ . So,

$$\begin{aligned}
& \prod_{i=1}^d \max_{1 \leq j \leq k} \prod_{i'=a_i}^{b_i-1} |\lambda_j - c'_{i'}| \\
& \leq \left( \prod_{1 \leq i \leq d: b_i - a_i \leq 2r} \max_{1 \leq j \leq k} \prod_{i'=a_i}^{b_i-1} |\lambda_j - c'_{i'}| \right) \\
& \cdot \prod_{\substack{1 \leq i \leq d: \\ b_i - a_i > 2r}} \max_{1 \leq j \leq k} \prod_{i'=a_i}^{a_i+r} |\lambda_j - c'_{i'}| \prod_{i'=1}^{y_i} \prod_{i''=g(f(a_i+r+1)+l \cdot (i'-1))}^{g(f(a_i+r+1)+l \cdot i')-1} |\lambda_j - c'_{i''}| \prod_{i'=g(f(a_i+r+1)+l \cdot y_i)}^{b_i-1} |\lambda_j - c'_{i'}| \\
& \leq \left( \max_{1 \leq j \leq k} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{\max(t''-2x, 0)} \lambda_1^{\max(t-(2x+d)(s-1)-l \cdot \max(t''-2x, 0), 0)} \\
& \cdot (\lambda_1 + \Lambda)^{t - \max(t-(2x+d)(s-1)-l \cdot \max(t''-2x, 0), 0) - l \cdot \max(t''-2x, 0)}.
\end{aligned}$$

□

**Lemma 13.** *Let  $c_0, \dots, c_m$  be a sequence of real numbers with at most  $y$  nonzero elements, and  $v'_0, \dots, v'_{m'}$  be vertices. For any integer  $x$ , there are at most  $3^y(m+1)^x n^{m-m'-x}$  pairs of a sequence of vertices  $v_1, \dots, v_m$  and a sequence of real numbers  $c'_0, \dots, c'_{m'}$  such that  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  is a level  $x$  shard of  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$ .*

*Proof.* Path-sum-conversion only deletes vertices that have a nonzero corresponding element of  $c_0, \dots, c_m$ , so there are at most  $3^y$  possibilities for where the vertices that were deleted originally were, and which of them were copies of vertices in  $v'_0, \dots, v'_{m'}$ . There are at most  $(m+1)^x$  possibilities for the identities of the  $x$  deleted vertices that are repetitions of vertices in  $(v'_0, \dots, v'_{m'})$  and  $n^{m-m'-x}$  possibilities for the remaining  $m - m' - x$  deleted vertices. Furthermore, once the deleted vertices and their original locations are specified, the only sequence of vertices that could give rise to  $v'_0, \dots, v'_{m'}$  after that sequence of deletions is the sequence formed by splicing the deleted vertices into  $v'_0, \dots, v'_{m'}$  at the specified locations. Also, whenever path-sum-conversion recurses, the vertex that the current iteration was considering is deleted in one child iteration, and it is not deleted in any of the expressions resulting from the other branch. So, only one of the expressions in the sum output by  $\text{path-sum-conversion}(W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m)))$  results from the specified series of deletions, which means that the list of deleted vertices and their original locations implies a unique value of  $(c'_0, \dots, c'_{m'})$ . □

**Lemma 14.** *Let  $\lambda'_1, \dots, \lambda'_{s-1}$  be a  $\Lambda$ -bounded eigenvalue approximation and  $c_0, \dots, c_m$  be an  $l[r]$ -cycle of this approximation for some  $l$  and  $r > 0$ . Then, let  $v_0, \dots, v_m$  be a series of vertices, and*

let  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  be a level  $x$  shard of  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m))$  such that  $v'_0, \dots, v'_{m'}$  is nonbacktracking. The absolute value of the coefficient of  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  in the sum resulting from  $\text{path-sum-conversion}(W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m)))$  is at most  $(\Lambda/n)^{m-m'}$ .

*Proof.* First, note that since  $r \geq 1$ , for any  $i < i'$  such that  $c_i \neq 0$  and  $c_{i'} \neq 0$ , it must be the case that  $i' - i \geq 3$ . Now, let  $I$  and  $I'$  be two distinct subsets of  $\{i : c_i \neq 0\}$  with  $m - m'$  elements, and choose the smallest  $i \in (I \setminus I') \cup (I' \setminus I)$ . Assume without loss of generality that  $i \in I$ . Note that  $m \geq i + 3$  because  $I' \setminus I \neq \emptyset$ , and the smallest  $i' > i$  such that  $c_{i'} \neq 0$  is at least  $i + 3$ . Now, let  $t = |I \cap \{0, \dots, i - 1\}| = |I' \cap \{0, \dots, i - 1\}|$ . Deleting all vertices from  $v_0, \dots, v_m$  that have indices in  $I$  leaves  $v_{i+2}$  in the  $(i + 1 - t)$ th position, while deleting the vertices that have indices in  $I'$  leaves  $v_{i+1}$  in that position. So, if  $v_{i+1} \neq v_{i+2}$  the resulting sequences are different and if  $v_{i+1} = v_{i+2}$  the resulting sequences backtrack. Thus, there is only one subseries of vertices that can be deleted from  $v_0, \dots, v_m$  to yield  $v'_0, \dots, v'_{m'}$ . That means that there is only one route through which  $\text{path-sum-conversion}(W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m)))$  arrives at  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$ . Every time the algorithm deletes a vertex, it multiplies the expression's coefficient by  $-c_i/n$ , where  $i$  is that vertex's original index in  $v_0, \dots, v_m$ . So, the conclusion holds because  $|-c_i/n| \leq \Lambda/n$  for all  $i$ .  $\square$

### 6.2.3 Bounding the variance of $W_{m/S}$

The previous section provides the key techniques we will need to prove bounds. Now we need to apply them. We need to prove that expression involving  $W_{m/S}$  are within a certain range with high probability rather than merely computing their expected values, which requires us to bound their variances. That, in turn, will require us to bound the expected values of expressions of the form  $W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m)) \cdot W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$ . In order to convert that to something more familiar, let  $u_1, \dots, u_r$  be some new vertices that are adjacent to every vertex in  $r$ , and then note that for any  $c_0, \dots, c_{m_1}, c''_0, \dots, c''_{m_2} \in \mathbb{R}$  and  $v_0, \dots, v_{m_1}, v''_0, \dots, v''_{m_2} \in G$ ,

$$\begin{aligned} & W_{(c_0, \dots, c_{m_1})[r]}((v_0, \dots, v_{m_1})) \cdot W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'})) \\ &= W_{(c_0, \dots, c_{m_1}, 0, \dots, 0, c''_{m_2}, c''_{m_2-1}, \dots, c''_0)[r]}((v_0, \dots, v_{m_1}, u_1, u_2, \dots, u_r, v''_{m_2}, v''_{m_2-1}, \dots, v''_0)) \end{aligned}$$

because all of the vertices are connected to the  $u_i$  and they create enough distance between the  $v_i$  and the  $v''_i$  that they will never be within  $r$  of each other. Next we will establish a series of bounds on the expected values of these expressions, starting with the following.

**Lemma 15.** Let  $\lambda'_1, \dots, \lambda'_{s-1}$  be a  $\Lambda$ -bounded eigenvalue approximation with error at most  $2\Lambda(\min_i |\lambda_s - \lambda_i|/4\Lambda)^{s-1} |\lambda_s/\Lambda|^{l-s+1}$  and  $c_0, \dots, c_{m_1}$  and  $c''_0, \dots, c''_{m_2}$  be  $l[r]$ -cycles of this approximation for some  $l$  and  $r$ . Then, let  $(c'''_0, \dots, c'''_{m_1+m_2+r+1}) = (c_0, \dots, c_{m_1}, 0, \dots, 0, c''_{m_2}, c''_{m_2-1}, \dots, c''_0)$ , where there are  $r$  0s in the middle. Next, let  $v_0, \dots, v_{m_1}, v''_0, \dots, v''_{m_2} \in G$  and  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  be a level  $x$  shard of  $W_{(c'''_0, \dots, c'''_{m_1+m_2+r+1})[r]}((v_0, \dots, v_m, u_1, u_2, \dots, u_r, v''_m, v''_{m-1}, \dots, v''_0))$ . Define  $w$  and  $e$  such that there are  $m' + 1 - w$  distinct vertices in  $v'_0, \dots, v'_{m'}$  and  $m' - e$  distinct unordered pairs  $\{v'_i, v'_{i+1}\}$ , and let  $m'' = m' - r - 1$ . Then  $m_1 + m_2 - m'' \leq x + 6(w - e) + 2 + e/r$ . Also, if  $(\Lambda/|\lambda_s|)^{l-s+1} \geq 2^{s-1}$  and  $\lambda_s^2 > \Lambda$ , then

$$\begin{aligned} & |\mathbb{E}(W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'})))| \\ & \leq n^{e-m''} (2^{s-1} k / \min p_i)^{3w-3e+2} |\lambda_s|^{m''} (2\Lambda/|\lambda_s|)^{(s-1)m''/l} \\ & \cdot (\Lambda/\lambda_s^2)^e ((\Lambda/|\lambda_s|)^{l-s+1} 2^{1-s})^{\frac{(3w-3e+2)(l+2r+1)}{l}} (\Lambda/|\lambda_s|)^{2(l-s+1)x} \end{aligned}$$

and also

$$\begin{aligned} & |\mathbb{E}(W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'})))| \\ & \leq n^{e-m''} (k/(\min p_i))^{3(w-e)+2} 2^{(2x+3(w-e)+2)(s-1)} \left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{m''/l} \\ & \cdot \left( \max_{s \leq j \leq h} (\lambda_j^2/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} (\lambda_j - \lambda'_i)^2/\Lambda \right)^{-e/l} \\ & \cdot \left( \max_{s \leq j \leq h} (|\lambda_j|/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i|/\Lambda \right)^{-\frac{(3w-3e+2)(l+2r+1)}{l} - 2x}. \end{aligned}$$

*Proof.* First, consider the standard decomposition of  $(v'_0, \dots, v'_{m'})$ . The only vertices that could have been deleted are vertices within  $r$  of the edge of a fresh segment with nonzero corresponding values of  $c'''_i$ , vertices that are not in a fresh segment with nonzero corresponding values of  $c'''_i$ , and the  $x$  vertices that were deleted as copies of vertices in  $v'_0, \dots, v'_{m'}$ . There are at most  $2e$  indices  $i$  such that  $(v'_i, v'_{i+1})$  is not in a fresh segment and there are no  $i$  and  $i'$  such that  $|i - i'| \leq 2r$ ,  $c'''_i \neq 0$ , and  $c'''_{i'} \neq 0$ . So, there are at most  $6(w - e) + 2 + e/r$  vertices with nonzero  $c'''_i$  that are not in a fresh segment and farther than  $r$  from its edge.

Now, remove  $u_1, \dots, u_r$  from whichever fresh segment they are in, splitting it into two fresh segments on either side of  $u_1, \dots, u_r$  if necessary. Let  $d, d', t, t'$  be the measures of the resulting set of

fresh segments, and  $t'' = \lceil \frac{t-d(l+2r+1)}{l} \rceil$ . By the walk decomposition lemma,  $d + d' \leq 3(w - e) + 2$ . Also, note that every edge in one of the repeated fresh segments is repeated later in the walk. So,  $t + 2t' \leq m''$ . Every edge except those that involve the  $u_i$  appears exactly once in a fresh segment, so  $t + t' = m'' - e$  and  $t \geq m'' - 2e$ . On another note, for every  $j < s$ , we have that

$$\begin{aligned} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| &\leq \Lambda^{l-s+1} |\lambda_j - \lambda'_j| (2\Lambda)^{s-2} \\ &\leq |\lambda_s|^{l-s+1} (\min_i |\lambda_s - \lambda_i|/2)^{s-1} \leq |\lambda_s|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_s - \lambda'_i|. \end{aligned}$$

Also, for every  $j \geq s$ , we have that

$$|\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \leq |\lambda_s|^{l-s+1} (2\Lambda)^{s-1}.$$

That means that if  $(\Lambda/|\lambda_s|)^{l-s+1} \geq 2^{s-1}$  and  $\lambda_s^2 > \Lambda$  then

$$\begin{aligned} &|\mathbb{E}(W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'})))| \\ &\leq n^{-t-t'} k^{d+d'} (\min p_i)^{-d-d'} \lambda_1^{t'+\max(t-(2x+d)(s-1)-l \cdot \max(t''-2x, 0), 0)} \\ &\quad \cdot (\lambda_1 + \Lambda)^{t-\max(t-(2x+d)(s-1)-l \cdot \max(t''-2x, 0), 0)-l \cdot \max(t''-2x, 0)} \\ &\quad \cdot \left( \max_{1 \leq j \leq k} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{\max(t''-2x, 0)} \\ &\leq n^{e-m''} k^{d+d'} (\min p_i)^{-d-d'} (2\Lambda)^{(2x+d)(s-1)} \\ &\quad \cdot \Lambda^{m''-e-(2x+d)(s-1)-l \cdot \max(t''-2x, 0)} \left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{\max(t''-2x, 0)} \\ &\leq n^{e-m''} (k/(\min p_i))^{3(w-e)+2} 2^{(2x+3(w-e)+2)(s-1)} \Lambda^{m''-e} \\ &\quad \cdot \left( \max_{s \leq j \leq h} (|\lambda_j|/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i|/\Lambda \right)^{\frac{m''-2e-(3w-3e+2)(l+2r+1)}{l}-2x} \\ &= n^{e-m''} (k/(\min p_i))^{3(w-e)+2} 2^{(2x+3(w-e)+2)(s-1)} \left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{m''/l} \\ &\quad \cdot \left( \max_{s \leq j \leq h} (\lambda_j^2/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} (\lambda_j - \lambda'_i)^2/\Lambda \right)^{-e/l} \\ &\quad \cdot \left( \max_{s \leq j \leq h} (|\lambda_j|/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i|/\Lambda \right)^{-\frac{(3w-3e+2)(l+2r+1)}{l}-2x}. \end{aligned}$$

Alternately,

$$\begin{aligned}
& |\mathbb{E}(W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'})))| \\
& \leq n^{-t-t'} k^{d+d'} (\min p_i)^{-d-d'} \lambda_1^{t'+\max(t-(2x+d)(s-1)-l \cdot \max(t''-2x, 0), 0)} \\
& \cdot (\lambda_1 + \Lambda)^{t-\max(t-(2x+d)(s-1)-l \cdot \max(t''-2x, 0), 0)-l \cdot \max(t''-2x, 0)} \\
& \cdot \left( \max_{1 \leq j \leq k} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{\max(t''-2x, 0)} \\
& \leq n^{e-m''} k^{d+d'} (\min p_i)^{-d-d'} 2^{(2x+d)(s-1)} \Lambda^{m''-e} ((|\lambda_s|/\Lambda)^{l-s+1} 2^{s-1})^{t''-2x} \\
& \leq n^{e-m''} k^{d+d'} (\min p_i)^{-d-d'} 2^{d(s-1)} \Lambda^{m''-e} ((|\lambda_s|/\Lambda)^{l-s+1} 2^{s-1})^{\frac{m''-2e-d(l+2r+1)}{l}} (\Lambda/|\lambda_s|)^{2(l-s+1)x} \\
& \leq n^{e-m''} k^{d+d'} (\min p_i)^{-d-d'} 2^{d(s-1)} \\
& \cdot (|\lambda_s|^{l-s+1} (2\Lambda)^{s-1})^{m''/l} ((\Lambda/\lambda_s^2)^{l-s+1} (1/4\Lambda)^{s-1})^{e/l} \\
& \cdot ((|\lambda_s|/\Lambda)^{l-s+1} 2^{s-1})^{-\frac{d(l+2r+1)}{l}} (\Lambda/|\lambda_s|)^{2(l-s+1)x} \\
& \leq n^{e-m''} k^{d+d'} (\min p_i)^{-d-d'} 2^{d(s-1)} |\lambda_s|^{m''} ((2\Lambda/|\lambda_s|)^{s-1})^{m''/l} \\
& \cdot (\Lambda/\lambda_s^2)^e ((|\lambda_s|/\Lambda)^{l-s+1} 2^{s-1})^{-\frac{d(l+2r+1)}{l}} (\Lambda/|\lambda_s|)^{2(l-s+1)x} \\
& \leq n^{e-m''} (2^{s-1} k / \min p_i)^{3w-3e+2} |\lambda_s|^{m''} (2\Lambda/|\lambda_s|)^{(s-1)m''/l} \\
& \cdot (\Lambda/\lambda_s^2)^e ((\Lambda/|\lambda_s|)^{l-s+1} 2^{1-s})^{\frac{(3w-3e+2)(l+2r+1)}{l}} (\Lambda/|\lambda_s|)^{2(l-s+1)x}.
\end{aligned}$$

□

**Definition 25.**  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  is a degree  $z$  shard for  $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ ,  $m_1, m_2, r, (c_0, \dots, c_{m_1})$ , and  $(c''_0, \dots, c''_{m_2})$  if there exists  $x \in \mathbb{Z}$  and  $v_0, \dots, v_{m_1}, v''_0, \dots, v''_{m_2}$  such that  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  is a level  $x$  shard of

$W_{(c_0, \dots, c_{m_1}, 0, \dots, 0, c''_{m_2}, c''_{m_2-1}, \dots, c''_0)[r]}((v_0, \dots, v_{m_1}, u_1, \dots, u_r, v''_{m_2}, v''_{m_2-1}, \dots, v''_0))$ , the number of distinct sets  $\{v'_i, v'_{i+1}\}$  minus the number of distinct vertices in  $v'_0, \dots, v'_{m'}$  is  $z - x - 1$ , and  $v_0, v_{m_1}, v''_0$ , and  $v''_{m_2}$  are in communities  $\sigma_1, \sigma_2, \sigma_3$ , and  $\sigma_4$  respectively.

**Lemma 16** (Degree bound lemma). For any functions  $m_1$  and  $m_2$  of  $n$  such that  $m_1, m_2 = O(\log n)$ , there exists  $r_0$  such that for any  $r \geq r_0$ ,  $z \geq 0$ , and  $\Lambda, s, l$  such that  $(\Lambda/|\lambda_s|)^{l-s+1} \geq 2^{s-1}$ ,  $\lambda_s^2 > \Lambda > \lambda_1$ ,  $l \geq (2r+1)(s-1)$ , and  $\lambda_s^{2l-2s+2} \prod_{i=1}^{s-1} (\lambda_s - \lambda'_i)^2 > (2\Lambda)^{2s-2} \lambda_1^{l-s+1}$  the following holds. Let  $\lambda'_1, \dots, \lambda'_{s-1}$  be a  $\Lambda$ -bounded eigenvalue approximation and  $c_0, \dots, c_{m_1}$  and  $c'_0, \dots, c'_{m_2}$  be  $l[r]$ -cycles of this approximation with error less than  $2\Lambda(\min_i |\lambda_s - \lambda'_i|/4\Lambda)^{s-1} |\lambda_s/\Lambda|^{l-s+1}$ . For any  $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ , the expected value of the sum of the absolute values of all shards of degree at least  $z$  for  $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ ,

$m_1, m_2, r, (c_0, \dots, c_{m_1})$ , and  $(c''_0, \dots, c''_{m_2})$  times their coefficients is  $O(n^{(15-5z)/6} \prod |\lambda_s - c_i| \cdot \prod |\lambda_s - c''_i|)$ .

*Proof.* Let  $m = m_1 + m_2$ . For any given  $x, w, m'$ , and  $e$  with  $x + w - e > z$  and  $x \leq m_1 + m_2 + r + 1 - m' \leq x + 6(w - e) + 2 + e/r$ , the expected sum of the values of all level  $x$  shards for  $(\sigma_{v_0}, \sigma_{v_{m_1}}, \sigma_{v''_0}, \sigma_{v''_{m_2}})$ ,  $m_1, m_2, r, (c_0, \dots, c_{m_1})$ , and  $(c''_0, \dots, c''_{m_2})$  with  $m' + 1$  vertices,  $m' + 1 - w$  distinct vertices, and  $m' - e$  distinct adjacent pairs of vertices has an absolute value of at most

$$\begin{aligned}
& \sum_{(v'_0, \dots, v'_{m'}) : |\{v' \in (v'_0, \dots, v'_{m'})\}| = m' + 1 - e, |\{v'_i, v'_{i+1}\} : 0 \leq i < m'\}| = m' - e, (c'_0, \dots, c'_{m'}) \in \mathbb{R}^{m'+1}} (\Lambda/n)^{m+r+1-m'} \\
& \cdot |\mathbb{E}[W_{(c'_0, \dots, c'_{m'})}(v'_0, \dots, v'_{m'})]| \\
& \cdot |\{(v_1, \dots, v_{m_1-1}, v''_1, \dots, v''_{m_2-1}) : W_{(c'_0, \dots, c'_{m'})}(v'_0, \dots, v'_{m'}) \\
& \in \text{Shar}_x(W_{(c_0, \dots, c_{m_1}, 0, \dots, 0, c''_0, \dots, c''_{m_2})[r]}((v_0, \dots, v_{m_1}, u_1, \dots, u_r, v''_{m_2}, \dots, v''_0)))\}| \\
& \leq [2n^{m'-w+1}(m+r+2)^{4(w-e)} 2^{2\lceil \frac{m'+1}{r+1} \rceil (w-e)}] \cdot (\Lambda/n)^{m+r+1-m'} \\
& \cdot [3^{m/r}(m+2)^x n^{m+1-m'-x}] \cdot [n^{e-m'+r+1}(2^{s-1}k/\min p_i)^{3w-3e+2} |\lambda_s|^{m'-r-1} \\
& \cdot (2\Lambda/|\lambda_s|)^{(s-1) \cdot (m'-r-1)/l} \cdot (\Lambda/\lambda_s^2)^e ((\Lambda/|\lambda_s|)^{l-s+1} 2^{1-s})^{\frac{(3w-3e+2)(l+2r+1)}{l}} (\Lambda/|\lambda_s|)^{2(l-s+1)x}] \\
& \leq n^{e-w+13/6-x} 2^{2\lceil \frac{m'+1}{r+1} \rceil (w-e)} (m+r+2)^{4(w-e)+x} \Lambda^{m+r+1-m'} (\Lambda/\lambda_s^2)^e \\
& 2(2^{s-1}k/\min p_i)^{3w-3e+2} |\lambda_s|^{m'-r-1} (2\Lambda/|\lambda_s|)^{(s-1) \cdot (m'-r-1)/l} \\
& \cdot ((\Lambda/|\lambda_s|)^{l-s+1} 2^{1-s})^{\frac{(3w-3e+2)(l+2r+1)}{l}} (\Lambda/|\lambda_s|)^{2(l-s+1)x} \\
& = |\lambda_s|^m (2\Lambda/|\lambda_s|)^{(s-1)m/l} \cdot n^{e-w+13/6-x} 2^{2\lceil \frac{m'+1}{r+1} \rceil (w-e)} \\
& \cdot 2(m+r+2)^{4(w-e)+x} (\Lambda/\lambda_s^2)^e (2^{s-1}k/\min p_i)^{3w-3e+2} (\Lambda/|\lambda_s| (2\Lambda/|\lambda_s|)^{-(s-1)/l})^{m+r+1-m'} \\
& \cdot ((\Lambda/|\lambda_s|)^{l-s+1} 2^{1-s})^{\frac{(3w-3e+2)(l+2r+1)}{l}} (\Lambda/|\lambda_s|)^{2(l-s+1)x}
\end{aligned}$$

and the last term above is upper-bounded by

$$\begin{aligned}
& |\lambda_s|^m (2\Lambda/|\lambda_s|)^{m(s-1)/l} n^{e-w+13/6-x} 2^{2\lceil \frac{m+r+2}{r+1} \rceil (w-e)} \\
& \cdot 2(m+r+2)^{4(w-e)+x} (\Lambda/\lambda_s^2)^e (2^{s-1}k/\min p_i)^{3w-3e+2} (\Lambda/|\lambda_s|(2\Lambda/|\lambda_s|)^{-(s-1)/l})^{x+6(w-e)+2+e/r} \\
& \cdot ((\Lambda/|\lambda_s|)^{l-s+1} 2^{1-s})^{\frac{(3w-3e+2)(l+2r+1)}{l}} (\Lambda/|\lambda_s|)^{2(l-s+1)x} \\
& \leq 2|\lambda_s|^m (2\Lambda/|\lambda_s|)^{m(s-1)/l} (2^{s-1}k/\min p_i)^2 (\Lambda/|\lambda_s|(2\Lambda/|\lambda_s|)^{-(s-1)/l})^2 \\
& ((\Lambda/|\lambda_s|)^{l-s+1} 2^{1-s})^{\frac{2(l+2r+1)}{l}} n^{13/6} \cdot \left( (m+r+2)(\Lambda/|\lambda_s|(2\Lambda/|\lambda_s|)^{-(s-1)/l})(\Lambda/|\lambda_s|)^{2(l-s+1)}/n \right)^x \\
& \cdot \left( 2^{2\lceil \frac{m+r+2}{r+1} \rceil} (m+r+2)^4 \left( \frac{2^{s-1}k}{\min p_i} \right)^3 \left( \frac{\Lambda}{|\lambda_s|} (2\Lambda/|\lambda_s|)^{-(s-1)/l} \right)^6 ((\Lambda/|\lambda_s|)^{l-s+1} 2^{1-s})^{\frac{3(l+2r+1)}{l}} /n \right)^{w-e} \\
& \cdot \left( \frac{\Lambda^r}{\lambda_s^{2r}} \cdot \frac{\Lambda}{|\lambda_s|} (2\Lambda/|\lambda_s|)^{-(s-1)/l} \right)^{e/r} \\
& \leq |\lambda_s|^m n^{(14-5z)/6}
\end{aligned}$$

provided that  $r > 12m/\log_2(n)$  and  $n$  is sufficiently large. There are at most  $2(s-1) + (s-1)m/l$  indices  $i$  such that  $c_i \neq 0$  or  $c'_i \neq 0$ , so  $|\lambda_s|^{-m} \prod |\lambda_s - c_i| \prod |\lambda_s - c'_i| \geq (\min_{s' < s} |\lambda_s - \lambda_{s'}|/2)^{(s-1)m/l}$ . So, as long as  $r_0$  is large enough that  $\frac{m}{2r+1} \ln(\min_{s' < s} |\lambda_s - \lambda_{s'}|/2) < \ln(n)/3$ , then  $|\lambda_s|^m n^{(14-5z)/6} \leq n^{(15-5z)/6} \prod |\lambda_s - c_i| \cdot \prod |\lambda_s - c'_i|$ .  $\square$

**Lemma 17.** Assume that  $m, \Lambda, \lambda'_1, \dots, \lambda'_{s-1}, l, r$ , and  $(c_0, \dots, c_m)$  satisfy the conditions of the degree bound lemma. Also assume that  $m = \Omega(\log n)$  and either  $s = h'$  or  $|\lambda_s| > |\lambda_{s+1}|$ . Now, let  $w$  be an eigenvector of  $PQ$  with an eigenvalue of  $\lambda_j$ . If  $j \neq s$  then with probability  $1 - o(1)$  the average value over all  $v$  of  $w_{\sigma_v} W_{m/\{c_i\}}(x, v)/p_{\sigma_v}$  is  $O(\frac{1}{\ln(n)\sqrt{n}} \prod |\lambda_s - c_i|)$  and if  $j = s$  then the average value over all  $v$  of  $w_{\sigma_v} W_{m/\{c_i\}}(x, v)/p_{\sigma_v}$  is  $\Theta(\frac{1}{\sqrt{n}} \prod |\lambda_j - c_i|)$  with probability  $1 - o(1)$ .

*Proof.* Let  $m''' = \lfloor \sqrt{\ln(n)} \rfloor$ , unless  $c_{\lfloor \sqrt{\ln(n)} \rfloor} \neq 0$ , in which case let  $m''' = \lfloor \sqrt{\ln(n)} \rfloor + 1$ . The first step in proving the desired bounds is to justify approximating  $\sum_{v \in \Omega_j} W_{m/\{c_i\}}(x, v)$  with  $\sum_{v_0, \dots, v_{m'''} \in G} x_{v_0} \cdot W_{(c_0, \dots, c_{m'''})[r]}((v_0, \dots, v_{m'''}) e_{\sigma_{v_{m'''}}}) \cdot \prod_{i=m'''+1}^m (PQ - c_i) e_j$ . In order to do that, we

observe that for any  $j$ ,

$$\begin{aligned}
& E \left[ \left( \sum_{v \in \Omega_j} W_{m/\{c_i\}}(x, v) - \sum_{v_0, \dots, v_{m'''} \in G} x_{v_0} \cdot W_{(c_0, \dots, c_{m'''})[r]}((v_0, \dots, v_{m'''})e_{\sigma_{v_{m'''}}} \cdot \prod_{i=m'''+1}^m (PQ - c_i)e_j) \right)^2 \right] \\
&= E \left[ \left( \sum_{v_0, \dots, v_m \in G: v_m \in \Omega_j} x_{v_0} W_{(c_0, \dots, c_m)[r]}(v_0, \dots, v_m) \right. \right. \\
&\quad \left. \left. - x_{v_0} \cdot W_{(c_0, \dots, c_{m'''})[r]}((v_0, \dots, v_{m'''})e_{\sigma_{v_{m'''}}} \cdot \prod_{i=m'''+1}^m \frac{1}{n}(PQ - c_i)e_j)^2 \right] \right. \\
&= E \left[ \left( \sum_{v_0, \dots, v_m, v'_0, \dots, v'_m \in G: v_m, v'_m \in \Omega_j} x_{v_0} x_{v'_0} W_{(c_0, \dots, c_m, 0, \dots, 0, c_m, \dots, c_0)[r]}(v_0, \dots, v_m, u_1, \dots, u_r, v'_m, \dots, v'_0) \right. \right. \\
&\quad \left. \left. - x_{v_0} x_{v'_0} W_{(c_0, \dots, c_m, 0, \dots, 0, c_{m'''}, \dots, c_0)[r]}(v_0, \dots, v_m, u_1, \dots, u_r, v'_{m'''}, \dots, v'_0) e_{\sigma_{v'_{m'''}}} \cdot \prod_{i=m'''+1}^m \frac{1}{n}(PQ - c_i)e_j \right. \right. \\
&\quad \left. \left. - x_{v_0} x_{v'_0} W_{(c_0, \dots, c_{m'''}, 0, \dots, 0, c_m, \dots, c_0)[r]}(v_0, \dots, v_{m'''}, u_1, \dots, u_r, v'_m, \dots, v'_0) e_{\sigma_{v_{m'''}}} \cdot \prod_{i=m'''+1}^m \frac{1}{n}(PQ - c_i)e_j \right. \right. \\
&\quad \left. \left. + x_{v_0} x_{v'_0} W_{(c_0, \dots, c_{m'''}, 0, \dots, 0, c_{m'''}, \dots, c_0)[r]}(v_0, \dots, v_{m'''}, u_1, \dots, u_r, v'_{m'''}, \dots, v'_0) \right. \right. \\
&\quad \left. \left. \left( e_{\sigma_{v_{m'''}}} \cdot \prod_{i=m'''+1}^m \frac{1}{n}(PQ - c_i)e_j \right) \left( e_{\sigma_{v'_{m'''}}} \cdot \prod_{i=m'''+1}^m \frac{1}{n}(PQ - c_i)e_j \right) \right) \right]
\end{aligned}$$

Note that  $E[v_0 v'_0]$  is 1 if  $v_0 = v'_0$  and 0 otherwise. So, the only  $(v_0, \dots, v_m), (v'_0, \dots, v'_m)$  that make a nonzero contribution to the expected value of the sum above are those for which  $v_0 = v'_0$ . Also, given any  $(v_0, \dots, v_m), (v'_0, \dots, v'_m)$  such that  $v_{m'''+1}, \dots, v_m$  are distinct, and  $\{v_{m'''+1}, \dots, v_m\} \cap (\{v_0, \dots, v_{m'''}\} \cup \{v'_0, \dots, v'_m\}) = \emptyset$ , then for any assignment of communities to  $v_{m'''}, v_m$ , and  $v'_m$ , and any possible value of the subgraph of  $G$  induced by  $\{v_0, \dots, v_{m'''}\} \cup \{v'_0, \dots, v'_m\}$ , the expected value of  $W_{(c_0, \dots, c_m)[r]}(v_0, \dots, v_m)$  given these values is  $W_{(c_0, \dots, c_{m'''})[r]}(v_0, \dots, v_{m'''})e_{\sigma_{v_{m'''}}} \cdot \prod_{i=m'''+1}^m (PQ - c_i)e_j$ . That means that the expected contribution to the sum above of the terms corresponding to such  $(v_0, \dots, v_m), (v'_0, \dots, v'_m)$  is 0. By the same logic, all  $(v_0, \dots, v_m), (v'_0, \dots, v'_m)$  such that  $v'_{m'''+1}, \dots, v'_m$  are distinct and  $\{v'_{m'''+1}, \dots, v'_m\} \cap (\{v'_0, \dots, v'_{m'''}\} \cup \{v_0, \dots, v_m\}) = \emptyset$  have an expected contribution of 0 to the sum above.

Now, let  $V$  be the set of all pairs of tuples  $((v_0, \dots, v_m), (v'_0, \dots, v'_m))$  such that  $v_0 = v'_0$ , either  $|\{v_{m'''+1}, \dots, v_m\}| < m - m'''$  or  $\{v_{m'''+1}, \dots, v_m\} \cap (\{v_0, \dots, v_{m'''}\} \cup \{v'_0, \dots, v'_m\}) \neq \emptyset$ , and either  $|\{v'_{m'''+1}, \dots, v'_m\}| < m - m'''$  or  $\{v'_{m'''+1}, \dots, v'_m\} \cap (\{v'_0, \dots, v'_{m'''}\} \cup \{v_0, \dots, v_m\}) \neq \emptyset$ . Note that for every  $v_0, \dots, v_{m'''} and  $v'_0, \dots, v'_m$ , there are at most  $(2m+2)^2 n^{m-m'''-1}$  possible choices of  $v_{m'''+1}, \dots, v_m$$

such that  $((v_0, \dots, v_m), (v'_0, \dots, v'_m)) \in V$ . So,

$$\begin{aligned}
& E[ \sum_{((v_0, \dots, v_m), (v'_0, \dots, v'_m)) \in V: v_m, v'_m \in \Omega_j} |x_{v_0} x_{v'_0} W_{(c_0, \dots, c_m, 0, \dots, 0, c_{m'''}, \dots, c_0)[r]}(v_0, \dots, v_m, u_1, \dots, u_r, v'_{m'''}, \dots, v'_0) \\
& \quad e_{\sigma_{v'_{m'''}}} \cdot \prod_{i=m'''+1}^m \frac{1}{n} (PQ - c_i) e_j |] \\
& \leq [(2m+2)^2 n^{m-m'''-1}] \cdot \left[ n^{m'''-m} / (\min_i p_i) \max_{1 \leq i' \leq h} \prod_{i=m'''+1}^m |\lambda_{i'} - c_i| \right] \\
& \quad \cdot E \left[ \sum_{v_0, \dots, v_m, v'_0, \dots, v'_{m'''} \in G: v_m, v'_m \in \Omega_j} |W_{(c_0, \dots, c_m, 0, \dots, 0, c_{m'''}, \dots, c_0)[r]}(v_0, \dots, v_m, u_1, \dots, u_r, v'_{m'''}, \dots, v'_0)| \right] \\
& = [(2m+2)^2 n^{m-m'''-1}] \cdot \left[ n^{m'''-m} / (\min_i p_i) \max_{1 \leq i' \leq h} \prod_{i=m'''+1}^m |\lambda_{i'} - c_i| \right] \\
& \quad \cdot O \left( n^{10/6} \prod_{i=1}^{m'''} |\lambda_s - c_i| \prod_{i=1}^m |\lambda_s - c_i| \right) \\
& = O \left( n^{5/6} \prod_{i=1}^m (\lambda_s - c_i)^2 \right)
\end{aligned}$$

where the second to last equality follows from the degree bound lemma and the fact that  $v_0 = v'_0$  implies that every shard of a  $W$  in this expression has degree at least 1. The last equality follows from the facts that  $\prod_{i=m'''+1}^m |\lambda_{i'} - c_i| = O(\prod_{i=m'''+1}^m |\lambda_s - c_i|)$  for any  $i'$  and  $m = O(\ln(n))$ . The same logic applies to the analogous expressions for  $W_{(c_0, \dots, c_{m'''}, 0, \dots, 0, c_m, \dots, c_0)[r]}(v_0, \dots, v_{m'''}, u_1, \dots, u_r, v'_m, \dots, v'_0)$  and

$W_{(c_0, \dots, c_{m'''}, 0, \dots, 0, c_{m'''}, \dots, c_0)[r]}(v_0, \dots, v_{m'''}, u_1, \dots, u_r, v'_{m'''}, \dots, v'_0)$ . This implies that

$$\begin{aligned}
& E \left[ \left( \sum_{v \in \Omega_j} W_{m/\{c_i\}}(x, v) - \sum_{v_0, \dots, v_{m'''} \in G} x_{v_0} \cdot W_{(c_0, \dots, c_{m'''}, 0, \dots, 0, c_m, \dots, c_0)[r]}((v_0, \dots, v_{m'''}) e_{\sigma_{v_{m'''}}} \cdot \prod_{i=m'''+1}^m (PQ - c_i) e_j) \right)^2 \right] \\
& = E \left[ \sum_{((v_0, \dots, v_m), (v'_0, \dots, v'_m)) \in V: v_m, v'_m \in \Omega_j} W_{(c_0, \dots, c_m, 0, \dots, 0, c_m, \dots, c_0)[r]}(v_0, \dots, v_m, u_1, \dots, u_r, v'_m, \dots, v'_0) \right] \\
& \quad + O \left( n^{5/6} \prod_{i=1}^m (\lambda_s - c_i)^2 \right)
\end{aligned}$$

If the  $W_{(c_0, \dots, c_m, 0, \dots, 0, c_m, \dots, c_0)[r]}(v_0, \dots, v_m, u_1, \dots, u_r, v'_m, \dots, v'_0)$  are broken down into weighted sums of shards, then every resulting shard has degree at least 1 for the same reason as in the previous cases, and all resulting shards of degree at least 2 have a combined contribution to the expected

value of at most  $n^{5/6} \prod_{i=1}^m (\lambda_s - c_i)^2$  by the degree bound lemma. Now, let  $W_{(c'_0, \dots, c'_{m'})[r]}(v''_0, \dots, v''_{m'})$  be a degree 1 shard of  $W_{(c_0, \dots, c_m, 0, \dots, 0, c_m, \dots, c_0)[r]}(v_0, \dots, v_m, u_1, \dots, u_r, v'_m, \dots, v'_0)$  for some  $((v_0, \dots, v_m), (v'_0, \dots, v'_m)) \in V$ . Also, let  $H$  be the walk graph of  $v''_0, \dots, v''_{m'}$  with  $u_1, \dots, u_r$  removed. We know that  $v''_0 = v_0 = v'_0 = v''_{m'}$ , so  $H$  is connected. That means that the only way that  $W_{(c'_0, \dots, c'_{m'})[r]}(v''_0, \dots, v''_{m'})$  can be a degree 1 shard is if  $H$  is a tree and no vertex that was deleted in the process of converting  $W_{(c_0, \dots, c_m, 0, \dots, 0, c_m, \dots, c_0)[r]}(v_0, \dots, v_m, u_1, \dots, u_r, v'_m, \dots, v'_0)$  to  $W_{(c'_0, \dots, c'_{m'})[r]}(v''_0, \dots, v''_{m'})$  is repeated in  $v''_0, \dots, v''_{m'}$ . So, there must exist some  $t_0 \geq 0$  such that  $v''_i = v''_{m'-i}$  for all  $i \leq t_0$ ,  $v''_{t_0+1}, \dots, v''_{m'-t_0-1}$  are distinct, and  $\{v''_0, \dots, v''_{t_0}\} \cap \{v''_{t_0+1}, \dots, v''_{m'-t_0-1}\} = \emptyset$ . Furthermore, since  $((v_0, \dots, v_m), (v'_0, \dots, v'_m)) \in V$ , there exist  $i$  and  $i'$  such that  $i > m''$  and  $v_i = v'_{i'}$  or  $v_i = v_{i'}$  with  $i \neq i'$ . Either neither of the repeated elements were deleted in the process of converting  $W_{(c_0, \dots, c_m, 0, \dots, 0, c_m, \dots, c_0)[r]}(v_0, \dots, v_m, u_1, \dots, u_r, v'_m, \dots, v'_0)$  to  $W_{(c'_0, \dots, c'_{m'})[r]}(v''_0, \dots, v''_{m'})$  or they both were, in which case  $v_i$  must have been within  $r$  of a repeated vertex that was not deleted during the conversion. The only vertices that could have been deleted during the conversion are those with nonzero corresponding  $c_i$ , so either way, we have that  $t_0 \geq \frac{2r}{2r+1}m'' - r$ . Also, by lemma 15, we have that

$$\begin{aligned}
& |E[W_{(c'_0, \dots, c'_{m'})[r]}(v''_0, \dots, v''_{m'})]| \\
& \leq n^{t_0 - m' + r + 1} (k / (\min p_i))^{5 \cdot 2^{5(s-1)}} \left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{(m'-r-1)/l} \\
& \cdot \left( \max_{s \leq j \leq h} (\lambda_j^2 / \Lambda)^{l-s+1} \prod_{i=1}^{s-1} (\lambda_j - \lambda'_i)^2 / \Lambda \right)^{-t_0/l} \\
& \cdot \left( \max_{s \leq j \leq h} (|\lambda_j| / \Lambda)^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| / \Lambda \right)^{-\frac{5(l+2r+1)}{l}}.
\end{aligned}$$

For fixed values of  $t_0$  and  $m'$ , there are at most  $2m \cdot n^{(m'-t_0-r)}$  possible values of  $(v''_0, \dots, v''_{m'})$ . Furthermore, for each possible value of  $(v''_0, \dots, v''_{m'})$ , there are at most  $2^{t_0/r+1} n^{2m+r+1-m'}$  possible values of  $v_0, \dots, v_m, v'_0, \dots, v'_m$ , and  $c'_0, \dots, c'_{m'}$  such that  $W_{(c'_0, \dots, c'_{m'})[r]}(v''_0, \dots, v''_{m'})$  is a degree 1 shard of  $W_{(c_0, \dots, c_m, 0, \dots, 0, c_m, \dots, c_0)[r]}(v_0, \dots, v_m, u_1, \dots, u_r, v'_m, \dots, v'_0)$ , and this shard has a coefficient with an absolute value of at most  $(\Lambda/n)^{2m+r+1-m'}$ . So, the combined contribution to the expected value

from all degree 1 shards with a given value of  $t_0$  and  $m'$  is at most

$$\begin{aligned}
& 2^{t_0/r+2} m \Lambda^{2m+r+1-m'} n(2^{s-1}k/(\min p_i))^5 \left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{(m'-r-1)/l} \\
& \cdot \left( \max_{s \leq j \leq h} (\lambda_j^2/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} (\lambda_j - \lambda'_i)^2/\Lambda \right)^{-t_0/l} \\
& \cdot \left( \max_{s \leq j \leq h} (|\lambda_j|/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i|/\Lambda \right)^{-\frac{5(l+2r+1)}{l}} \\
& \leq 2^{2m''/(2r+1)+1} m \Lambda^{2m''/(2r+1)} n(2^{s-1}k/(\min p_i))^5 \left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{(2m-2m''/(2r+1))/l} \\
& \cdot \left( \max_{s \leq j \leq h} (\lambda_j^2/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} (\lambda_j - \lambda'_i)^2/\Lambda \right)^{-(\frac{2r}{2r+1}m''-r)/l} \\
& \cdot \left( \max_{s \leq j \leq h} (|\lambda_j|/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i|/\Lambda \right)^{-\frac{5(l+2r+1)}{l}} \\
& = O \left( n \ln^{-5}(n) \prod_{i=0}^m (\lambda_s - c_i)^2 \right)
\end{aligned}$$

There are only  $O(\ln^2(n))$  possible values of  $t_0$  and  $m'$ , so the combined contribution of all degree 1 shards is  $O(n \ln^{-3}(n) \prod_{i=0}^m (\lambda_s - c_i)^2)$ . So, with probability  $1 - o(1)$ , the average value over all  $v \in \Omega_\sigma$  of  $W_{m/\{c_i\}}(x, v)$  is within  $o(\frac{1}{\log(n)\sqrt{n}} \prod |\lambda_s - c_i|)$  of

$$\frac{1}{p_\sigma} e_\sigma \cdot \prod_{i=m''+1}^m (PQ - c_i) \sum_{(v_0, \dots, v_{m''})} x_{v_0} \cdot W_{(c_0, \dots, c_{m''})[r]}((v_0, \dots, v_{m''})) e_{\sigma_{v_{m''}}}$$

If  $w$  is an eigenvector of  $PQ$  with eigenvalue  $\lambda_j$ , then this means that the average value of  $w_{\sigma_v} W_{m/\{c_i\}}(x, v)/p_{\sigma_v}$  is within  $o(\frac{1}{\log(n)\sqrt{n}} \prod |\lambda_s - c_i|)$  of

$$\frac{1}{n} \prod_{i=m''+1}^m (\lambda_j - c_i) \sum_{(v_0, \dots, v_{m''})} x_{v_0} \cdot W_{(c_0, \dots, c_{m''})[r]}((v_0, \dots, v_{m''})) w_{\sigma_{v_{m''}}}/p_{\sigma_{v_{m''}}}$$

For fixed  $G$  and  $\sigma$  but not fixed  $x$ , the probability distribution of this expression is a normal distribution with mean 0 and variance

$$\frac{1}{n^2} \prod_{i=m''+1}^m (\lambda_j - c_i)^2 \sum_{v_0} \left( \sum_{(v_1, \dots, v_{m''})} W_{(c_0, \dots, c_{m''})[r]}((v_0, \dots, v_{m''})) w_{\sigma_{v_{m''}}}/p_{\sigma_{v_{m''}}} \right)^2$$

With probability  $1 - o(1)$ , there is no  $v_0 \in G$  that has more than one cycle within  $m'''$  edges

of it, there are  $O(\lambda_1^{2m'''})$  vertices that have a cycle within  $m'''$  edges of them, and no vertex in the graph has degree greater than  $\ln^2(n)$ . This implies that

$$\sum_{v_0} \left( \sum_{(v_1, \dots, v_{m'''})} W_{(c_0, \dots, c_{m'''})[r]}((v_0, \dots, v_{m'''})) w_{\sigma_{v_{m''}}} / p_{\sigma_{v_{m''}}} \right)^2 = O(n \ln^{2m'''}(n))$$

If  $j \neq s$ , then there exists  $\epsilon > 0$  such that  $\prod_{i=m''+1}^m (\lambda_j - c_i)^2 = O(n^{-\epsilon} \prod_{i=m''+1}^m (\lambda_s - c_i)^2)$ , so the variance is  $o(\frac{1}{n \log^2(n)} \prod (\lambda_s - c_i)^2)$ . Thus, the specified average is  $O(\frac{1}{\log(n)\sqrt{n}} \prod |\lambda_s - c_i|)$  with probability  $1 - o(1)$ , as desired.

Now, consider the case where  $j = s$ . If there is no cycle in the portion of the graph within  $m''$  edges of  $v_0$  then every nonbacktracking walk of length  $m''$  or less starting at  $v_0$  is a path. So, conditioned on the absence of such cycles near  $v_0$ , we have that

$$\begin{aligned} E & \left[ \left( \sum_{(v_1, \dots, v_{m''})} W_{(c_0, \dots, c_{m''})[r]}((v_0, \dots, v_{m''})) w_{\sigma_{v_{m''}}} / p_{\sigma_{v_{m''}}} \right)^2 \right] \\ &= \Theta \left( \sum_{j=1}^{m'''-1} \lambda_1^j \prod_{i=1}^{m'''-1} (\lambda_s - c_i)^2 / \lambda_s^{2i} \right) + \Theta \left( m''' \lambda_1^{m'''-1-|\{i:c_i \neq 0\}|} \prod_{i:c_i \neq 0} (|\lambda_s| + |c_i|)^2 \right) \\ &= \Theta \left( \prod_{i=1}^{m'''-1} (\lambda_s - c_i)^2 \right) \end{aligned}$$

Furthermore, for any  $v$  and  $v'$  there is no vertex within  $m''$  edges of both  $v$  and  $v'$  with probability  $1 - O(\lambda_1^{2m''}/n)$ . So, the joint probability distribution of the subgraphs of  $G$  within  $m''$  edges of  $v$  and  $v'$  differs from the product of the individual distributions by  $O(\lambda_1^{2m''}/n)$ . So, with probability  $1 - o(1)$  we have that

$$\sum_{v_0} \left( \sum_{(v_1, \dots, v_{m''})} W_{(c_0, \dots, c_{m''})[r]}((v_0, \dots, v_{m''})) w_{\sigma_{v_{m''}}} / p_{\sigma_{v_{m''}}} \right)^2 = \Theta \left( n \prod_{i=1}^{m'''-1} (\lambda_s - c_i)^2 \right).$$

Therefore, with probability  $1 - o(1)$ , the average value over all  $v$  of  $w_{\sigma_v} W_{m/\{c_i\}}(x, v) / p_{\sigma_v}$  is  $\frac{1}{\sqrt{n}} \Theta(\prod |\lambda_s - c_i|)$ .  $\square$

**Lemma 18.** *There exists a constant  $r_0$  and  $m_0 = \Theta(\log n)$  such that if  $m$ ,  $\Lambda$ ,  $\lambda'_1, \dots, \lambda'_{s-1}$ ,  $l$ ,  $r$ , and  $(c_0, \dots, c_m)$  satisfy the conditions of the degree bound lemma,  $r > r_0$  and  $m > m_0$ , then for any*

communities  $\sigma_1, \sigma_2$ ,

$$\begin{aligned} & \left| \mathbb{E} \left[ \sum_{\substack{v_0, \dots, v_m, v'_0, \dots, v'_m \in G: \\ v_m = v'_m, v_0 = v'_0, \sigma v_m = \sigma_1, \sigma v_0 = \sigma_2}} W_{(c_0, \dots, c_m)[r]}((v_0, \dots, v_m)) \cdot W_{(c_0, \dots, c_m)[r]}((v'_0, \dots, v'_m)) \right] \right| \\ &= O \left( \left( \sum_{i=s}^h \prod_{j=0}^m |\lambda_i - c_j| \right)^2 \right) \end{aligned}$$

*Proof.* Let  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  be a level  $x$  shard of

$$W_{(c_0, \dots, c_m, 0, \dots, 0, c_m, \dots, c_0)[r]}((v_0, \dots, v_m, u_1, \dots, u_r, v''_m, \dots, v''_0)),$$

and  $G'$  be  $(v'_0, \dots, v'_{m'})$ 's walk graph.  $c_0 = c'_0 = 0$ , so  $v'_0 = v'_{m'}$  is the only vertex in  $G'$  that can have degree 1. Since  $c_m = c'_m = 0$ ,  $v_m \in G$  and it is adjacent to  $u_1$ ,  $u_r$ , and  $V_{m-1}$ , meaning that  $v_m$  has degree greater than 2 and this shard has nonzero degree.

By the degree bound lemma, the contribution to this expected value of all shards of degree greater than 2 is  $O(\prod(\lambda_s - c_i)^2)$ . Now, assume that this shard has degree less than 3, and let  $G''$  be  $G'$  with  $u_1, \dots, u_r$  and all of their edges removed.  $G''$  is still connected, and the only vertices that might have degree 1 in it are  $v_0$  and  $v_m$ . First, consider the case in which  $G''$  is a tree. It must be a path, so  $v'_0, \dots, v'_{m'}$  simply consists of a path, followed by a cycle, followed by the same path in reverse. There are at most  $m/r$  nonzero elements of  $(c_0, \dots, c_m, 0, \dots, 0, c_m, \dots, c_0)$ , so if  $I$  is the set of indices of vertices in  $(v_0, \dots, v_m, u_1, \dots, u_r, v''_m, \dots, v''_0)$  that were deleted in the process of converting it to this shard, there are at most  $2^{m/r}$  possible values of  $I$ . Also,  $m' = 2m + r + 1 - |I|$  and for a given value of  $I$ , there are at most  $n^{(2m+2-|I|)/2}$  possible values of  $(v'_0, \dots, v'_{m'})$ , each of which could have come from at most  $n^{|I|}$  possible values of  $(v_0, \dots, v_m)$  and  $(v'_0, \dots, v'_m)$ . The coefficient of the shard is at most  $(\Lambda/n)^{|I|}$ , and  $c'_i = 0$  for all  $i$ . That means that  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  is one if each of the  $m - |I|/2$  edges in  $G''$  are also in  $G$  and 0 otherwise. So,  $|\mathbb{E}[W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))]| = O((\lambda_1/n)^{m-|I|/2})$ . That means that the total contribution to the expected value of all such shards is  $O(2^{m/r} \Lambda^{m/r} \lambda_1^{m-m/2r} n)$ . So, as long as

$$m > \ln(n) / \left( \ln(\lambda_s^2/\lambda_1) - \ln(4\Lambda^2/\lambda_1)/2r + \frac{2(s-1)}{l} \ln \left( \min_{i < s} |\lambda_i - \lambda_s|/2|\lambda_s| \right) \right)$$

the contribution to the expected value of all such shards is  $O \left( \left( \sum_{i=s}^h \prod_{j=0}^m |\lambda_i - c_j| \right)^2 \right)$ , as desired.

That leaves the case where  $G''$  is not a tree. The fact that the shard has degree less than 3

implies that  $G'$  has at most 1 more edges than vertices, so  $G''$  has at least as many vertices as edges. So, if it is not a tree, it must have an equal number of edges and vertices, which means that it contains exactly one cycle. The only vertices in  $G''$  that might have degree 1 are  $v_0$  and  $v_m$ , so  $G''$  consists of a cycle and two paths (possibly of length 0) that connect  $v_0$  and  $v_m$  to the cycle. Now, let  $e$  be the length of the path from  $v_0$  to the cycle,  $e'$  be the length of the path from  $v_m$  to the cycle, and  $|I|$  be the set of all indices of vertices in  $(v_0, \dots, v_m, u_1, \dots, u_r, v_m'', \dots, v_0'')$  that were deleted in the process of converting it to  $(v_0', \dots, v_{m'}')$ .  $m' = 2m + r + 1 - |I|$ , and the shard will have a coefficient of magnitude at most  $(\Lambda/n)^{|I|}$ . There are a few subcases to consider.

1. No edge in the cycle in  $G''$  shows up more than once in the walk defined by  $(v_0', \dots, v_{m'}')$ :

In this case, there must be  $m' - 2e - 2e' - r - 1 = 2m - 2e - 2e' - |I|$  edges in the cycle. For given values of  $e$  and  $e'$ , there are at most  $(e + e')/r + 2$  indices of vertices that could have been deleted in the transition from  $W_{(c_0, \dots, c_m, 0, \dots, 0, c_m, \dots, c_0)[r]}((v_0, \dots, v_m, u_1, \dots, u_r, v_m'', \dots, v_0''))$  to  $W_{(c_0', \dots, c_{m'}')[r]}((v_0', \dots, v_{m'}'))$ . For fixed values of  $e$ ,  $e'$ , and  $I$ , there are at most  $n^{2m-e-e'-|I|}$  possible values of  $(v_0', \dots, v_{m'}')$ , each of which could be derived from at most  $n^{|I|}$  possible values of  $(v_0, \dots, v_m)$  and  $(v_0'', \dots, v_m'')$ . Also, the absolute value of the shard's expected value is at most

$$\begin{aligned} & n^{e+e'-2m+|I|} (k/(\min p_i))^{82^{8(s-1)}} \left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{(2m-|I|)/l} \\ & \cdot \left( \max_{s \leq j \leq h} (\lambda_j^2/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} (\lambda_j - \lambda'_i)^2/\Lambda \right)^{-(e+e')/l} \\ & \cdot \left( \max_{s \leq j \leq h} (|\lambda_j|/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i|/\Lambda \right)^{-\frac{8(l+2r+1)}{l}} \end{aligned}$$

which means that all such shards for given values of  $e$  and  $e'$  have a combined expected contribution of absolute value at most

$$\begin{aligned} & \Lambda^{(e+e'+2r)/r} 2^{(e+e'+2r)/r} (k/(\min p_i))^{82^{8(s-1)}} \left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{(2m-(e+e')/r-2)/l} \\ & \cdot \left( \max_{s \leq j \leq h} (\lambda_j^2/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} (\lambda_j - \lambda'_i)^2/\Lambda \right)^{-(e+e')/l} \\ & \cdot \left( \max_{s \leq j \leq h} (|\lambda_j|/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i|/\Lambda \right)^{-\frac{8(l+2r+1)}{l}} \end{aligned}$$

and thus all such shards for all  $e$  and  $e'$  have a combined expected contribution of absolute

value at most

$$\begin{aligned}
& 4\Lambda^2(k/(\min p_i))^8 2^{8(s-1)} \left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{(2m-2)/l} \\
& \cdot \left( \max_{s \leq j \leq h} (|\lambda_j|/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i|/\Lambda \right)^{-\frac{8(l+2r+1)}{l}} \\
& \cdot \left[ 1 - (4\Lambda)^{1/2r} \left( \max_{s \leq j \leq h} (\lambda_j^2/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} (\lambda_j - \lambda'_i)^2/\Lambda \right)^{-(2r+1)/2rl} \right]^{-2}
\end{aligned}$$

which is within a constant factor of

$$\left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{2m/l} = O \left( \left( \sum_{i=s}^h \prod_{j=0}^m (\lambda_i - c_j) \right)^2 \right).$$

2. Some but not all of the edges in the cycle in  $G''$  show up more than once in the walk defined by  $(v'_0, \dots, v'_{m'})$ :

The only way for this to happen is if the edges on one side of the cycle are repeated and the edges on the other side are not. The only way for this to happen is if  $(v'_0, \dots, v'_{m'})$  simply goes across the repeated branch of the cycle when it goes from  $v_0$  to  $v_m$  and then goes around the cycle one and a fraction times on its way back or vice-versa. So, each edge in the repeated branch shows up exactly 3 times in the walk defined by  $(v'_0, \dots, v'_{m'})$ . Let  $e''$  be the number of edges on the repeated side, and  $e'''$  be the number of edges on the non-repeated side.  $e''' = 2m - 2e - 2e' - 3e'' - |I| \leq 2(m - e - e' - e'') - |I|$ , and there are at most  $(2e + 2e' + 3e'' + 2r)/2r$  indices of vertices that could be in  $I$ . For fixed values of  $e$ ,  $e'$ ,  $e''$ , and  $I$ , there are at most  $n^{2m - e - e' - 2e'' - |I|}$  possible labelings of the vertices in  $G''$ , each of which corresponds to at most 2 possible values of  $(v'_0, \dots, v'_{m'})$ . Each of those could be derived from at most  $n^{|I|}$  possible values of  $(v_0, \dots, v_m)$  and  $(v''_0, \dots, v''_m)$ . Also, the absolute value of the shard's expected value is at most

$$\begin{aligned}
& n^{e+e'+2e''-2m+|I|} (k/(\min p_i))^8 2^{8(s-1)} \left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{(2m-|I|)/l} \\
& \cdot \left( \max_{s \leq j \leq h} (\lambda_j^2/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} (\lambda_j - \lambda'_i)^2/\Lambda \right)^{-(e+e'+2e'')/l} \\
& \cdot \left( \max_{s \leq j \leq h} (|\lambda_j|/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i|/\Lambda \right)^{-\frac{8(l+2r+1)}{l}}.
\end{aligned}$$

So, by the same logic as in the previous case, all such shards have a combined expected contribution of absolute value at most

$$\begin{aligned}
& 4\Lambda(k/(\min p_i))^8 2^{8(s-1)} \left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{(2m-1)/l} \\
& \cdot \left( \max_{s \leq j \leq h} (|\lambda_j|/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i|/\Lambda \right)^{-\frac{8(l+2r+1)}{l}} \\
& \cdot \left[ 1 - (4\Lambda)^{1/2r} \left( \max_{s \leq j \leq h} (\lambda_j^2/\Lambda)^{l-s+1} \prod_{i=1}^{s-1} (\lambda_j - \lambda'_i)^2/\Lambda \right)^{-(2r+1)/2rl} \right]^{-3}
\end{aligned}$$

which is within a constant factor of

$$\left( \max_{s \leq j \leq h} |\lambda_j|^{l-s+1} \prod_{i=1}^{s-1} |\lambda_j - \lambda'_i| \right)^{2m/l} = O \left( \left( \sum_{i=s}^h \prod_{j=0}^m (\lambda_i - c_j) \right)^2 \right).$$

3. Every edge in the cycle in  $G''$  appears more than once in the walk defined by  $(v'_0, \dots, v'_{m'})$ :

Let  $e''$  and  $e'''$  be the numbers of edges on the two branches of the cycle, and  $f$  be the number of times the walk defined by  $(v'_0, \dots, v'_{m'})$  makes around the cycle on its way from  $v_0$  to  $v_m$ , and  $f'$  be the number of times it goes around the cycle on its way back.  $G''$  has at most  $(2m - |I|)/2$  edges, so  $e' + e'' + e''' + e'''' \leq m - |I|/2$ . For fixed values of  $e, e', e'', e''', f, f'$ , and  $I$ , there are at most  $n^{e+e'+e''+e'''}$  options for  $G''$ . Each of these options corresponds to at most 4 possible values of  $(v'_0, \dots, v'_{m'})$ , and each of those corresponds to at most  $n^{|I|}$  possible values of  $(v_0, \dots, v_m)$  and  $(v''_0, \dots, v''_m)$ . Also,  $W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))$  is one if every edge in  $G''$  is also in  $G$  and 0 otherwise. That means that

$$|\mathbb{E}[W_{(c'_0, \dots, c'_{m'})[r]}((v'_0, \dots, v'_{m'}))]| = O((\lambda_1/n)^{e+e'+e''+e'''})$$

There are at most  $m/r$  nonzero elements of  $(c_0, \dots, c_m, 0 \dots 0, c_m, \dots, c_0)$ , and thus at most  $2^{m/r}$  possible values of  $I$ . So, all shards for given values of  $e, e', e'', e''', f, f'$  and  $I$  make a combined contribution with an absolute value of

$$O(\Lambda^{|I|} \lambda_1^{m-|I|/2}) \subseteq O(\Lambda^{m/r} \lambda_1^{m-m/2r}).$$

Since  $0 \leq e, e', e'', e''', f, f' \leq m$ , the combined expected contribution of all these shards is

$$O(m^6 2^{m/r} \Lambda^{m/r} \lambda_1^{m-m/2r}) \subseteq o \left( \left( \sum_{i=s}^h \prod_{j=0}^m (\lambda_i - c_j) \right)^2 \right).$$

□

**Lemma 19.** *There exists a constant  $r_0$  and  $m_0 = \Theta(\log n)$  such that if  $m, \Lambda, \lambda'_1, \dots, \lambda'_{s-1}, l, r$ , and  $(c_0, \dots, c_m)$  satisfy the conditions of the degree bound lemma,  $r > r_0$  and  $m > m_0$ , then for all  $v$ , we have that*

$$\text{Var}[W_{m/\{c_i\}}(x, v)] = O \left( \frac{1}{n} \prod (\lambda_s - c_i)^2 \right)$$

*Proof.* We have

$$W_{m/\{c_i\}}^2(x, v) = \sum_{\substack{v_0, \dots, v_m, v'_0, \dots, v'_m \in G: \\ v_m = v'_m = v}} x_{v_0} x_{v'_0} W_{(c_0, \dots, c_m)}((v_0, \dots, v_m)) \cdot W_{(c_0, \dots, c_m)}((v'_0, \dots, v'_m)).$$

If  $v_0 \neq v'_0$  then  $E[x_{v_0} \cdot x_{v'_0}] = 0$ , while if  $v_0 = v'_0$  then  $E[x_{v_0} \cdot x_{v'_0}] = 1$ . So,

$$\begin{aligned} E[W_{m/\{c_i\}}^2(x, v)] &= \sum_{\substack{v_0, \dots, v_m, v'_0, \dots, v'_m \in G: \\ v_0 = v'_0, v_m = v'_m = v}} E[W_{(c_0, \dots, c_m)}((v_0, \dots, v_m)) \cdot W_{(c_0, \dots, c_m)}((v'_0, \dots, v'_m))] \\ &= O \left( \frac{1}{n} \prod (\lambda_s - c_i)^2 \right) \end{aligned}$$

where the last equality follows by the previous lemma.

□

### 6.3 Degree profiling and exact recovery

Recall that  $p$  is a probability vector of dimension  $k$ ,  $Q$  is a  $k \times k$  symmetric matrix with positive entries, and let  $\mathbb{G}_2(n, p, Q)$  denote the stochastic block model with community prior  $p$  and connectivity matrix  $\ln(n)Q/n$ . A random graph  $G$  drawn under  $\mathbb{G}_2(n, p, Q)$  has a planted community assignment, which we denote by  $\sigma \in [k]^n$  and sometime call the true community assignment. Also, recall that exact recovery is solvable if there exists an algorithm that assigns each vertex in  $G$  to its community<sup>2</sup> with probability  $1 - o(1)$ .

---

<sup>2</sup>Up to a relabelling of the communities

### 6.3.1 Formal results

**Definition 26.** Let  $\mu, \nu$  be two positive measures on a discrete set  $\mathcal{X}$ , i.e., two functions from  $\mathcal{X}$  to  $\mathbb{R}_+$ . We define the CH-divergence between  $\mu$  and  $\nu$  by

$$D_+(\mu, \nu) := \max_{t \in [0,1]} \sum_{x \in \mathcal{X}} (t\mu(x) + (1-t)\nu(x) - \mu(x)^t \nu(x)^{1-t}). \quad (6.57)$$

Note that for a fixed  $t$ ,

$$\sum_{x \in \mathcal{X}} (t\mu(x) + (1-t)\nu(x) - \mu(x)^t \nu(x)^{1-t})$$

is an  $f$ -divergence. For  $t = 1/2$ , i.e., the gap between the arithmetic and geometric means, we have

$$\sum_{x \in \mathcal{X}} t\mu(x) + (1-t)\nu(x) - \mu(x)^t \nu(x)^{1-t} = \frac{1}{2} \|\sqrt{\mu} - \sqrt{\nu}\|_2^2 \quad (6.58)$$

which is the Hellinger divergence (or distance), and the maximization over  $t$  of the part  $\sum_x \mu(x)^t \nu(x)^{1-t}$  is the exponential of to the Chernoff divergence. We refer to Section 8.3 for further discussions on  $D_+$ . Note also that we will often evaluate  $D_+$  as  $D_+(x, y)$  where  $x, y$  are vectors instead of measures.

Next we present our main theorem for exact recovery. We first provide necessary and sufficient conditions for exact recovery, and then provide an algorithm that solves exact recovery efficiently, more precisely, in quasi-linear time.

**Theorem 11.** Let  $k \in \mathbb{Z}_+$  denote the number of communities,  $p \in (0, 1)^k$  with  $|p| = 1$  denote the community prior,  $P = \text{diag}(p)$ , and let  $Q \in (0, \infty)^{k \times k}$  symmetric with no two rows equal.

- Exact recovery is solvable in the stochastic block model  $\mathbb{G}_2(n, p, Q)$  if and only if for all  $i \neq j$  in different subsets of the partition,

$$D_+((PQ)_i, (PQ)_j) \geq 1, \quad (6.59)$$

where  $(PQ)_i$  denotes the  $i$ -th row of the matrix  $PQ$ .

- For  $G \sim \mathbb{G}_2(n, p, Q)$  on which exact recovery is solvable, the algorithm **Degree-profiling**( $G, p, Q, \gamma$ ) (see below<sup>3</sup>) recovers the communities with probability  $1 - o_n(1)$  and runs in  $o(n^{1+\epsilon})$  time for all  $\epsilon > 0$ .

---

<sup>3</sup> $\gamma = \gamma(n, p, Q)$  is set to  $\frac{\Delta-1}{2\Delta} + \frac{\ln \ln n}{4 \ln n}$ , where  $\Delta = \min_{i \neq j} D_+((pQ)_i, (pQ)_j)$

Note that this gives an operational meaning to a new divergence function, analog to the operational meaning given to the KL-divergence by the channel coding theorem.

**Remark 10.** *f*  $Q_{ij} = 0$  for some  $i$  and  $j$  then the results above still hold, except that if for all  $i \neq j$ ,

$$D_+((PQ)_i, (PQ)_j) \geq 1, \quad (6.60)$$

but there exist  $i \neq j$  such that  $D_+((PQ)_i, (PQ)_j) = 1$  and  $((PQ)_{i,k} \cdot (PQ)_{j,k} \cdot ((PQ)_{i,k} - (PQ)_{j,k}) = 0$  for all  $k$ , then the optimal algorithm will have an asymptotically constant failure rate. The recovery algorithm also needs to be modified to accomodate 0's in  $Q$ .

**Remark 11.** *As shown in the proof of Theorem 11, when exact recovery is not solvable, any algorithm must confuse at least one vertex with probability  $1 - o_n(1)$ , and not just with probability away from 0. Hence exact recovery has a sharp threshold at  $\min_{i,j \in [k], i \neq j} D_+((PQ)_i || (PQ)_j) = 1$ .*

**Example 2.** *For the symmetric block model where  $p$  is equiprobable on  $[k]$  and  $Q$  takes only two different values,  $\alpha$  on the diagonal and  $\beta$  outside the diagonal (with  $\alpha, \beta > 0$ ), the requirement in Theorem 11 for recovery of any (or all) communities is equivalent to*

$$|\sqrt{\alpha} - \sqrt{\beta}| \geq \sqrt{k}, \quad (6.61)$$

which generalizes the result obtained in [1, 19] for  $k = 2$ .

The algorithm **Degree-profiling** is given in Section 2.4 and replicated below. The idea is to recover the communities with a two-step procedure, similarly to one of the algorithms used in [1] for the two-community case. In the first step, we run **Sphere-comparison** on a sparsified version of  $\mathbb{G}_2(n, p, Q)$  which has a slowly growing average degree. Hence, from Corollary 1, **Sphere-comparison** recovers a fraction of nodes that is arbitrarily close to 1 correctly (w.h.p.). In the second step, we proceed to an improvement of the first step classification by making local checks for each node in the residue graph and deciding whether the node should be moved to another community or not. This step requires solving a hypothesis testing problem for deciding the local degree profile of vertices in the SBM. The CH-divergence appears when resolving this problem, as the mis-classification error exponent. We present this result of independent interest in Section 6.3.2. The proof of Theorem 11 is given in Section 6.3.3.

**Degree-profiling algorithm.**

Inputs: a graph  $g = ([n], E)$ , the SBM parameters  $p_i$ ,  $i \in [k]$ ,  $Q_{i,j}$ ,  $i, j \in [k]$ , and a splitting parameter  $\gamma \in [0, 1]$  (see Theorem 11 for the choice of  $\gamma$ ).

Output: Each node  $v \in [n]$  is assigned a community.

Algorithm:

- (1) Define the graph  $g'$  on the vertex set  $[n]$  by selecting each edge in  $g$  independently with probability  $\gamma$ , and define the graph  $g''$  that contains the edges in  $g$  that are not in  $g'$ .
- (2) Run **Sphere-comparison** on  $g'$  to obtain the preliminary classification  $\sigma' \in [k]^n$  (see Section 2.3 and Corollary 1.)
- (3) Determine the edge density between each pair of alleged communities, and use this information and the alleged communities' sizes to attempt to identify the communities up to symmetry.
- (4) For each node  $v \in [n]$ , determine in which community node  $v$  is most likely to belong to based on its degree profile computed from the preliminary classification  $\sigma'$  (see Section 6.3.2), and call it  $\sigma''_v$ .
- (5) For each node  $v \in [n]$ , determine in which community node  $v$  is most likely to belong to based on its degree profile computed from the preliminary classification  $\sigma''$  (see Section 6.3.2).

### 6.3.2 Testing degree profiles

In this section, we consider the problem of deciding which community a node in the SBM belongs to based on its degree profile. We first make the latter terminology precise.

**Definition 27.** *The degree profile of a node  $v \in [n]$  for a partition of the graph's vertices into  $k$  communities is the vector  $d(v) \in \mathbb{Z}_+^k$ , where the  $j$ -th component  $d_j(v)$  counts the number of edges between  $v$  and the vertices in community  $j$ .*

For  $G \sim \mathbb{G}_2(n, p, Q)$ , community  $i \in [k]$  has a relative size that concentrates exponentially fast to  $p_i$ . Hence, for a node  $v$  in community  $j$ ,  $d(v)$  is approximately given by  $\sum_{i \in [k]} X_{ij} e_i$ , where  $X_{ij}$  are independent and distributed as  $\text{Bin}(np_i, \ln(n)Q_{i,j}/n)$ , and where  $\text{Bin}(a, b)$  denotes<sup>4</sup> the binomial distribution with  $a$  trials and success probability  $b$ . Moreover, the Binomial is well-enough approximated by a Poisson distribution of the same mean in this regime. In particular, Le Cam's inequality gives

$$\left\| \text{Bin}\left(na, \frac{\ln(n)}{n}b\right) - \mathcal{P}(ab \ln(n)) \right\|_{TV} \leq 2 \frac{ab^2 \ln^2(n)}{n}, \quad (6.62)$$

---

<sup>4</sup> $\text{Bin}(a, b)$  refers to  $\text{Bin}(\lfloor a \rfloor, b)$  if  $a$  is not an integer.

hence, by the additivity of Poisson distribution and the triangular inequality,

$$\|\mu_{d(v)} - \mathcal{P}(\ln(n) \sum_{i \in [k]} p_i Q_{i,j} e_i)\|_{TV} = O\left(\frac{\ln^2(n)}{n}\right). \quad (6.63)$$

We will rely on a simple one-sided bound (see (6.94)) to approximate our events under the Poisson measure.

Consider now the following problem. Let  $G$  be drawn under the  $\mathbb{G}_2(n, p, Q)$  SBM and assume that the planted partition is revealed except for a given vertex. Based on the degree profile of that vertex, is it possible to classify the vertex correctly with high probability? We have to resolve a hypothesis testing problem, which involves multivariate Poisson distributions in view of the previous observations. We next study this problem.

**Testing multivariate Poisson distributions.** Consider the following Bayesian hypothesis testing problem with  $k$  hypotheses. The random variable  $H$  takes values in  $[k]$  with  $\mathbb{P}\{H = j\} = p_j$  (this is the a priori distribution of  $H$ ). Under  $H = j$ , an observed random variable  $D$  is drawn from a multivariate Poisson distribution with mean  $\lambda(j) \in \mathbb{R}_+^k$ , i.e.,

$$\mathbb{P}\{D = d | H = j\} = \mathcal{P}_{\lambda(j)}(d), \quad d \in \mathbb{Z}_+^k, \quad (6.64)$$

where

$$\mathcal{P}_{\lambda(j)}(d) = \prod_{i \in [k]} \mathcal{P}_{\lambda_i(j)}(d_i), \quad (6.65)$$

and

$$\mathcal{P}_{\lambda_i(j)}(d_i) = \frac{\lambda_i(j)^{d_i}}{d_i!} e^{-\lambda_i(j)}. \quad (6.66)$$

In other words,  $D$  has independent Poisson entries with different means. We use the following notation to summarize the above setting:

$$D | H = j \sim \mathcal{P}(\lambda(j)), \quad j \in [k]. \quad (6.67)$$

Our goal is to infer the value of  $H$  by observing a realization of  $D$ . To minimize the error probability given a realization of  $D$ , we must pick the most likely hypothesis conditioned on this realization,

i.e.,

$$\operatorname{argmax}_{j \in [k]} \mathbb{P}\{D = d | H = j\} p_j, \quad (6.68)$$

which is the Maximum A Posteriori (MAP) decoding rule.<sup>5</sup> To resolve this maximization, we can proceed to a tournament of  $k - 1$  pairwise comparisons of the hypotheses. Each comparison allows us to eliminate one candidate for the maxima, i.e.,

$$\mathbb{P}\{D = d | H = i\} p_i > \mathbb{P}\{D = d | H = j\} p_j \quad \Rightarrow \quad H \neq j. \quad (6.69)$$

The error probability  $P_e$  of this decoding rule is then given by,

$$P_e = \sum_{i \in [k]} \mathbb{P}\{D \in \text{Bad}(i) | H = i\} p_i, \quad (6.70)$$

where  $\text{Bad}(i)$  is the region in  $\mathbb{Z}_+^k$  where  $i$  is not maximizing (6.68). Moreover, for any  $i \in [k]$ ,

$$\mathbb{P}\{D \in \text{Bad}(i) | H = i\} \leq \sum_{j \neq i} \mathbb{P}\{D \in \text{Bad}_j(i) | H = i\} \quad (6.71)$$

where  $\text{Bad}_j(i)$  is the region in  $\mathbb{Z}_+^k$  where  $\mathbb{P}\{D = x | H = i\} p_i \leq \mathbb{P}\{D = x | H = j\} p_j$ . Note that with this upper-bound, we are counting the overlap regions where  $\mathbb{P}\{D = x | H = i\} p_i \leq \mathbb{P}\{D = x | H = j\} p_j$  for different  $j$ 's multiple times, but no more than  $k - 1$  times. Hence,

$$\sum_{j \neq i} \mathbb{P}\{D \in \text{Bad}_j(i) | H = i\} \leq (k - 1) \mathbb{P}\{D \in \text{Bad}(i) | H = i\}. \quad (6.72)$$

Putting (6.70) and (6.71) together, we have

$$P_e \leq \sum_{i \neq j} \mathbb{P}\{D \in \text{Bad}_j(i) | H = i\} p_i, \quad (6.73)$$

$$= \sum_{i < j} \sum_{d \in \mathbb{Z}_+^k} \min(\mathbb{P}\{D = d | H = i\} p_i, \mathbb{P}\{D = d | H = j\} p_j) \quad (6.74)$$

and from (6.72),

$$P_e \geq \frac{1}{k - 1} \sum_{i < j} \sum_{d \in \mathbb{Z}_+^k} \min(\mathbb{P}\{D = d | H = i\} p_i, \mathbb{P}\{D = d | H = j\} p_j). \quad (6.75)$$

---

<sup>5</sup>Ties can be broken arbitrarily.

Therefore the error probability  $P_e$  can be controlled by estimating the terms  $\sum_{d \in \mathbb{Z}_+^k} \min(\mathbb{P}\{D = d|H = i\}p_i, \mathbb{P}\{D = d|H = j\}p_j)$ . In our case, recall that

$$\mathbb{P}\{D = d|H = i\} = \mathcal{P}_{\lambda(i)}(d), \quad (6.76)$$

which is a multivariate Poisson distribution. In particular, we are interested in the regime where  $k$  is constant and  $\lambda(i) = \ln(n)c_i$ ,  $c_i \in \mathbb{R}_+^k$ , and  $n$  diverges. Due to (6.74), (6.75), we can then control the error probability by controlling  $\sum_{x \in \mathbb{Z}_+^k} \min(\mathcal{P}_{\ln(n)c_1}(x)p_1, \mathcal{P}_{\ln(n)c_2}(x)p_2)$ , which we will want to be  $o(1/n)$  to classify vertices in the SBM correctly with high probability based on their degree profiles (see next section). The following lemma provides the relevant estimates.

**Lemma 20.** *For any  $c_1, c_2 \in (\mathbb{R}_+ \setminus \{0\})^k$  with  $c_1 \neq c_2$  and  $p_1, p_2 \in \mathbb{R}_+ \setminus \{0\}$ ,*

$$\sum_{x \in \mathbb{Z}_+^k} \min(\mathcal{P}_{\ln(n)c_1}(x)p_1, \mathcal{P}_{\ln(n)c_2}(x)p_2) = O\left(n^{-D_+(c_1, c_2) - \frac{\ln \ln(n)}{2 \ln(n)}}\right), \quad (6.77)$$

$$\sum_{x \in \mathbb{Z}_+^k} \min(\mathcal{P}_{\ln(n)c_1}(x)p_1, \mathcal{P}_{\ln(n)c_2}(x)p_2) = \Omega\left(n^{-D_+(c_1, c_2) - \frac{k \ln \ln(n)}{2 \ln(n)}}\right), \quad (6.78)$$

where  $D_+(c_1, c_2)$  is the CH-divergence as defined in (6.57).

In other words, the CH-divergence provides the error exponent for deciding among multivariate Poisson distributions. We did not find this result in the literature, but found a similar result obtained by Verdú [21], who shows that the Hellinger distance (the special case with  $t = 1/2$  instead of the maximization over  $t$ ) appears in the error exponent for testing Poisson point-processes, although [21] does not investigate the exact error exponent.

*Proof of Lemma 20.* Assume without loss of generality that  $c_{1,1} \neq c_{2,1}$ . To prove the first half of the lemma, note that for any  $t \in [0, 1]$ ,

$$\sum_{x \in \mathbb{Z}_+^k} \min(\mathcal{P}_{\ln(n)c_1}(x)p_1, \mathcal{P}_{\ln(n)c_2}(x)p_2) \quad (6.79)$$

$$\leq \max(p_1, p_2) \sum_{x \in \mathbb{Z}_+^k} \min(\mathcal{P}_{\ln(n)c_1}(x), \mathcal{P}_{\ln(n)c_2}(x)) \quad (6.80)$$

$$= \max(p_1, p_2) \sum \min(e^{-\ln n \sum c_{1,i}} \prod (\ln n \cdot c_{1,i})^{x_i} / x_i!, e^{-\ln n \sum c_{2,i}} \prod (\ln n \cdot c_{2,i})^{x_i} / x_i!) \quad (6.81)$$

$$= \max(p_1, p_2) e^{-\ln n \sum t c_{1,i} + (1-t) c_{2,i}} \sum (\ln n \cdot c_{1,i}^t c_{2,i}^{1-t})^{x_i} / x_i! \quad (6.82)$$

$$\cdot \min\left(e^{-\ln n (1-t) \sum c_{1,i} - c_{2,i}} \prod (c_{1,i} / c_{2,i})^{(1-t)x_i}, e^{-\ln n t \sum c_{2,i} - c_{1,i}} \prod (c_{2,i} / c_{1,i})^{tx_i}\right) \quad (6.83)$$

For any choice of  $x_2, \dots, x_k$ , there must exist  $x_1$  (not necessarily an integer) such that  $e^{-\ln n(1-t) \sum c_{1,i} - c_{2,i}} \prod (c_{1,i}/c_{2,i})^{(1-t)x_i} = 1$ . As a result, the expression above must be less than or equal to

$$\max(p_1, p_2) e^{-\ln n \sum t c_{1,i} + (1-t) c_{2,i}} \sum (\ln n \cdot c_{1,i}^t c_{2,i}^{1-t})^{x_i} / x_i! \quad (6.84)$$

$$\cdot \min((c_{1,1}/c_{2,1})^{1-t}, (c_{2,1}/c_{1,1})^t)^{|x_1 - \ln n c_{1,1}^t c_{2,1}^{1-t}| - 1} \quad (6.85)$$

$$= O\left(e^{\ln n \sum (c_{1,i}^t c_{2,i}^{1-t} - t c_{1,i} - (1-t) c_{2,i})} / \sqrt{\ln n}\right) \quad (6.86)$$

When  $t$  is chosen to maximize  $\sum t c_{1,i} + (1-t) c_{2,i} - c_{1,i}^t c_{2,i}^{1-t}$ , this is

$$O\left(n^{-D_+(c_1, c_2) - \frac{\ln \ln(n)}{2 \ln(n)}}\right).$$

To prove the second half, let  $t$  maximize  $\sum t c_{1,i} + (1-t) c_{2,i} - c_{1,i}^t c_{2,i}^{1-t}$ . Hence

$$\sum c_{1,i} - c_{2,i} - \ln(c_{1,i}/c_{2,i}) c_{1,i}^t c_{2,i}^{1-t} = 0.$$

This implies that

$$e^{-\ln n \sum c_{1,i}} \prod (\ln n \cdot c_{1,i})^{\ln n c_{1,i}^t c_{2,i}^{1-t}} = e^{-\ln n \sum c_{2,i}} \prod (\ln n \cdot c_{2,i})^{\ln n c_{1,i}^t c_{2,i}^{1-t}}.$$

As a result,

$$\min(\mathcal{P}_{\ln(n)c_1}(c_{1,i}^t c_{2,i}^{1-t} \ln n) p_1, \mathcal{P}_{\ln(n)c_2}(c_{1,i}^t c_{2,i}^{1-t} \ln n) p_2) \quad (6.87)$$

$$\geq \min(p_1, p_2) e^{-\ln n \sum c_{1,i}} \prod (\ln n \cdot c_{1,i})^{c_{1,i}^t c_{2,i}^{1-t} \ln n} / (c_{1,i}^t c_{2,i}^{1-t} \ln n)! \quad (6.88)$$

$$= \min(p_1, p_2) e^{-\ln n \sum c_{2,i}} \prod (\ln n \cdot c_{2,i})^{c_{1,i}^t c_{2,i}^{1-t} \ln n} / (c_{1,i}^t c_{2,i}^{1-t} \ln n)! \quad (6.89)$$

$$= \min(p_1, p_2) e^{-\ln n \sum t c_{1,i} + (1-t) c_{2,i}} \prod (\ln n \cdot c_{1,i}^t c_{2,i}^{1-t})^{c_{1,i}^t c_{2,i}^{1-t} \ln n} / (c_{1,i}^t c_{2,i}^{1-t} \ln n)! \quad (6.90)$$

$$= \Omega(\min(p_1, p_2) e^{-\ln n \sum t c_{1,i} + (1-t) c_{2,i} - c_{1,i}^t c_{2,i}^{1-t}} / (\ln n)^{k/2}) \quad (6.91)$$

$$= \Omega\left(n^{-D_+(c_1, c_2) - \frac{k \ln \ln(n)}{2 \ln(n)}}\right). \quad (6.92)$$

Thus,

$$\sum_{x \in \mathbb{Z}_+^k} \min(\mathcal{P}_{\ln(n)c_1}(x)p_1, \mathcal{P}_{\ln(n)c_2}(x)p_2) = \Omega\left(n^{-D_+(c_1, c_2) - \frac{k \ln \ln(n)}{2 \ln(n)}}\right). \quad (6.93)$$

□

This lemma together with previous bounds on  $P_e$  imply that if  $D_+(c_i, c_j) > 1$  for all  $i \neq j$ , the true hypothesis is correctly recovered with probability  $o(1/n)$ . Also, for any  $p$  and  $p'$  such that  $|p_j - p'_j| < \ln n / \sqrt{n}$  for each  $j$ ,  $Q$ ,  $\gamma(n)$ , and  $i$ ,

$$\sum_{x \in \mathbb{Z}_+^k} \max\left(\text{Bin}_{(np', \frac{(1-\gamma(n)) \ln(n)}{n} Q_i)}(x) - 2\mathcal{P}_{P_{Q_i}(1-\gamma(n)) \ln(n)/n}(x), 0\right) = O(1/n^2). \quad (6.94)$$

So, the error rate for any algorithm that classifies vertices based on their degree profile in a graph drawn from a sparse SBM is at most  $O(1/n^2)$  more than twice what it would be if the probability distribution of degree profiles really was the poisson distribution.

In summary, we have proved the following.

**Lemma 21.** *Let  $k \in \mathbb{Z}_+$  and  $G$  be a random graph drawn under  $\mathbb{G}_2(n, p, (1 - \gamma(n))Q)$ . Assigning the most likely community to a node  $v$  based on its degree profile  $d(v)$  gives the correct assignment with probability*

$$1 - O\left(n^{-(1-\gamma(n))\Delta - \frac{1}{2} \ln((1-\gamma(n)) \ln n) / \ln n} + \frac{1}{n^2}\right),$$

where

$$\Delta = \min_{i \neq j} D_+((pQ)_i, (pQ)_j). \quad (6.95)$$

Moreover, we will need the following “robust” version of this lemma to prove Theorem 11.

**Lemma 22.** *Let  $k \in \mathbb{Z}_+$  and  $G$  be a random graph drawn under  $\mathbb{G}_2(n, p, (1 - \gamma(n))Q)$ . There exist  $c_1$ ,  $c_2$ , and  $c_3$  such that for any  $\delta$ , assigning the most likely community to a node  $v$  based on a distortion of its degree profile that independently gets each node’s community wrong with probability at most  $\delta$  gives the correct assignment with probability at least*

$$1 - c_2 \cdot (1 + c_1 \delta)^{c_3 \ln n} \cdot \left(n^{-(1-\gamma(n))\Delta - \frac{1}{2} \ln((1-\gamma(n)) \ln n) / \ln n}\right) - \frac{1}{n^2},$$

where

$$\Delta = \min_{i \neq j} D_+((pQ)_i, (pQ)_j). \quad (6.96)$$

*Proof.* Let

$$c_1 = \max_{i,j} \sum p_{i'} q_{i',j} / (p_i q_{i,j}).$$

The key observation is that  $v$ 's  $m$ th neighbor had at least a  $\min_{i,j} (p_i q_{i,j}) / \sum p_{i'} q_{i',j}$  chance of actually being in community  $\sigma$  for each  $\sigma$ , so its probability of being reported as being in community  $\sigma$  is at most  $1 + c_1 \delta$  times the probability that it actually is. So, the probability that its reported degree profile is bad is at most  $(1 + c_1 \delta)^{|N_1(v)|}$  times the probability that its actual degree profile is bad. Choose  $c_3$  such that each vertex in the graph has degree less than  $c_3 \ln n$  with probability  $1 - \frac{1}{n^2}$  and the conclusion follows from this and the previous bounds on the probability that classifying a vertex based on its degree profile fails.  $\square$

### 6.3.3 Proof of Theorem 11

We break the proof into two parts, the possibility and impossibility parts.

**Claim 1 (achievability).** Let  $G \sim \mathbb{G}_2(n, p, Q)$  and  $\gamma = \frac{\Delta-1}{2\Delta} + \frac{\ln \ln n}{4 \ln n}$ , where

$$\Delta = \min_{i \neq j} D_+((pQ)_i, (pQ)_j)$$

**Degree-profiling** $(G, p, Q, \gamma)$  recovers the communities with probability  $1 - o_n(1)$  if for all  $i \neq j$  in  $[k]$ ,

$$D_+((PQ)_i, (PQ)_j) \geq 1. \quad (6.97)$$

The idea behind Claim 1 is contained in Lemma 21. However, there are several technical steps that need to be handled:

1. The graphs  $G'$  and  $G''$  obtained in step 1 of the algorithm are correlated, since an edge cannot be both in  $G'$  and  $G''$ . However, this effect can be discarded since two independent versions would share edges with low enough probability.
2. The classification in step 2 using **Sphere-comparison** has a vanishing fraction of vertices which

are wrongly labeled. This requires using the robust version of Lemma 21, namely Lemma 22.

3. In the case where  $D_+((PQ)_i, (PQ)_j) = 1$  a more careful classification is needed as carried in steps 3 and 4 of the algorithm.

*Proof.* With probability  $1 - O(1/n)$ , no vertex in the graph has degree greater than  $c_3 \ln n$ . Assuming that this holds, no vertex's set of neighbors in  $G''$  is more than

$$(1 - \max_{i,j} q_{i,j} \ln n/n)^{-c_3 \ln n} \cdot (n/(n - c_3 \ln n))^{c_3 \ln n} = 1 + o(1)$$

times as likely to occur as it would be if  $G''$  were independent of  $G'$ . So, the fact that they are not has negligible impact on the algorithm's error rate. As  $n$  goes to infinity,  $\sigma'$ 's expected error rate goes to 0, so the expected error in the algorithm's estimates of the connectivity rates between its communities also goes to 0. Thus, the algorithm labels the communities correctly (up to equivalent relabeling) with probability  $1 - o(1)$ . Now, let

$$\delta = (e^{\frac{(1-\gamma)}{2c_3} \min_{i \neq j} D_+((PQ)_i, (PQ)_j)} - 1)/c_1.$$

By Lemma 22, if the classification in step 2 has an error rate of at most  $\delta$ , then the classification in step 3 has an error rate of

$$O(n^{-(1-\gamma) \min_{i \neq j} D_+((PQ)_i, (PQ)_j)/2} + 1/n^2),$$

observing that if  $\sigma'_v \neq \sigma_v$  the error rate of  $\sigma''_{v'}$  for  $v'$  adjacent to  $v$  is at worst multiplied by a constant. That in turn ensures that the final classification has an error rate of at most

$$O\left((1 + O(n^{-(1-\gamma) \min_{i \neq j} D_+((PQ)_i, (PQ)_j)/2} + 1/n^2))^{c_3 \ln n} \frac{1}{n} \ln n^{-1/4}\right) = O\left(\frac{1}{n} \ln n^{-1/4}\right).$$

□

**Claim 2 (converse).** Let  $G \sim \mathbb{G}_2(n, p, Q)$ . If there exist  $i \neq j$  such that

$$D_+((PQ)_i, (PQ)_j) < 1, \tag{6.98}$$

then every algorithm classifying the vertices of  $G$  must mis-classify at least one vertex with probability  $1 - o_n(1)$ .

*Proof.* With probability  $1-o(1)$ , every community of  $G$  has a size that is within a factor of  $1+\ln n/\sqrt{n}$  of its expected size. Assume that this holds. Let  $S$  be a random set of  $n/\ln^3(n)$  of  $G$ 's vertices. With probability  $1-o(1)$  the number of vertices in  $S$  in community  $\ell$  is within  $\sqrt{n}$  of  $p_\ell n/\log^3 n$  for each  $\ell$ , and a randomly selected vertex in  $S$  is adjacent to another vertex in  $S$  with probability  $o(1)$ . Next, choose  $i \neq j$  such that  $D_+((PQ)_i, (PQ)_j) < 1$ . Now, let

$$x_\ell = \lfloor (PQ)_{\ell,i}^t (PQ)_{\ell,j}^{1-t} \ln n \rfloor \quad (6.99)$$

for each  $\ell \in [k]$ , where  $t \in [0, 1]$  is chosen to maximize

$$\sum t(PQ)_{\ell,i} + (1-t)(PQ)_{\ell,j} - (PQ)_{\ell,i}^t (PQ)_{\ell,j}^{1-t}. \quad (6.100)$$

Roughly speaking, we want to show that every vertex in community  $i$  or  $j$  has a degree profile of  $x$  with probability

$$\Omega(n^{-D_+((PQ)_i, (PQ)_j)} / \ln^{k/2}(n)) \quad (6.101)$$

and that there is no way to determine which community the vertices with this degree profile are in.

More precisely, call a vertex in  $S$  ambiguous if it has exactly  $x_\ell$  neighbors in community  $\ell$  that are not in  $S$  for each  $\ell$ . The probability distribution of a vertex's numbers of neighbors in  $G \setminus S$  in each community is essentially a multivariable poisson distribution. So, by the assumption that  $D_+((PQ)_i, (PQ)_j) < 1$  and the argument in Lemma 20, there exists  $\epsilon > 0$  such that a vertex in  $S$  that is in either community  $i$  or community  $j$  is ambiguous with probability  $\Omega(n^{\epsilon-1})$ . Furthermore, for a fixed community assignment and choice of  $S$ , there is no dependence between whether or not any two vertices are ambiguous. Also, an ambiguous vertex is not adjacent to any other vertex in  $S$  with probability  $1-o(1)$ . So, with probability  $1-o(1)$  there are at least  $\ln n$  ambiguous vertices in community  $i$  that are not adjacent to any other vertices in  $S$  and  $\ln n$  ambiguous vertices in community  $j$  that are not adjacent to any other vertices in  $S$ . These vertices are indistinguishable, so no algorithm classifies them all correctly with probability greater than  $1/\binom{2\ln n}{\ln n}$ . Therefore, every algorithm must misclassify at least one vertex with probability  $1-o(1)$ .  $\square$

## Chapter 7

# Simulations and code

Our results on ABP strongly suggest that in a graph drawn from the sparse SBM with  $SNR > 1$ , one can recover communities with accuracy greater than  $1/k$  by classifying vertices based on their edges' entries in the dominant eigenvectors of the nonbacktracking walk matrix, and [10] proved that this is true when  $\lambda_2$  has a degree of 1. As such, we wrote the following program in matlab which assigns vertices to communities based on the entries of the eigenvectors corresponding to their edges.

```
%Attempts to divide a graph's vertices into clusters by using the dominant
%eigenvectors of the graph's nonbacktracking walk matrix.

%G is a matrix listing all edges in the graph.
%More precisely, G has 2 columns and a row for each edge listing the vertices it is incident to.
%e is the number of edges in the graph.
%n is the number of vertices in the graph.
%k is the number of clusters to divide the graph into.

%This function outputs a list of length n that lists a number from 0 to k
%for each vertex. 1 to k are the communities, while vertices that are
%outside the main component get an entry of 0 in the output.
function C=graphCluster(G,e,n,k)
%Computes the graph's nonbacktracking walk matrix.
I1=spalloc(2*e,n,2*e);
I2=spalloc(2*e,n,2*e);
```

```

for i=1:e
    I1(2*i-1,G(i,1))=1;
    I1(2*i,G(i,2))=1;
    I2(2*i-1,G(i,2))=1;
    I2(2*i,G(i,1))=1;
end
B=I2*I1';
for i=1:e
    B(2*i-1,2*i)=0;
    B(2*i,2*i-1)=0;
end
%Finds the top k eigenvectors of the graph, or as many as it can if that is
%less than k.
flag=1;
maxV=k+1;
while flag>0
    maxV=maxV-1;
    [V,D,flag]=eigs(B,maxV);
end
%Convert the eigenvectors from vectors over the edges to vectors over the
%vertices, and deletes the entries corresponding to vertices outside the
%main community.
d=max(max(D));
V2=I1'*V;
indices=zeros(n,1);
count=n;
V2copy=V2;
for i=0:(n-1)
    if V2(n-i,1)==0
        V2=[V2(1:n-i-1,:);V2(n-i+1:count,:)];
        indices=[indices(1:n-i-1,1);indices(n-i+1:count,1)];
        count=count-1;
    end
end

```

```

end
%Recalibrates the magnitudes of the vectors in an effort to make their
%magnitudes correspond to the usefulness of the information they provide.
%Also nullifies the first eigenvector (which is irrelevant under standard
%assumptions) and any eigenvector with a significantly complex eigenvalue
%(the useful eigenvectors should have real eigenvalues).
s0=V2'*V2;
s=zeros(maxV,maxV);
for i=1:maxV
    l=D(i,i);
    x=((d-l)^2+l*1)/(d*(1-l-d));
    s(i,i)=sqrt((1+(1/x))/s0(i,i));
    if D(i,i)==d
        s(i,i)=0;
    end
    if abs(imag(D(i,i)))>.01
        s(i,i)=0;
    end
end
end
V2=V2*s;
%Runs kmeans to assign the vertices in the main component to communities.
[C,C2,dis]=kmeans([real(V2),imag(V2)],k,'replicates',10);
%Adds the vertices outside the main component back into the output as being
%in an unknown community.
for i=1:n
    if V2copy(i,1)==0
        C=[C(1:i-1);0;C(i:count)];
        count=count+1;
    end
end
end
end

```

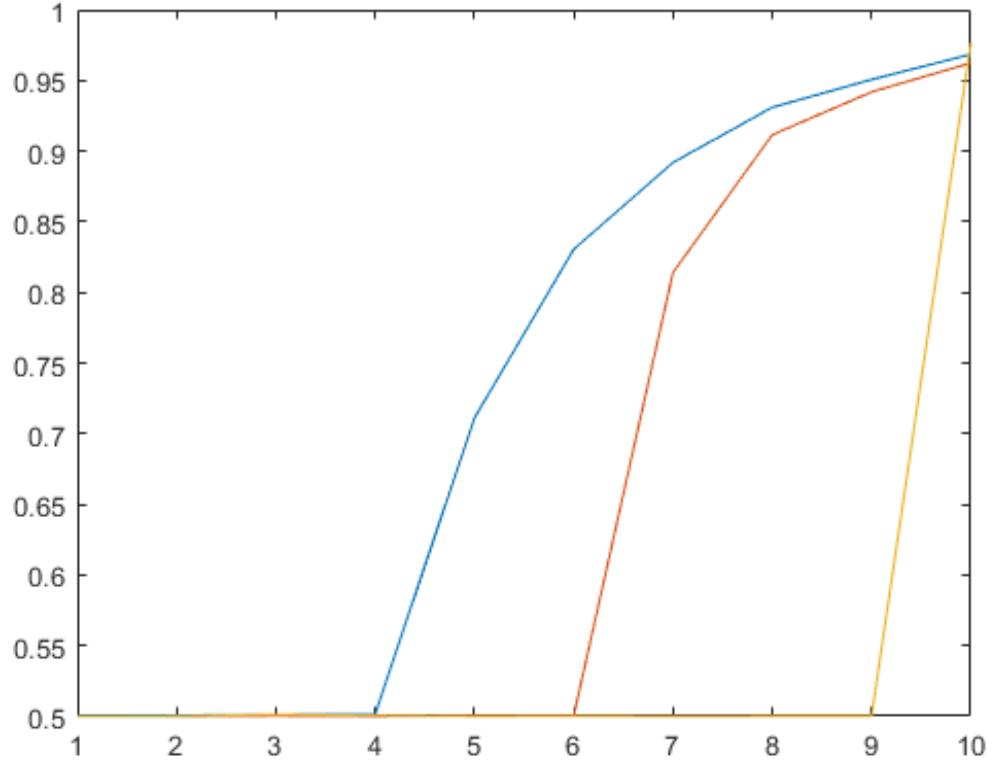


Figure 7.1: Accuracy of three algorithms at classifying vertices of graphs drawn from  $SSBM(20000, 2, a/20000, 1/20000)$  for different values of  $a$ . The blue line is for the algorithm given above, the red line is for an algorithm that classifies vertices based on the second eigenvector of the graph's adjacency matrix,  $A$ , and the yellow line is for an algorithm that classifies vertices based on the second eigenvector of  $D^{-1/2}AD^{-1/2}$ , where  $D$  is the diagonal matrix such that  $D_v = \text{degree}(v)$  for all  $v$ .

In order to test the effectiveness of this algorithm, we generated 10 graphs as follows. For each integer  $1 \leq a \leq 10$ , we generated a graph drawn from  $SSBM(20000, 2, a/2000, 1/20000)$  with the modification that we had exactly the expected number of vertices in each community and exactly the expected number of edges between each community. In other words, each graph had two communities of 10000 vertices each,  $2500a$  edges within each community, and 5000 edges between the communities. Then, we used three different algorithms to attempt to classify each graph's vertices. One of these algorithms was the one above. The second algorithm finds the second eigenvector of a graph's adjacency matrix, and then divides the vertices into those that have large and small entries in this vector. The third computes the second eigenvector of the matrix  $D^{-1/2}AD^{-1/2}$ , where  $A$  is

the graph's adjacency matrix and  $D$  is the diagonal matrix that lists the vertex's degrees. Then it divides the vertices into those that have large entries in this vector and those that have small entries in this vector.

Note that  $a = 5$  is the smallest integer for which  $SSBM(20000, 2, a/2000, b/20000)$  has a signal to noise ratio that is greater than 1. So, at least in the cases we have tried, our algorithm classifies vertices with an accuracy nontrivially greater than  $1/k$  whenever that is efficiently possible, and neither of the others do.

# Bibliography

- [1] E. Abbe, A. Bandeira, G. Hall. *Exact Recovery in the Stochastic Block Model*. IEEE Transactions on Information Theory 62, Number 1 (2016) pp. 471-487.
- [2] E. Abbe, C. Sandon. *Community detection in general stochastic block models: fundamental limits and efficient recovery algorithms*. FOCS 2015, pp. 670-688.
- [3] E. Abbe, C. Sandon. *Recovering communities in the general stochastic block model without knowing the parameters*. NIPS 28(2015) pp. 676-684.
- [4] E. Abbe, C. Sandon. *Achieving the KS Threshold in the General Stochastic Block Model with Linearized Acyclic Belief Propagation*. NIPS 29(2016) pp. 1334-1342.
- [5] E. Abbe, C. Sandon. *Crossing the KS Threshold in the Stochastic Block Model with Information Theory*. In the proceedings of ISIT 2016, pp. 840-844.
- [6] E. Abbe. *Community detection and stochastic block models: recent developments*. arXiv:1703.10146.
- [7] E. Abbe, C. Sandon. *Proof of the achievability conjectures in the general stochastic block model*. To appear in CPAM 2017.
- [8] S. Bhattacharyya, P. Bickel. *Community Detection in Networks using Graph Distance*. arXiv:1401.3915.
- [9] B. Bollobás, S. Janson, O. Riordan. *The Phase Transition in Inhomogeneous Random Graphs*. Random Structure Algorithms 31, Number 1 (2007), pp. 3-122.
- [10] C. Bordenave, M. Lelarge, L. Massoulié. *Non-backtracking spectrum of random graphs: community detection and non-regular Ramanujan graphs*. FOCS 2015, pp. 1347-1357.
- [11] A. Cayley. *Collected Mathematical Papers*. Cambridge University Press (1889-1897).

- [12] A. Deceelle, F. Krzakala, C. Moore, L. Zdeborova. *Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications*. Physical Review E 84, 066106, December 2011.
- [13] P. Erdős and A. Rényi, *On the Evolution of Random Graphs*. Publication of the Mathematical Institute of the Hungarian Academy of Sciences, 1960, pp. 17-61.
- [14] K. Hashimoto. *Zeta functions of finite graphs and representations of  $p$ -adic groups* Automorphic Forms and Geometry of Arithmetic Varieties 15 (1989), pp. 211-280.
- [15] L. Massoulié. *Community detection thresholds and the weak Ramanujan property*. 46th Annual Symposium on the Theory of Computing(2014) pp. 1-10.
- [16] E. Mossel, J. Neeman, A. Sly. *Stochastic Block Models and Reconstruction*. arXiv:1202.1499.
- [17] E. Mossel, J. Neeman, A. Sly. *Belief Propagation, Robust Reconstruction and Optimal Recovery of Block Models*. The Annals of Applied Probability Volume 26, Number 4 (2016).
- [18] E. Mossel, J. Neeman, A. Sly. *A Proof Of The Block Model Threshold Conjecture*. arXiv:1311.4115.
- [19] E. Mossel, J. Neeman, A. Sly. *Consistency Thresholds for the Planted Bisection Model*. arXiv:1407.1591.
- [20] A. Rényi. *Some Remarks on the Theory of Trees*. Publications of the Mathematical Institute of the Hungarian Academy of Sciences 4 (1959), pp. 73–85.
- [21] S. Verdú. *Asymptotic Error Probability of Binary Hypothesis Testing for Poisson Point-Process Observations*. IEEE Transactions on Information Theory, Volume 32, Number 1(1986), pp.113-115.