COMMUNITY DETECTION IN PREFERENTIAL ATTACHMENT GRAPHS

BY

SURYANARAYANA SANKAGIRI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Professor Bruce Hajek

ABSTRACT

This thesis examines the problem of community detection in a new random graph model, which is a generalization of preferential attachment graphs. This model has some features that are more realistic than those of the often-studied stochastic block model (SBM). A message passing algorithm for community detection is derived, and multiple simulation results are shown that demonstrate the efficacy of the algorithm. The algorithm is based on certain asymptotic properties unique to this model. These properties, some of which were discovered as part of this work, prove to be useful for other purposes as well, which are described in this thesis. In particular, a theoretical performance analysis is given for a simple, hypothesis-testing based community recovery algorithm. This thesis opens avenues to further theoretical analysis of this model, and takes a step toward developing community detection algorithms with strong theoretical foundations that work well on real-world networks.

To my teachers and parents, for their love and support.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Professor Bruce Hajek, for giving me a very challenging problem to work on, and thereafter spending many hours with me discussing the details of the problem. His insights are fascinating and inspiring, and he serves as a role model for me to do research. I would like to thank all my teachers at UIUC and at IIT Bombay for having taught so well that I could use these concepts in research, but perhaps more importantly, for showing me the beauty of electrical engineering. I also thank my past research advisors and mentors, for having faith in me that I could do research; their confidence in me boosts my own. My friends at UIUC have also been a vital component of my academic life here. I have learned much from them and trust I will learn a lot more. Finally, I would like to thank my parents, for giving me the freedom to choose my own path and also for their unwavering support.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1
1.1 Random graph models 2
1.2 Algorithms for community detection 5

CHAPTER 2 THE RANDOM GRAPH MODEL 7
2.1 Model description 7
2.2 Half-edges and their convergence 10
2.3 The degree growth process 12
2.4 The Z-process 15
2.5 Asymptotic degree distribution 16

CHAPTER 3 COMMUNITY DETECTION ALGORITHMS-BINARY HYPOTHESIS TESTING 19
3.1 Binary hypothesis testing 20
3.2 Hypothesis testing for Z 22

CHAPTER 4 MESSAGE PASSING 25
4.1 Overview 25
4.2 Algorithm 27
4.3 Derivation 29
4.4 Derivation discussion 34

CHAPTER 5 SIMULATIONS 37
5.1 Single community with outliers 37
5.2 Two equal-sized communities 40

CHAPTER 6 CONCLUSION 42

REFERENCES 43

CHAPTER 1

INTRODUCTION

Community detection, a.k.a. graph clustering, is the task of identifying clusters of vertices in a given graph such that vertices are more densely connected within a group than across groups. Posed in this manner, it is an unsupervised learning problem, much like clustering in a Euclidean space. The problem arises in many data mining applications, where the data is in the form of a network. The prototypical example of community detection is that of identifying groups of like-minded people in a social network (for example, distinguishing between people practicing science, arts or business). This information would be of use for sociological studies, and also for commercial exploitation by social media companies. In some cases, the underlying network may have many vertices that do not belong to any cluster. For example, in a citation network, the set of papers on community detection form a densely clustered subgroup among all papers with the phrase "community detection", where the outliers are papers that simply mention the term in passing. The popular review article by Fortunato [1] lists many applications, and also gives examples of popular data sets.

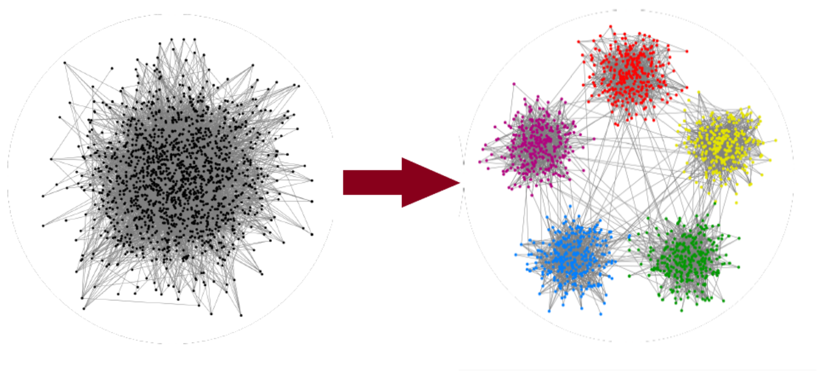

Figure 1.1: The task of community detection: separating groups of vertices into densely connected subgroups (figure from [2])

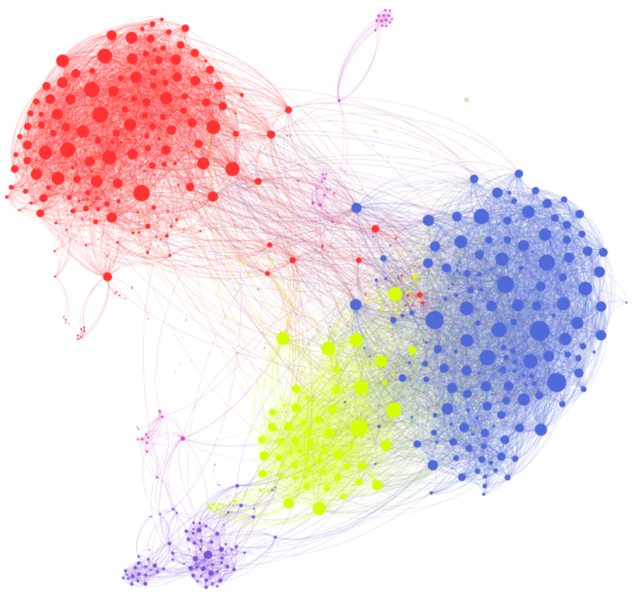

Figure 1.2: A social network formed from a single person's friends, with communities marked in colors; friends from school, friends from college, spouse's friends, etc. (figure from [3])

Because of its practical interest, a vast number of algorithms have been proposed for community detection. To analyze the performance of these algorithms, improve them, and eventually specify *optimal* algorithms requires mathematical modeling and probabilistic reasoning. Community detection is therefore of interest from a theoretical standpoint, and is a topic of research in both theoretical computer science as well as statistics. The recent review by Moore [4] explains the different viewpoints adopted by different fields. In this work, we adopt an approach that is more popular in the statistics community, which is to study random graph models with planted communities.

1.1 Random graph models

There are two advantages of adopting a random graph model. First, the model specifies the ground truth community structure, which allows us to evaluate the performance of any algorithm in terms of the number of vertices that were correctly classified. (In contrast, the min-bisection formulation does not lead to this natural error metric.) Such data is typically not available for practical data sets. Second, if one is able to show that an algorithm achieves a certain performance with high probability (w.h.p.), then that characterizes the algorithm's performance for a large number of graphs (the *typical* graphs of the model). This is another desirable property, that gives some performance guarantee on unseen graphs.

The described advantages also come with its set of disadvantages. An algorithm that is shown to be optimal for a certain random graph model may not work well for a graph generated by a different model. In practice, there is no concrete way to determine whether a given graph was generated according to a random graph model or not. There are two ways to tackle this problem: Develop algorithms that are *robust* to variations in the model, or study random graph models that are more realistic.

1.1.1 Stochastic block model

By far, the most popular (as well as the simplest) random graph model with planted communities is the stochastic block model (SBM). To generate a graph with two equal-sized communities, one begins with a set of n vertices, and randomly assigns a label $\ell \in \{1, 2\}$ to every vertex. Having assigned the labels, between any two pairs, an edge is placed with probability p if the two vertices have the same label, and probability q if the vertices have different labels. Usually, we choose $p > q$ to reflect the fact that entities (people) in the same community are more likely to be connected/linked than those in different communities. A description of the general SBM with any number of communities of possibly different sizes is given in [2]. Given the model, one can generate graphs that have an inherent community structure in them. *The task of community detection is to recover the labels from the graph.*

The SBM has been analyzed extensively in the literature. Many different algorithms have been proposed, many of which have theoretical guarantees of exactly recovering the communities with high probability for large graphs (e.g., [5], [6]). We review some of these algorithms in Section 1.2.

1.1.2 Preferential attachment graphs

The preferential attachment (PA) random graph model was proposed by Barabasi and Albert in [7] as a more realistic model of real-world networks than the Erdos-Renyi model. It describes a random graph that grows with time, as new vertices are added sequentially. At every stage, the new vertex chooses a fixed number of neighbors to attach to, within the existing graph (say m). A probability distribution over the vertices is constructed such that the probability of choosing any vertex is proportional to its degree. The neighbors are then chosen by sampling with replacement from this distribution m times.

A striking feature of this model is that the graphs have a heavy-tailed degree sequence (in the asymptotic limit). This feature is absent in the Erdos-Renyi model. Loosely speaking, this means that there exist a few vertices whose degree is much larger than the average degree. Such a feature is indeed exhibited in real-world networks such as the web, or in citation networks/social networks. This can be intuitively understood as a fallout of the rich-get-richer phenomenon in the model; vertices that have a high degree at a certain instance are more likely to attract newer vertices than others, and hence their degree is likely to grow even further. This property was first proven rigorously in [8], and is also explained in detail in the book by Van Der Hofstad [9].

1.1.3 Preferential attachment graphs with planted communities

Much like an Erdos-Renyi graph, the SBM does *not* have a heavy-tailed degree distribution. In order to develop community detection algorithms that are likely to succeed in real-world networks, it is necessary to study models that have more realistic features. With this aim, we study a model that incorporates community structure within PA graphs, in much the same way as the SBM incorporates communities within Erdos-Renyi graphs. Such a model was first proposed by Jordan in [10], in the context of geometric random graphs. The problem of community detection was not studied in this paper. This is the problem that we study in this thesis.

1.2 Algorithms for community detection

In this section, we briefly describe a few algorithms that have been proven to work well for community detection in the SBM.

1.2.1 Semi-definite programming

Given a graph generated according to a particular model, the task of finding the labels of the vertices (communities) is an inference problem. The edges (i.e., the graph) is the observed data, and the labels are the unknown parameters, with known prior distribution. In such a scenario, to minimize the probability of error, the best technique is to use the MAP estimator. For any problem, finding the MAP estimator is an optimization problem. For the case of two equal-sized communities, the MAP estimator is the same as finding the minimum bisection of the graph (to partition the vertices into two equal sets such that the number of edges across the sets is minimized). This is well known to be an NP-hard problem. However, it is possible to re-write the problem as a real-valued optimization problem, and then relax certain constraints so as to make it a semi-definite program. Semi-definite programs can be solved in polynomial time. Hajek et al. [5] show that the convex relaxation is tight, and gives exact recovery whenever it is statistically possible.

1.2.2 Spectral methods

The key idea behind spectral methods (for the SBM) is that the expected value of the adjacency matrix of the graph has a low-rank structure, and whose principal components give information about the communities. The expected adjacency matrix is thus the signal of interest. The *actual* adjacency matrix is treated as a perturbed version of its expected value. In other words, the true adjacency matrix is the sum of its expected value and a noisy matrix. Loosely speaking, if the noise levels are not too high, the eigenvectors of the perturbed matrix should be close to the eigenvectors of the clean signal, and should thus contain sufficient information about the communities. See [11] as an example of work done along this direction.

1.2.3 Belief propagation

Belief propagation is a general technique/algorithm to solve inference problems, especially when there are many latent random variables involved. The algorithm is best known in the context of probabilistic graphical models, where random variables are denoted by edges of a “factor graph”, and dependencies between them are represented via edges. The algorithm takes advantage of the conditional independence between the random variables, in order to compute the MAP estimator efficiently. The algorithm is guaranteed to converge to the best estimator if the underlying graph is a tree. In the context of community detection, the given graph itself captures the underlying dependencies in the following manner: the label of one vertex influences (and is influenced by) the number of neighbors it has and the labels of its neighbors, but is conditionally independent of any other vertex label given the labels of its neighbors. Belief propagation algorithms can also be theoretically analyzed, as shown in [12].

CHAPTER 2

THE RANDOM GRAPH MODEL

We begin this chapter by describing the generative random graph model that we study: preferential attachment graphs with planted communities. This model was first proposed by Jordan in [10]. We first describe the model in the most general setting, where it may include any finite number of communities, possible of unequal size. We then describe the parameters for two special cases that are most studied in the literature; that of two equal-sized communities and that of a single community with outliers. Having described the model, we present a fundamental property about the model that greatly simplifies subsequent analysis. This result is also from the paper by Jordan [10]. Finally, we describe and analyze the "degree-growth process", which serves as a tool for community detection as well as to describe the degree sequence of the graph.

2.1 Model description

The random graph model that we study describes the growth of a labeled, directed graph with time. Equivalently, the model describes a sequence of directed graphs $\{G_t : t \in \mathbb{N}\}$ and labels $\{\ell_t : t \in \mathbb{N}\}$. G_{t+1} is built by adding a new vertex and a few connecting edges to G_t, in much the same manner as in the original preferential attachment graph. The only difference is in the manner in which the neighbors of the new vertex are chosen. To describe the evolution of this process more precisely, we need the following parameters:

- $r \geqslant 1$: number of possible labels; labels are selected from $[r]$
- $\rho = (\rho_1, \ldots, \rho_r)$: probability vector for selection of labels of added vertices
- $m \geqslant 1$: out degree of each vertex

- $\beta \in \mathbb{R}^{r\times r}$: matrix of strictly positive affinities for vertices of different labels; β_{uv} is the affinity of a new vertex with label u to an existing vertex of label v
- $t_0 \geqslant 1$: initial time
- $G_{t_0} = (V_{t_0}, E_{t_0})$: initial directed graph with $V_{t_0} = [t_0]$ and mt_0 directed edges
- $T \geqslant t_0$: final time

Given the labeled graph G_t, the graph G_{t+1} is constructed as follows:

1. The vertex $t+1$ is added and its label ℓ_{t+1} is randomly selected from $[r]$ using distribution ρ, independently of G_t.
2. The m vertices in $V_t = [t]$ are selected using sampling with replacement according to a distribution f_t on $[t]$. The distribution f_t is proportional to the degree of the vertices *and is weighted based on the affinity matrix* β (this is the key difference from the standard preferential attachment model).
3. The m directed edges are added, with tails at vertex $t+1$ and with heads attached to the vertices chosen in step 2.

Figures 2.1-2.3 provide an illustration for these three steps.

In this model, vertices are referred to by their time of arrival. Let τ be some vertex in the graph, and consider a time point $t \geqslant \tau$. Let the degree of τ at time t be denoted by Y_t^τ. When vertex $t+1$ arrives, each edge from $t+1$ attaches to vertex τ with probability proportional to $Y_t^\tau \beta_{\ell_{t+1}\ell_\tau}$. Therefore, the number of edges from vertex $t+1$ to τ is a random variable with distribution:

$$\mathcal{L}\left(\#\text{ edges between } (t+1,\tau) | G_t, \ell_{[1,t+1]}\right) = \mathsf{Binom}\left(m, \frac{\beta_{\ell_{t+1}\ell_\tau} Y_t^\tau}{\sum_{\tau'} \beta_{\ell_{t+1}\ell_{\tau'}} Y_t^{\tau'}}\right) \tag{2.1}$$

$\mathcal{L}$ stands for probability law or probability distribution.

Remark 1. *For concreteness, one may assume that the initial graph has one vertex with m self-loops, and its label is chosen randomly with distribution ρ. However, all the asymptotic results are independent of the initial graph.*

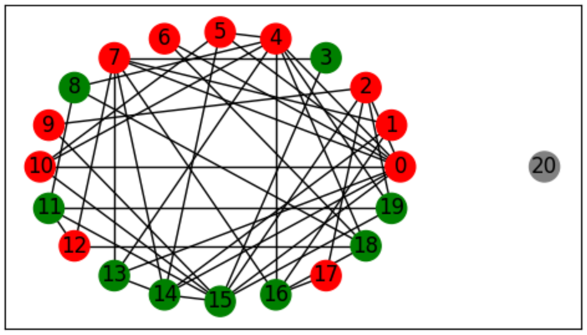

Figure 2.1: New vertex arrives

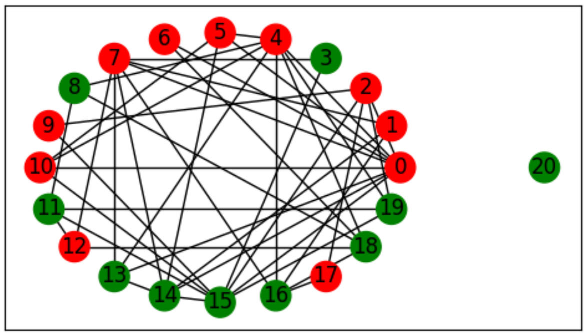

Figure 2.2: Label is randomly assigned to the new vertex

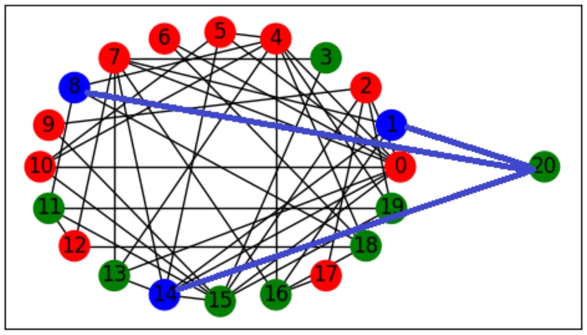

Figure 2.3: New vertex attaches to existing vertices using preferential attachment rule

A graph with two equal-sized communities can be created by choosing the parameters $r = 2$, $\rho = (0.5, 0.5)$, $\beta = \begin{bmatrix} b & 1 \\ 1 & b \end{bmatrix}$, where $b > 1$. The parameter m dictates the sparsity of the graph, and can be an arbitrary finite number (usually chosen to be much smaller than the size of the graph, T).

A graph with a single community with outliers can be created by choosing the parameters $r = 2$, $\beta = \begin{bmatrix} b & 1 \\ 1 & 1 \end{bmatrix}$, where $b > 1$. The parameter ρ dictates the size of the community. As above, the parameter m dictates the sparsity of the graph.

Having described the model, we move on to studying its characteristics. As mentioned earlier, the above model has features of both the PA model and the SBM. Thus, two properties that we expect in the model are:

1. a heavy-tailed degree sequence, like the PA model
2. existence of algorithms that (at least) partially recover the communities, as in the SBM

The heavy-tailed degree sequence property was proven in [10]; we only quote the result here and give empirical evidence in Section 2.5. This thesis focuses on the community detection aspect. We develop a message passing algorithm for community detection, and demonstrate empirically that it performs partial recovery. The algorithm and its derivation is given in Chapter 4, and the simulation results are given in Chapter 5. Before we proceed, however, we discuss a few basic results in the rest of this chapter that prove useful for further analysis.

2.2 Half-edges and their convergence

To analyze preferential attachment graphs, it is helpful to have a notion of a "half-edge" (see, e.g., [8]). In any graph, each edge is composed of two half-edges, one attached to each vertex the edge is incident on. Therefore, the degree of a vertex is the number of half-edges attached to it. In the PA model, the affinity to larger degree vertices can be viewed in terms of a tendency of an existing half-edge to attract newer half-edges. At every epoch, there are m new half-edges. Every incoming half-edge has an equal affinity toward existing half-edges. In other words, it chooses a partner half-edge *uniformly at random* (with probability $1/2mt$).

In preferential attachment graphs with labels, each half-edge inherits the label of the vertex it is incident upon. Given the labels, the affinity between half-edges is no longer uniform; they are weighted by a factor β_{uv}, where u, v are the labels of the new and existing half-edges respectively (hence β is called the affinity matrix). Moreover, the exact probability of attachment depends upon how many half-edges of each type there are at that point of time, which makes analysis difficult.

Suppose the labels of all vertices of G_t are known. Let the fraction of half-edges with label v in G_t be $\eta_{v,t}$. The probability that an incoming half-edge with label u attaches to an existing half-edge with label v is $\frac{\theta_{u,v,t}}{mt}$, where

$$\theta_{u,v,t} = \frac{\beta_{uv}}{2\sum_{v'} \beta_{uv'}\eta_{v',t}} \tag{2.2}$$

This follows from the following analysis:

$$\sum_{\tau' \in [t]} \beta_{u\ell'_\tau} Y_t^{\tau'} = \sum_{v' \in [r]} \beta_{uv'} \left(\sum_{\tau' \in [t]} Y_t^{\tau'} \mathbf{1}_{\ell_{\tau'} = v'} \right) \approx (2mt) \sum_{v' \in [r]} \beta_{uv'}\eta_{v'}$$

Although the above expression is complex, it simplifies significantly in the asymptotic limit, due to the following result, originally given in [10]:

Proposition 1 (convergence of fraction of half-edges)**.**

$$\eta_{v,t} \xrightarrow{a.s.} \eta_v^* \quad \forall\ v \in [r]$$

where $\eta^* \equiv \{\eta_v^*\}_{v \in [r]}$ *is a probability vector that is determined based on the parameters* ρ, β*. (The exact characterization of* η_v^* *is given in [13].)*

As an immediate corollary, we have that for large t, the probability that an incoming half-edge with label u attaches to an existing half-edge of label v is approximately $\frac{\theta_{u,v}}{mt}$, where

$$\theta_{u,v} \triangleq \frac{\beta_{u,v}}{2\sum_{v' \in [r]} \beta_{u,v'}\eta_v^*} \tag{2.3}$$

and is *independent* of the labels of the other vertices. This setting is now very similar to the PA model, with $\theta_{u,v}$ replacing 1/2. The parameters $\theta_{u,v}$ in this model play a similar role to the parameters $p_{u,v}$ in the SBM.

A rigorous proof of Proposition 1 is given by Jordan [10]. Here, we just give the outline of the proof. It is shown that the vector $\{\eta_{v,t}\}_{v\in[r]}$ varies with time according to a stochastic approximation scheme with decreasing step-sizes. The stochastic approximation scheme is viewed as a noisy discretization of an appropriate ordinary differential equation (ODE) [14]. It is shown in [10] that the underlying differential equation has a unique, globally asymptotically stable (GAS) equilibrium, by constructing an appropriate Lyapunov function. The convergence of the underlying ODE implies convergence of the stochastic approximation recursion [14].

2.3 The degree growth process

In this section, we study the degree growth process, *i.e.*, the growth of the degree of a fixed vertex in the graph with time. Due to the time-evolutionary nature of the model, the degree is best described as a *random process*, rather than a random variable. Obtaining the distribution (or probabilistic description) of the degree of a vertex allows us to establish both the properties mentioned above:

1. **Degree sequence:** Characterizing the degree sequence of graph G_T is equivalent to calculating the probability of a vertex picked uniformly at random among $[T]$ has degree greater than k, for each k. To get this figure, it is useful to know the probability of the degree of each vertex τ being greater than k. Studying the degree-growth process gives us this probability.

2. **Community detection:** If the time of arrival of a vertex is known, then the degree of a vertex is simply dependent upon its label. If the distributions of the degree for different labels are well separated, one can infer the labels by observing the degree. This is true for the model of a single community with outliers (but not the case of two equal-sized communities). Knowing the exact distribution of the degree allows us to formulate a hypothesis testing problem.

Having motivated the study of the degree-growth process, we now define it formally.

Definition 1 (The degree growth process)**.** *Given a vertex τ with $\tau \geqslant t_0 + 1$, consider the process $(Y_t : t \geqslant \tau)$, where Y_t is the degree of vertex τ at time t. Thus $Y_\tau = m$. The conditional distribution (i.e. probability law) of $Y_{t+1} - Y_t$ given $(Y_t, \eta_t, \ell_\tau = v, \ell_{t+1} = u)$ is given by:*

$$\mathcal{L}(Y_{t+1} - Y_t | Y_t, \eta_t, \ell_\tau = v, \ell_{t+1} = u) = \mathsf{binom}\left(m, \frac{\theta_{u,v,t} Y_t}{mt}\right) \tag{2.4}$$

where $\theta_{u,v,t}$ is given by (2.2). It follows that, given $(Y_t, \eta_t, \ell_\tau = v)$, the conditional distribution of $Y_{t+1} - Y_t$ is a mixture of binomial distributions with selection probability distribution ρ:

$$\mathcal{L}(Y_{t+1} - Y_t | Y_t, \eta_t, \ell_\tau = v) = \sum_{u \in [r]} \rho_u \mathsf{binom}\left(m, \frac{\theta_{u,v,t} Y_t}{mt}\right) \tag{2.5}$$

The above probability law essentially follows by rewriting (2.1) using the notion of half-edges introduced above:

$$\mathcal{L}\left(\#\text{ edges between } (t+1, \tau) | G_t, \ell_{[1,t+1]}\right) = \mathsf{binom}\left(m, \frac{\theta_{u,v,t} Y_t}{mt}\right) \tag{2.6}$$

From Proposition 1, we see that in the asymptotic limit, the parameters $\theta_{u,v,t}$ converge to $\theta_{u,v}$. Moreover, for large t, the quantity $\frac{\theta_{u,v}}{mt}$ is really small. A binomial random variable with a small p parameter takes values either 0 or 1 with high probability, and can thus be approximated by a Bernoulli random variable of the same mean. With these considerations, we see that:

$$\begin{aligned} \mathcal{L}(Y_{t+1} - Y_t | Y_t, \eta_t, \ell_\tau = v) &= \sum_{u \in [r]} \rho_u \mathsf{binom}\left(m, \frac{\theta_{u,v,t} Y_t}{mt}\right) && (2.7) \\ &\approx \sum_{u \in [r]} \rho_u \mathsf{binom}\left(m, \frac{\theta_{u,v} Y_t}{mt}\right) \\ &\approx \sum_{u \in [r]} \rho_u \mathsf{Ber}\left(\theta_{u,v} \frac{d_\tau(t)}{t}\right) \\ &= \mathsf{Ber}\left(\theta_v \frac{d_\tau(t)}{t}\right) && (2.8) \end{aligned}$$

where $\theta_v \triangleq \sum_{u \in [r]} \rho_u \theta_{u,v}$. We see that the degree-growth process has a simple probability law in the asymptotic regime.

In Proposition 3 of [13], the above approximation is made more precise. This is stated in terms of the total variation distance between the actual degree-growth process and an idealized version of it going to zero in the asymptotic limit. We first define a random process $\widetilde{Y}$ that is an idealized variation of Y obtained by replacing η_t by the constant vector η^* and allowing jumps of size one only.

Definition 2. *The process $\widetilde{Y}$ has parameters τ, m, and ϑ, where τ is the initial time, m is the initial state, and $\vartheta > 0$ is a rate parameter. The process $\widetilde{Y}$ is a time-inhomogeneous Markov process with initial value m. $\widetilde{Y}$ is defined as follows: for $t \geqslant \tau$ and y such that $\frac{\vartheta y}{t} \leqslant 1$, we require*

$$(\widetilde{Y}_{t+1} - \widetilde{Y}_t | \widetilde{Y}_t = y) \overset{d.}{=} \mathsf{Ber}\left(\frac{\vartheta y}{t}\right) \tag{2.9}$$

We have the following coupling result.

Proposition 2. *Suppose $\tau, T \to \infty$ such that $T > \tau$ and T/τ is bounded. Fix $v \in [r]$. Then*

$$d_{TV}((Y_{[\tau,T]} | \ell_\tau = v), \widetilde{Y}_{[\tau,T]}(\theta_v)) \to 0 \tag{2.10}$$

From Proposition 2, we infer that the degree growth process is essentially a time-inhomogeneous Markov process in the asymptotic limit. It is interesting to note that the degree growth process depends on m only via its initial condition. The state may either increase by 1, or stay the same at any point of time. The parameter θ_v gives us the rate of growth or the "jump rate" of this process. The larger this parameter, the larger is the rate at which the degree of vertex τ grows. This property is useful for community detection, as explained in Section 1.2.

Remark 2. *The line of arguments leading to* (2.8) *can also incorporate partial or complete information about other vertex labels. Suppose, from some other considerations, we know that the distribution of ℓ_{t+1} is $\widetilde{\rho}$. Then this distribution may be used in place of ρ in* (2.8). *We do not use this property here, but it is used in formulating the message passing algorithm later.*

2.4 The Z-process

The process $\widetilde{Y}$ can be thought of a discrete time birth process with activation time τ and birth probability at a time t proportional to the number of individuals. However, the birth probability (or birth rate) per individual, $\frac{\theta_{u,v}}{t}$, has a factor $\frac{1}{t}$, which tends to decrease the birth rate per individual. This makes the process difficult to analyze. By a simple time-scaling, we can obtain a process with constant birth rate. The scaling involves considering the process $(Y_{e^s} : s \geqslant 0)$. The intuition behind the scaling is the following: the graph of the function $\log t$ has slope $1/t$. By scaling the horizontal axis exponentially, we obtain the function $\log(\mathrm{e}^t) = t$, which has a constant slope 1.

Definition 3 (Z-process)**.** *The process $Z = (Z_s : s \geqslant 0)$ is a continuous time pure birth Markov process with initial state $Z_0 = m$ and birth rate ϑk in state k, for some $\vartheta > 0$. The process Z represents the total number of individuals in a continuous time branching process beginning with m individuals activated at time 0, such that each individual spawns another at rate ϑ.*

Definition 4. *Let $\check{Y}_t = Z_{\ln(t/\tau)}$ for integers $t \geqslant \tau$*

The following proposition, proven in [13], shows that $\widetilde{Y}$ and $\check{Y}$ are asymptotically equivalent in the sense of total variation distance.

Proposition 3. *For any $\vartheta > 0$,*

$$d_{TV}\left(\widetilde{Y}_{[\tau,T]}(\vartheta), \check{Y}_{[\tau,T]}(\vartheta)\right) \to 0 \tag{2.11}$$

Propositions 2 and 3 allow us to map many properties about the Z-process to the degree-growth process. We illustrate this by calculating the distribution of the degree of any vertex τ at time T.

We first establish the following useful fact about the Z-process.

Proposition 4. *For fixed s, Z_s has the negative binomial distribution $\mathsf{negbinom}(m, \mathrm{e}^{-s\vartheta})$. In other words, its marginal probability distribution $(\pi_n(s,\vartheta,m) : n \in \mathbb{N})$ is given by*

$$\pi_n(s,\vartheta,m) = \binom{n-1}{m-1}\mathrm{e}^{-m\vartheta s}(1-\mathrm{e}^{-\vartheta s})^{n-m} \quad \textit{for } n \geqslant m \tag{2.12}$$

In particular, taking $m = 1$ shows $\pi(s, \vartheta, 1)$ is the geometric distribution with parameter $\mathrm{e}^{-\vartheta s}$, and hence, mean $\mathrm{e}^{\vartheta s}$.

Proof. We prove the result only for $m = 1$. For $m \geqslant 2$, the process Z has the same distribution as the sum of m independent copies of Z with $m = 1$. The negative binomial distribution is also the sum of m independent geometric distributions, from which the result follows.

We shall prove that $\pi_i(s, \vartheta, 1) = e^{-\vartheta s}(1 - e^{-\vartheta s})^{i-1}$ by induction on i.

Induction base: From Definition 3, we see that the time stayed in state i is a random variable with distribution $\mathrm{Exp}(\lambda i)$. Therefore, $\pi_1(s, \vartheta, 1) = e^{-\vartheta s}$, since it is the probability that the system was in state 1 for time more than s.

Induction step: For $i > 1$, $\pi_i(s, \vartheta, 1)$ increases by the influx from state $i-1$ and decreases by the outflux from state i.

$$\frac{d\pi_i(s, \vartheta, 1)}{ds} = -i\vartheta\pi_i(s, \vartheta, 1) + (i-1)\vartheta\pi_{i-1}(s, \vartheta, 1) \; ; \quad \pi_i(0) = 0$$
$$\pi_i(s, \vartheta, 1) = \int_0^s \mathrm{e}^{-i\vartheta(s-\tau)}(i-1)\vartheta\pi_{i-1}(\tau, \vartheta, 1)d\tau$$

By the induction hypothesis, $\pi_{i-1}(s, \vartheta, 1) = \mathrm{e}^{-\vartheta s}(1 - \mathrm{e}^{-\vartheta s})^{i-2}$. Performing a change of variables $\tau \to \mathrm{e}^{\vartheta\tau} - 1$ in the integral above gives us the necessary expression for $\pi_i(t)$. □

We now have the following corollary from Proposition 4:

Corollary 1. *For $T > \tau >> 1$, $\mathcal{L}(Y_T^\tau | \ell_\tau = v) \approx \mathsf{NegBinom}\left(m, (T/\tau)^{\theta_v}\right)$*

.

2.5 Asymptotic degree distribution

A second limit result we restate from [10] concerns the empirical degree distribution for the vertices with a given label. For $v \in [r]$ and integers $n \geqslant m$ and T, let:

- $H^v(T)$ denote the number of vertices with label v in G_T
- $N_n^v(T)$ denote the number of vertices with label v and with degree n in G_T

- $P_n^v(T) = \frac{N_n^v(T)}{H^v(T)}$ denote the fraction of vertices with label v that have degree n in G_T

Proposition 5 (Limiting empirical degree seq. for a given label [10])**.** *Let $n \geqslant m$ and $v \in [r]$ be fixed. Then* $\lim_{T\to\infty} P_n^v(T) = p_n(\theta_v, m)$ *almost surely, where*

$$\begin{aligned} p_n(\theta, m) &= \frac{\Gamma\left(\frac{1}{\theta} + m\right)\Gamma(n)}{\theta\Gamma(m)\Gamma\left(n + \frac{1}{\theta} + 1\right)} \\ &\asymp \left[\frac{\Gamma\left(\frac{1}{\theta} + m\right)}{\theta\Gamma(m)}\right]\frac{1}{n^{\frac{1}{\theta}+1}} \end{aligned} \tag{2.13}$$

Proposition 5 is illustrated by plotting the degree distribution of a graph with the structure of a single community with outliers. To illustrate some interesting features, we plot the degree distribution of vertices in the community and outside separately, in Figures 2.4 and 2.5 respectively. The parameters chosen are $m = 1, b = 4, T = 10,000, \rho = (0.5, 0.5)$. In such a setting, the average degree of the graph is 2. As expected, we do see that there are vertices of both types with degrees much higher than the average degree. Further, we also see that vertices within the community have a higher degree on average than vertices outside the community. It is precisely this property that is exploited in the degree thresholding algorithm described in Chapter 3.

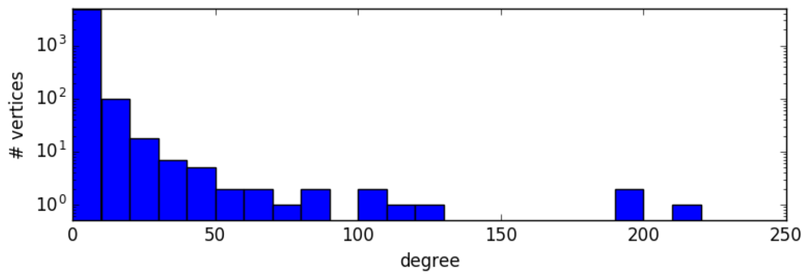

Figure 2.4: Degree distribution within community

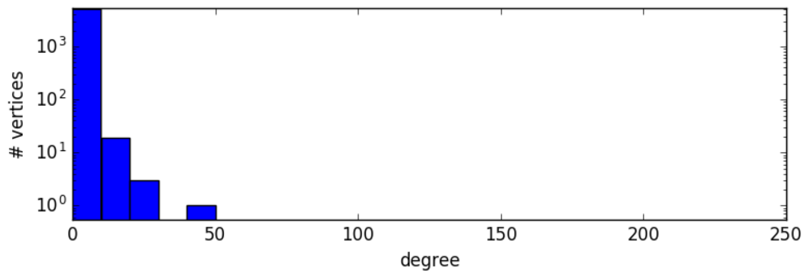

Figure 2.5: Degree distribution among outliers

CHAPTER 3

COMMUNITY DETECTION ALGORITHMS-BINARY HYPOTHESIS TESTING

In this chapter, we formulate the problem of community detection as a statistical inference problem. The labels on the vertices are the hidden variables, and the observed data is the graph, from which the labels must be inferred. Although, in principle, one can write down the likelihood of observing a given graph for every possible sequence of vertex labels, $\{\ell_t\}_{t=1}^{T}$ this expression is immensely complex. It is thus infeasible to optimize over the set of possible vertex labels and find the maximum aposteriori probability (MAP) estimator. We thus settle for other estimators that are easier to compute. These estimators are shown to be MAP estimators for certain simpler problems.

One simplification is to decouple the inference problem for different vertices. This simplifies the problem immensely, since instead of optimizing over r^T possible choices, one only has to optimize over r choices. A second simplifying technique is to perform the inference based only on observing a small neighborhood of the vertex, instead of the whole graph. This allows us to have a simple expression for the likelihood of the observation, conditioned on the label.

For concreteness, consider the model of a single community with outliers, where $\theta_1 > \theta_2$ (the label 1 indicates that the vertex is in the community). Consider any vertex τ s.t. $1 << \tau < T$ with $\ell_\tau = v$. The number of children of τ has distribution $\sim \mathsf{NegBinom}\left(m, (T/\tau)^{\theta_v}\right)$ as seen from Corollary 1. Thus, observing the number of children gives some information about the label v; the larger the number of children, the more likely it is that the vertex belongs to the community.

In fact, using the results from Chapter 2, we can compute the likelihood of observing the entire degree-growth process of any vertex τ. This allows us to calculate the MAP estimator of the vertex label based on additional information of which vertices attached to τ. This idea is presented in more detail in the rest of this chapter.

The algorithms presented in this chapter can be used, in principle, to perform community detection for any set of parameters ρ, β, as long as $\theta_v \neq \theta_{v'}$ for $v \neq v'$. However, for brevity, we focus only on the case of a single community with outliers. Note that these algorithms cannot be used to identify labels in a graph with equal-sized communities.

To comply with standard notations of binary hypothesis testing, we slightly change our notation:

- $\ell_\tau = 1$: Vertex is in the community
- $\ell_\tau = 0$: Vertex is an outlier

3.1 Binary hypothesis testing

Let $1 << \tau < T$ be integers. Consider the problem of estimating ℓ_τ given observation of a random object $\mathcal{O}$. For instance, the object could be the degree of vertex τ in G_T, or it could be the set of children of τ in G_T, or it could be the entire graph. We have two hypotheses about the label ℓ_τ:

$$\mathbf{H_1} : \ell_\tau = 1$$
$$\mathbf{H_0} : \ell_\tau = 0$$

The prior probabilities are $\pi_1 = \rho$ and $\pi_0 = 1 - \rho$ respectively. We seek to calculate the MAP estimator, which is a decision rule that minimize the average error probability, $p_e \triangleq \pi_1 p_{e,1} + \pi_0 p_{e,0}$, where $p_{e,v}$ is the conditional probability of error given H_v is true.

For binary hypothesis testing, the MAP estimator can be expressed in terms of thresholding either the log-belief ratio ξ or the log-likelihood ratio Λ:

$$\widehat{\tau}_{MAP} = \mathbf{1}_{\{\xi > 0\}} = \mathbf{1}_{\{\Lambda > \ln(\pi_0/\pi_1)\}}$$
$$\xi \triangleq \ln \frac{\mathbb{P}\{H_1|\mathcal{O}\}}{\mathbb{P}\{H_0|\mathcal{O}\}} \quad \Lambda \triangleq \ln \frac{p(\mathcal{O}|H_1)}{p(\mathcal{O}|H_0)}$$

Algorithm C The first recovery algorithm we describe, Algorithm C ("C" for "children"), is to let $\mathcal{O}$ denote the set of children, $\partial\tau = \{t_1, \ldots, t_n\}$, of vertex τ in G_T. This is equivalent to observing of $Y_{[\tau,T]}$. Given $\partial t =$

$\{t_1, \ldots, t_n\}$ with $\tau+1 \leqslant t_1 < \cdots < t_n \leqslant T$, let $y^{\partial\tau}_{[\tau,T]}$ denote the corresponding degree evolution sample path: $y^{\partial\tau}_t = m + |\partial\tau \cap [\tau, t]|$ for $\tau \leqslant t \leqslant T$.

Proposition 2 allows us to approximate the probability of observing a sample path of Y with the probability of observing the same sample path of $\widetilde{Y}$.

$$\begin{aligned} P(\partial\tau = \{t_1, \ldots, t_n\}) &= P(Y_{[\tau,T]} = y^{\partial\tau}_{[\tau,T]}) \\ &\approx P(\widetilde{Y}_{[\tau,T]} = y^{\partial\tau}_{[\tau,T]}) \\ &= \prod_{t\in[\tau+1,T]\setminus\partial\tau} \left(1 - \frac{y^{\partial\tau}_{t-1}\theta^*_v}{t-1}\right) \prod_{t\in\partial\tau} \frac{y^{\partial\tau}_{t-1}\theta^*_v}{t-1} \end{aligned}$$

so the log-likelihood ratio for observation $\widetilde{Y}_{[\tau,T]} = y^{\partial\tau}_{[\tau,T]}$ is

$$\Lambda^C = n \ln \frac{\theta^*_1}{\theta^*_0} + \sum_{t\in[\tau+1,T]\setminus\partial\tau} \ln\left(1 - \frac{y^{\partial\tau}_{t-1}\theta_1}{t-1}\right) - \ln\left(1 - \frac{y^{\partial\tau}_{t-1}\theta_0}{t-1}\right)$$

Algorithm C for estimating ℓ_v is to perform the likelihood ratio test using Λ^C. Using the approximation $\ln(1+s) = s$ and approximating the sum by an integral, we find

$$\begin{aligned} \Lambda^C &\approx n \ln \frac{\theta_1}{\theta_0} - (\theta_1 - \theta_0) \int_\tau^T \frac{y^{\partial\tau}_t}{t} dt \\ &= n \ln \frac{\theta_1}{\theta_0} - (\theta_1 - \theta_0) \left(m \ln(T/\tau) + \sum_{t\in\partial\tau} \ln(T/t) \right) \end{aligned} \tag{3.1}$$

Algorithm C is similar to performing hypothesis testing based on $\mathcal{O} = Z_{[0,\bar{s}]}$, where $\bar{s} = \ln(T/\tau)$. Let $f^C_Z(\rho, \theta_1, \theta_0, m, \bar{s})$ denote the resulting average error probability p_e. This quantity is estimated Section 3.2.

Algorithm DT The second recovery algorithm we describe, Algorithm DT ("DT" for "degree thresholding"), is to let $\mathcal{O}$ denote the number of children of vertex τ in G_T. Equivalently, $\mathcal{O}$ denotes the observation of Y_T.

Here too, using Proposition 3, we approximate the distribution of Y_T by that of $\check{Y}_T$, which has the $\mathsf{negbinom}\left(m, (\tau/T)^{\theta_v}\right)$ distribution under H_v for $v \in \{0, 1\}$. The log-likelihood ratio in this case, given the number of children is n, is

$$\Lambda^{DT} = -m(\theta_1 - \theta_0)\ln(T/\tau) + (n-m)\ln\left(\frac{1-(\tau/T)^{\theta_1}}{1-(\tau/T)^{\theta_0}}\right)$$

Algorithm DT for estimating ℓ_v is to perform the likelihood ratio test using Λ^{DT}, or in other words, the MAP decision rule based on $\mathcal{O} = Y_T$. This is similar to performing hypothesis testing based on $\mathcal{O} = Z_{\bar{s}}$, where $\bar{s} = \ln(T/\tau)$ (because $\check{Y}_T = Z_{\bar{s}}$). Let $f_Z^{DT}(\rho, \theta_1, \theta_0, m, \bar{s})$ denote the resulting average error probability p_e. It is clear that we expect $f_Z^{DT} \geqslant f_Z^C$, as Algorithm C is based on more information.

3.2 Hypothesis testing for Z

We consider here the binary hypothesis testing problem based on observation of $Z_{[0,\ln(T/\tau)]}$ such that under H_v it has rate parameter $\vartheta = \theta_v$ for $v \in \{0,1\}$. To this end, we derive the likelihood ratio.

Suppose $\{s_1, \ldots, s_n\} \subset (0, \bar{s}]$ such that $0 < s_1 < \cdots < s_n$ and $\bar{s} = \ln T/\tau$. Since the inter-jump periods are independent (exponential) random variables, the likelihood of $s_1, \ldots, s_n$ being the jump times during $[0, \bar{s}]$ under hypothesis H_v, is the product of the likelihoods of the observed inter-jump periods, with an additional factor of the likelihood of not seeing a jump in the last interval:

$$\left(\prod_{i=0}^{n-1} \theta_v(m+i)e^{-\theta_v(m+i)(s_{i+1}-s_i)}\right) e^{-\theta_v(n+m)(\bar{s}-s_n)}$$

Thus, the log-likelihood ratio for observing this is (letting $s_0 = 0$):

$$\Lambda^Z = n\ln\frac{\theta_1}{\theta_0} - (\theta_1 - \theta_0)\left(m\bar{s} + \sum_{i=1}^{n}(\bar{s} - s_i)\right) \tag{3.2}$$

(With $s_i = \ln(t_i/\tau)$, (3.2) is the same as (3.1), although in (3.1) the variables t_i are supposed to be integer valued.) Let $A_{\bar{s}} \triangleq \left(m\bar{s} + \sum_{i=1}^{n}(\bar{s} - s_i)\right)$. Note that $A_{\bar{s}}$ is the area under the trajectory of $Z_{[0,\bar{s}]}$. Moreover, $n+m$ is the value of $Z_{\bar{s}}$. So the log-likelihood ratio is given by

$$\Lambda^Z = (Z_{\bar{s}} - m)\ln\frac{\theta_1}{\theta_0} - (A_{\bar{s}})(\theta_1 - \theta_0) \tag{3.3}$$

which is a linear combination of $Z_{\bar{s}} - m$ and $A_{\bar{s}}$. Thus, the MAP decision rule has a simple form. Let $f_Z^C(\rho, \theta_1, \theta_0, m, \bar{s})$ denote the average error probability p_e for the MAP decision rule based on observation of $Z_{[0,\bar{s}]}$.

There is apparently no closed-form expression for the distribution of Λ^Z, so computation of $f_Z^C(\rho, \theta_1, \theta_0, m, \bar{s})$ requires Monte Carlo simulation or some other numerical method. A closed-form expression for the moment generating function of Λ^Z is given in the following proposition, which can be used to bound the probability of error. The proof of Proposition 6 is given in [13].

Proposition 6. *The joint moment generating function of Z_s and A_s is given as follows, where $\mathbb{E}_{\lambda,m}\left[\cdot\right]$ denotes expectation assuming the parameters of Z are λ, m :*

$$\begin{aligned}\psi_{\lambda,m}(u, v, s) &\triangleq \mathbb{E}_{\lambda,m}\left[e^{uZ_s + vA_s}\right] \\ &= \left(\frac{e^{(v-\lambda)s+u}}{1 + \frac{\lambda e^u}{v-\lambda}\left(1 - e^{(v-\lambda)s}\right)}\right)^m \end{aligned} \tag{3.4}$$

Proposition 6 can be used to bound p_e as follows. By a standard result in the theory of binary hypothesis testing, the probability of error for the MAP decision rule is bounded by

$$\pi_1\pi_0\rho_B^2 \leqslant p_e \leqslant \sqrt{\pi_1\pi_0}\rho_B \tag{3.5}$$

where the Bhattacharyya coefficient (or Hellinger integral) ρ_B is defined by $\rho_B = \mathbb{E}\left[\mathrm{e}^{\Lambda/2}\middle|H_0\right]$, and π_1 and π_0 are the prior probabilities on the hypotheses. The proposition with $\lambda = \theta_0$, $u = \frac{1}{2}\ln(\theta_1/\theta_0)$, $v = -\frac{\theta_1-\theta_0}{2}$, and $s = \bar{s}$ yields

$$\begin{aligned}\rho_{B,C} &= \mathbb{E}_{\lambda,m}\left[e^{u(Z_s-m)+vA_s}\right] = \psi_{\lambda,m}(u, v, s)e^{-mu} \\ &= \left(\frac{\mathrm{e}^{-(\theta_1+\theta_0)\bar{s}/2}}{1 - \frac{2\sqrt{\theta_1\theta_0}}{\theta_1+\theta_0}\left(1 - \mathrm{e}^{-(\theta_1+\theta_0)\bar{s}/2}\right)}\right)^m\end{aligned}$$

Here we wrote $\rho_{B,C}$ to denote it as the Bhattacharyya coefficient for Algorithm C (for the large T limit). Using this expression in (3.5) provides upper and lower bounds on $p_e = f_Z^C(\rho, \theta_1, \theta_0, m, \bar{s})$.

For the sake of comparison, we note that the Bhattacharyya coefficient for the hypothesis testing problem based on $\check{Y}_T$ alone, i.e. Algorithm DT, is easily found to be

$$\rho_{B,DT} = \left(\frac{\mathrm{e}^{-(\theta_1+\theta_0)\bar{s}/2}}{1-\sqrt{(1-\mathrm{e}^{-\theta_1\bar{s}})(1-\mathrm{e}^{-\theta_0\bar{s}})}} \right)^m$$

CHAPTER 4

MESSAGE PASSING

In this chapter, we describe how Algorithm C (the MAP rule given children) can be extended to a message passing algorithm. The message passing algorithm performs better than Algorithm C in recovering the labels of a graph with a single community with outliers, as each vertex takes into account more information than just its set of children. Further, message passing allows us to recover planted communities in graphs with equal-sized communities, which is impossible via Algorithms C and DT.

4.1 Overview

In Algorithm C, it was assumed that there was no prior information about the labels of any of the vertices. Here, information about a vertex label means a distribution of the vertex label that is not the prior distribution (this includes knowing the vertex label exactly). Message passing extends Algorithm C to take into account information about the children's labels. In addition, it takes into account any information about the parents' labels. Intuitively, having more information about one's neighbors' labels allows one to compute a better estimate of one's own label, as they are correlated.

In Chapter 3, we showed that Algorithm C gives some information about a vertex's label. If this information is passed on to the vertex's parent (as a message), the parent can form a better estimate of its own label. Messages from one vertex to another are posterior distributions of the sender's vertex label (or equivalent information). In effect, the parent vertex observes the graph up to a depth of two. Upon iterating, each vertex sees a larger portion of the graph at every step, ultimately observing the whole graph. In the rest of this chapter, we make the above notions precise. We discuss what the messages mean, how they are computed and sent from one vertex to the other, and make clear why indeed the extra information is useful for better inference.

We describe the algorithm for the case of two possible labels for a general 2×2 matrix β with positive entries, and fixed $m \geqslant 1$. The algorithm can be extended for the case of more than two communities, as shown in [13]. For binary valued random variables, distributions can be inferred from log-likelihood ratios, which is a compact representation (a scalar instead of a two-dimensional vector). Thus, all messages are represented here as log-likelihood ratios.

In general, a message passing algorithm on any graph is an iterative algorithm where vertices in the graph send "messages" to their neighbors based on messages that they receive in the previous iteration. Specific to the application at hand, messages are ascribed a certain meaning, and are computed accordingly. Message passing algorithms are generally used to solve inference problems in a factor graph (or more generally, a probabilistic graphical model) (see, e.g., [15]). A factor graph is a graphical representation of the joint distribution of multiple random variables, that takes into account the dependence (and conditional independence) conditions among the random variables. The advantage of this representation is that the marginal distribution (or marginal conditional distribution) of any random variable can be computed efficiently via a particular form of message passing called belief propagation. In belief propagation, the messages in every iteration can be thought of as a particular distribution.

For community detection, the message passing algorithm is similar to belief propagation, except for the fact that one does not construct a factor graph. The algorithm is run on the given graph itself, as the dependence/independence conditions are well represented by the given graph. In the rest of this section, we state and then derive a principled message passing algorithm for inferring the label of each vertex, given the graph. Throughout the remainder of this section, let (V, E) be a fixed instance of the random graph, (V_T, E_T), with two communities and known parameters $m, \beta, \rho, t_o, G_{t_o}$, and T. The message passing algorithm is run on this graph, with the aim of calculating Λ_τ for $1 \leqslant \tau \leqslant T$, where for each τ, Λ_τ is a log-likelihood ratio:

$$\Lambda_\tau \triangleq \ln \frac{\mathbb{P}\left\{E_T = E | \ell_\tau = 1\right\}}{\mathbb{P}\left\{E_T = E | \ell_\tau = 0\right\}} \tag{4.1}$$

Then we can calculate the maximum likelihood (ML) and maximum a posteriori probability (MAP) estimators of the label of a vertex τ as prescribed in Chapter 3. This estimator gives us the smallest probability of error in inferring each vertex.

Remark 3. *Note that this does not give us an estimator that gives us the smallest error of getting the entire set of vertices correct. See [2] for a detailed discussion on the best estimators for exact recovery versus partial recovery.*

4.2 Algorithm

A complete specification of a message passing algorithm includes specification of the following elements:

1. initial messages
2. mappings from messages received at a vertex to messages sent by the vertex
3. timing of message passing and termination criterion
4. mappings from messages received at a vertex to the output log-likelihood vector of the vertex

In this section, we specify the equations for elements (1), (2), and (4). A discussion on element (3) is given in the chapter on simulations (Chapter 5). To specify these equations, we need the following notation. Given vertices τ and τ_0, we say τ is a child of τ_0, and τ_0 is a parent of τ, if $\tau \geqslant \max\{\tau_0, t_o\} + 1$, and there is an edge from τ to τ_0. It is assumed that the known initial graph G_{t_o} is arbitrary, i.e., they are not placed according to particular model. Thus, for the inference problem at hand, the edges in G_{t_o} are not relevant beyond the degrees that they imply for the vertices in G_{t_o}.

Let $\partial\tau$ denote the children of τ in G_T and $\wp\tau$ the parents of τ. So $\wp\tau = \varnothing$ if $\tau \leqslant t_o$ and $\partial\tau \subset \{t_o + 1, \dots, T\}$. Let $\nu_{\tau\to\tau_0}$ denote a message passed from child to parent, and $\mu_{\tau_0\to\tau}$ denote a message passed from parent to child.

The functions $g^{cp} : \mathbb{R} \mapsto \mathbb{R}$ and $g^{pc} : \mathbb{R} \mapsto \mathbb{R}$ are defined as follows (here "cp" denotes child to parent, and "pc" denotes parent to child):

$$g^{cp}(\nu) = \ln\left(\frac{e^{\nu}\rho\,\theta_{1,1} + (1-\rho)\theta_{0,1}}{e^{\nu}\rho\,\theta_{1,0} + (1-\rho)\theta_{0,0}}\right) - \ln\frac{\theta_1}{\theta_0} \quad \text{for } \nu \in \mathbb{R}$$
$$g^{pc}(\mu) = \ln\left(\frac{e^{\mu}\rho\,\theta_{1,1} + (1-\rho)\theta_{1,0}}{e^{\mu}\rho\,\theta_{0,1} + (1-\rho)\theta_{0,0}}\right) \quad \text{for } \mu \in \mathbb{R}$$

where $\theta_{u,v}$ and θ_v are defined in Section 2.2.

For convenience, we repeat the expression in (3.1) for the log-likelihood ratio based on observation of children, for any vertex τ:

$$\lambda_\tau^C = |\partial\tau| \ln\frac{\theta_1}{\theta_0} + (\theta_1 - \theta_0)\left(d_0(\tau)\ln\frac{\tau \vee t_o}{T} + \sum_{t\in\partial\tau}\ln\frac{t}{T}\right) \tag{4.2}$$

where $\tau \vee t_o = \max\{\tau, t_o\}$ and $d_0(\tau)$ is the initial degree of vertex τ, defined to be the degree of τ in G_{t_o} if $\tau \leqslant t_o$ and $d_0(\tau) = m$ otherwise. This slight modification from (3.1) takes into account the fact that an initial vertex's degree growth process starts effectively at time t_o, and such a vertex can have arbitrary initial degree, not necessarily m.

The message passing equations are given as follows:

$$\nu_{\tau\to\tau_0} = \lambda_\tau^C + \sum_{t\in\partial\tau} g^{cp}(\nu_{t\to\tau}) + \sum_{\tau_1\in\wp\tau\backslash\{\tau_0\}} g^{pc}(\mu_{\tau_1\to\tau}) \tag{4.3}$$
$$\mu_{\tau_0\to\tau} = \lambda_{\tau_0}^C + \sum_{t\in\partial\tau_0\backslash\{\tau\}} g^{cp}(\nu_{t\to\tau}) + \sum_{\tau_1\in\wp\tau_0} g^{pc}(\mu_{\tau_1\to\tau}) - \ln\frac{\theta_1}{\theta_0} \tag{4.4}$$
$$\Lambda_\tau = \lambda_\tau^C + \sum_{t\in\partial\tau} g^{cp}(\nu_{t\to\tau}) + \sum_{\tau_0\in\wp\tau} g^{pc}(\mu_{\tau_0\to\tau}) \tag{4.5}$$

with the initial conditions:

$$\nu_{\tau\to\tau_0} = \lambda_\tau^C \quad \mu_{\tau_0\to\tau} = \lambda_{\tau_0}^C - \ln\frac{\theta_1}{\theta_0} \tag{4.6}$$

The message passing equations are written as if there are no parallel edges in (V, E). While the fraction of edges that are parallel to other edges will be small for large T, they are permitted. The convention used in the message passing algorithm is that $\partial\tau$ and $\wp\tau$ are to be considered as multisets, so that if a vertex appears with some multiplicity in one of those sets, then the corresponding term in the summations will be appearing the corresponding number of times. The initial conditions given by (4.15) are chosen to mimic the scenario of Algorithm C, where no messages are received.

Essentially, each vertex assimilates all the incoming messages to form an estimate of its own log-likelihood ratio. It then passes this quantity to all of its neighbors (with a small modification), so that they may use this information in their own likelihood calculations. The modification is to not include information gained from one neighbor in the message being sent to that neighbor. This form of the messages appears naturally in deriving the messages assuming the graph is a tree, and is a useful heuristic even in loopy message passing.

4.3 Derivation

Instead of deriving the message passing algorithms as stated above, we shall state and derive them in a slightly different form. The advantage of this alternative form is that the derivation becomes much more intuitive. The only price we pay is that of additional notation. Following the derivation, we show how the alternate form of the messages is equivalent to the original form presented in equations (4.3) - (4.5).

In what follows, message passing equations are stated and derived for the special case $m = 1$, with the initial graph G_{t_o} consisting of a single vertex (i.e. $t_o = 1$) with a self-loop. In that case, the graph (V, E) is a tree (ignoring any self-loops among the initial vertices) so the message passing algorithm is conceptually simpler. Equations (4.3) - (4.5) for any finite $m \geqslant 1$ are simply taken to have the same form as for $m = 1$ on the grounds that loopy message passing is obtained by using the same equations as for message passing without loops.

A second major difference from the equations given in Section 4.2 is that here, the messages are vectors in $\mathbb{R}^2$, rather than scalar quantities. This is because the earlier set of messages were interpreted as log-likelihood ratios, while here, the messages are simply log-likelihoods.

4.3.1 Alternate form

Let the notation of $\partial\tau, \wp\tau$ be the same as before. We abuse notation by redefining certain quantities differently than before as follows:

$$g^{cp}(\nu)(v) = \ln\left(\sum_{u\in[r]} \mathrm{e}^{\nu(u)}\rho_u \frac{\theta_{u,v}}{\theta_v}\right) \text{ for } \nu \in \mathbb{R}^r \tag{4.7}$$

$$g^{pc}(\mu)(u) = \ln\left(\sum_{v\in[r]} \mathrm{e}^{\mu(v)}\rho_v \frac{\theta_{u,v}}{\theta_v}\right) \text{ for } \mu \in \mathbb{R}^r \tag{4.8}$$

$$\lambda^C_\tau(v) = |\partial\tau| \ln\theta_v + \theta_v \left(d_0(\tau)\ln\frac{\tau \vee t_o}{T} + \sum_{t\in\partial\tau} \ln\frac{t}{T}\right) \tag{4.9}$$

The new message passing iterative equations are given by

$$\nu_{\tau\to\tau_0} = \lambda^C_\tau + \sum_{t\in\partial\tau} \widetilde{\nu}_{t\to\tau} \tag{4.10}$$

$$\mu_{\tau_0\to\tau} = \lambda^C_{\tau_0} + \sum_{t\in\partial\tau_0\setminus\{\tau\}} \widetilde{\nu}_{t\to\tau_0} + \widetilde{\mu}_{\tau_1\to\tau_0} \tag{4.11}$$

$$\widetilde{\nu}_{\tau\to\tau_0} = g^{cp}(\nu_{\tau\to\tau_0}) \tag{4.12}$$

$$\widetilde{\mu}_{\tau_0\to\tau} = g^{pc}(\mu_{\tau_0\to\tau}) \tag{4.13}$$

$$\Lambda_\tau = \lambda^C_\tau + \sum_{t\in\partial\tau} \widetilde{\nu}_{t\to\tau} + \widetilde{\mu}_{\tau_0\to\tau} \tag{4.14}$$

with the initial conditions:

$$\widetilde{\nu}_{\tau\to\tau_0} = 0 \quad \widetilde{\mu}_{\tau_0\to\tau} = 0 \tag{4.15}$$

or equivalently

$$\nu_{\tau\to\tau_0} = \lambda^C_\tau \;\; \mu_{\tau_0\to\tau} = \lambda^C_{\tau_0} \tag{4.16}$$

Note that the above are vector equations; they are to be interpreted as performing the same operation component-wise.

The initial conditions are chosen such that the initial value of $\Lambda_\tau = \lambda^C_\tau$. That is, the first step of message passing is equivalent to performing Algorithm C.

4.3.2 Derivation of alternate form

Conceptually, we view the entire graph as the collection of children sets of different vertices. As before, $\partial\tau$ denotes the children set of τ. $\partial\tau$ is a random quantity, while $\partial\tau = \{t_1, \ldots, t_n\}$ is an event. Let D_τ denote the set of all descendants of τ in the given graph. This includes the observation that $\partial\tau = \{t_1, \ldots, t_n\}$, $\partial t_1 = \{t_{1,1}, \ldots, t_{1,n_1}\}$, and so on. The entire graph is thus represented by D_1.

To derive the above recursion, we first define the quantities $\nu, \mu, \widetilde{\nu}, \widetilde{\mu}, \Lambda$ to be appropriate log-likelihoods. Equations (4.10) - (4.14) then follow from the relation among these log-likelihood quantities.

The aim of message passing is to calculate the log-likelihood ratio of each vertex's label upon observing the whole graph. Thus, we define Λ_τ as

$$\Lambda_\tau(v) \triangleq \ln \mathbb{P}\left\{D_1 | \ell_\tau = v\right\} \tag{4.17}$$

Using the following identity:

$$\begin{aligned}\mathbb{P}\left\{D_1 | \ell_\tau = v\right\} &= \mathbb{P}\left\{D_1 \backslash D_\tau | \ell_\tau = v\right\} \mathbb{P}\left\{D_\tau | \ell_\tau = v\right\} \\ &= \mathbb{P}\left\{D_1 \backslash D_\tau | \ell_\tau = v\right\} \mathbb{P}\left\{\partial\tau | \ell_\tau = v\right\} \prod_{t \in \partial\tau} \mathbb{P}\left\{D_t | \ell_\tau = v, \partial\tau\right\}\end{aligned}$$

and the definitions:

$$\widetilde{\mu}_{\tau_0 \to \tau}(v) \triangleq \ln \mathbb{P}\left\{D_1 \backslash D_\tau | \ell_\tau = v\right\} \tag{4.18}$$

$$\widetilde{\nu}_{t \to \tau}(v) \triangleq \ln \mathbb{P}\left\{D_t | \ell_\tau = v, \partial\tau\right\} \tag{4.19}$$

we get

$$\Lambda_\tau = \lambda_\tau^C + \sum_{t \in \partial\tau} \widetilde{\nu}_{t \to \tau} + \widetilde{\mu}_{\tau_0 \to \tau}$$

which is (4.14).

An expression for $\widetilde{\nu}_{t\to\tau}$ is obtained as follows:

$$\begin{aligned}
\mathrm{e}^{\widetilde{\nu}_{t\to\tau}(v)} &= \mathbb{P}\left\{D_t|\ell_\tau = v, \partial\tau\right\} \\
&= \sum_{u\in[r]} \mathbb{P}\left\{D_t|\ell_\tau = v, \ell_t = u, \partial\tau\right\} \mathbb{P}\left\{\ell_t = u|\partial\tau, \ell_\tau = v\right\} \\
&= \sum_{u\in[r]} \mathbb{P}\left\{D_t|\ell_t = u\right\} \frac{\mathbb{P}\left\{\partial\tau|\ell_\tau = v, \ell_t = u\right\}\mathbb{P}\left\{\ell_t = u|\ell_\tau = v\right\}}{\mathbb{P}\left\{\partial\tau|\ell_\tau = v\right\}} \\
&= \sum_{u\in[r]} \mathbb{P}\left\{D_t|\ell_t = u\right\} \frac{\theta_{u,v}}{\theta_v}\rho_u \qquad (4.20)
\end{aligned}$$

Define $\nu_{t\to\tau}$ as

$$\nu_{t\to\tau}(u) \triangleq \ln \mathbb{P}\left\{D_t|\ell_t = u\right\} \qquad (4.21)$$

Plugging this definition into (4.20), we obtain (4.12):

$$\widetilde{\nu}_{t\to\tau}(v) = \ln\left(\sum_{u\in[r]} \mathrm{e}^{\nu_{t\to\tau}(u)} \frac{\theta_{u,v}}{\theta_v}\rho_u\right)$$

From the definition of $\nu_{t\to\tau}$, and the following identity:

$$\mathbb{P}\left\{D_\tau|\ell_\tau = v\right\} = \mathbb{P}\left\{\partial\tau|\ell_\tau = v\right\} \prod_{t\in\partial\tau} \mathbb{P}\left\{D_t|\ell_\tau = v, \partial\tau\right\}$$

we get (4.10):

$$\nu_{t\to\tau} = \lambda_\tau^C + \sum_{t\in\partial\tau} \widetilde{\nu}_{t\to\tau}$$

Next, we derive an expression for $\widetilde{\mu}_{\tau_0\to\tau}$, i.e., (4.13).

We know that

$$\begin{aligned}
\mathbb{P}\left\{D_1 \backslash D_\tau | \ell_\tau = v\right\} &= \sum_{v'} \mathbb{P}\left\{D_1 \backslash D_\tau | \ell_\tau = v, \ell_{\tau_0} = v'\right\} \mathbb{P}\left\{\ell_{\tau_0} = v'\right\} \\
&= \sum_{v'} \frac{\mathbb{P}\left\{D_1 \backslash D_\tau | \ell_\tau = v, \ell_{\tau_0} = v'\right\} \theta_{v'}}{\theta_{v,v'}} \ \frac{\theta_{v,v'}}{\theta_{v'}} \rho_{v'} \\
&= \sum_{v'} \mathrm{e}^{\mu_{\tau_0 \to \tau}(v')} \ \frac{\theta_{v,v'}}{\theta_{v'}} \rho_{v'} \\
\Rightarrow \widetilde{\mu}_{\tau_0 \to \tau}(v) &= \ln \left(\sum_{v'} \mathrm{e}^{\mu_{\tau_0 \to \tau}(v')} \ \rho_{v'} \frac{\theta_{v,v'}}{\theta_{v'}} \right) \qquad (4.22)
\end{aligned}$$

In deriving (4.22), we have used the following definition of $\mu_{\tau_0 \to \tau}$:

$$\mu_{\tau_0 \to \tau}(v') \triangleq \ln \frac{\mathbb{P}\left\{D_1 \backslash D_\tau | \ell_\tau = v, \ell_{\tau_0} = v'\right\} \theta_{v'}}{\theta_{v,v'}} \qquad (4.23)$$

For this definition to be precise, we need to show that $\frac{\mathbb{P}\left\{D_1 \backslash D_\tau | \ell_\tau = v, \ell_{\tau_0} = v'\right\} \theta_{v'}}{\theta_{v,v'}}$ is independent of v. This is shown below. In parallel, we derive an expression for $\mu_{\tau_0 \to \tau}$.

$$\begin{aligned}
&\mathbb{P}\left\{D_1 \backslash D_\tau | \ell_\tau = v, \ell_{\tau_0} = v'\right\} \\
&= \mathbb{P}\left\{D_1 \backslash D_{\tau_0} | \ell_\tau = v, \ell_{\tau_0} = v'\right\} \mathbb{P}\left\{D_{\tau_0} \backslash D_\tau | \ell_\tau = v, \ell_{\tau_0} = v'\right\} \\
&= \mathbb{P}\left\{D_1 \backslash D_{\tau_0} | \ell_\tau = v, \ell_{\tau_0} = v'\right\} \mathbb{P}\left\{\partial \tau_0 | \ell_\tau = v, \ell_{\tau_0} = v'\right\} \\
&\times \prod_{t' \in \partial \tau_0 \backslash \{\tau_0\}} \mathbb{P}\left\{D_{t'} | \ell_{\tau_0} = v', \partial \tau_0\right\} \qquad (4.24)
\end{aligned}$$

We know:

$$\begin{aligned}
\mathbb{P}\left\{D_1 \backslash D_{\tau_0} | \ell_\tau = v, \ell_{\tau_0} = v'\right\} &= \mathbb{P}\left\{D_1 \backslash D_{\tau_0} | \ell_{\tau_0} = v'\right\} = \mathrm{e}^{\widetilde{\mu}_{\tau_1 \to \tau}(v')} \\
\frac{\mathbb{P}\left\{\partial \tau_0 | \ell_\tau = v, \ell_{\tau_0} = v'\right\} \theta_{v'}}{\theta_{v,v'}} &= \mathbb{P}\left\{\partial \tau_0 | \ell_{\tau_0} = v'\right\} = \mathrm{e}^{\lambda_C^{\tau_0}} \\
\mathbb{P}\left\{D_{t'} | \ell_{\tau_0} = v', \partial \tau_0\right\} &= \mathrm{e}^{\widetilde{\nu}_{t' \to \tau_0}(v')}
\end{aligned}$$

Combining these into (4.24), we get:

$$\mathbb{P}\left\{D_1 \backslash D_{\tau_0} | \ell_\tau = v, \ell_{\tau_0} = v'\right\} \frac{\theta_{v'}}{\theta_{v,v'}} = \mathrm{e}^{\widetilde{\mu}_{\tau_1 \to \tau}(v')} \mathrm{e}^{\lambda_C^{\tau_0}} \prod_{t' \in \partial \tau_0 \backslash \{\tau_0\}} \mathrm{e}^{\widetilde{\nu}_{t' \to \tau_0}(v')} \qquad (4.25)$$

Taking the logarithm of (4.25), we get

$$\mu_{\tau_0\to\tau} = \lambda_C^{\tau_0} + \widetilde{\mu}_{\tau_1\to\tau} + \sum_{t'\in\partial\tau_0\setminus\{\tau_0\}} \widetilde{\nu}_{t'\to\tau_0}(v')$$

which is (4.11). This completes the derivation of (4.10) - (4.14).

4.3.3 Deriving the original message passing equations

To get the scalar message passing Equations (4.3) - (4.5), we subtract the second component of the vector equations from the first component. This operation indicates that we are working with log-likelihood ratios, instead of log-likelihoods. In particular, the following quantities are now interpreted as:

$$\Lambda_\tau = \ln \frac{\mathbb{P}\{D_1|\ell_\tau = 1\}}{\mathbb{P}\{D_1|\ell_\tau = 0\}} \tag{4.26}$$

$$\nu_{\tau\to\tau_0} = \ln \frac{\mathbb{P}\{D_\tau|\ell_\tau = 1\}}{\mathbb{P}\{D_\tau|\ell_\tau = 0\}} \tag{4.27}$$

$$\mu_{\tau_0\to\tau} = \ln \frac{\mathbb{P}\{D_1\setminus D_\tau|\ell_\tau = 0, \ell_{\tau_0} = 1\}}{\mathbb{P}\{D_1\setminus D_\tau|\ell_\tau = 0, \ell_{\tau_0} = 0\}}\frac{\theta_1}{\theta_0}\frac{\theta_{0,0}}{\theta_{0,1}} \tag{4.28}$$

The quantities $\widetilde{\mu}, \widetilde{\nu}$ are done away with, by substituting (4.12) and (4.13) in (4.10) and (4.11).

Lastly, we sightly modify the definition of μ:

$$\mu_{\tau_0\to\tau} = \ln \frac{\mathbb{P}\{D_1\setminus D_\tau|\ell_\tau = 0, \ell_{\tau_0} = 1\}}{\mathbb{P}\{D_1\setminus D_\tau|\ell_\tau = 0, \ell_{\tau_0} = 0\}}\frac{\theta_{0,0}}{\theta_{0,1}} \tag{4.29}$$

This modification is to let μ be as close as possible to a log-likelihood ratio, without depending upon the label of τ (one could condition on $\ell_\tau = 1$ as well). This modification leads to a slight modification of g^{pc}, and the adding of the term $\ln \frac{\theta_1}{\theta_0}$ in (4.4).

4.4 Derivation discussion

The aim of message passing is to calculate Λ_τ (as defined in (4.1)) for every vertex τ. However, it is computationally hard to compute Λ_τ exactly, even for the case $m = 1$. On making some assumptions about the distribution of the graph and about conditional dependencies, the computation becomes more efficient. These assumptions do not hold for any finite-sized graph, but they are good approximations for large graphs.

Conceptually, we view the entire graph as being generated by a set of degree-growth processes. The initial vertex start their degree-growth processes at time t_o. Thereafter, every vertex that arrives starts its own degree-growth process, depending only on its label, but independent of other processes. This point of view allows us to make some principled assumptions that follow.

- **Distribution of $\partial\tau$**: We assume that the distribution of a children set is according to the law of the process $\check{Y}$:

$$\mathbb{P}\{\partial\tau = \{t_1, \ldots, t_n\} | \ell_\tau = v\} = n!\,\theta_v^n \left(\frac{\tau \vee t_o}{T} \times \prod_{t \in \partial\tau} \frac{t}{T} \right)^{\theta_v}$$

  Accordingly, given the children set $\partial\tau$, the log-likelihood ratio of the vertex label is given by λ_τ^C.

- **Dependence of $\partial\tau$ on other labels**: We assume that the distribution of the children set depends on the labels of only those vertices in the children set, i.e.:

$$\begin{aligned}&\mathbb{P}\{\partial\tau = \{t_1, \ldots, t_n\} | \ell_\tau = v, \ell_{\tau'} = u\} \\ &= \begin{cases} \mathbb{P}\{\partial\tau = \{t_1, \ldots, t_n\} | \ell_\tau = v\}\, \theta^*_{u,v}/\theta^*_v & \text{if } \tau' \in \partial\tau \\ \mathbb{P}\{\partial\tau = \{t_1, \ldots, t_n\} | \ell_\tau = v\} & \text{if } \tau' \notin \partial\tau \end{cases}\end{aligned}$$

  This implies that the posterior distribution of $\ell_{\tau'}$ given ℓ_τ and $\partial\tau$ is given by:

$$\begin{aligned}\mathbb{P}\{\ell_{\tau'} = u | \ell_\tau = v, \partial\tau\} &= \frac{\mathbb{P}\{\partial\tau | \ell_\tau = v, \ell_{\tau'} = u\}\,\mathbb{P}\{\ell_{\tau'} = u | \ell_\tau = v\}}{\mathbb{P}\{\partial\tau | \ell_\tau = v\}} \\ &= \begin{cases} \frac{\theta_{uv}\rho_u}{\theta_v} \text{ if } \tau' \in \partial\tau \\ \rho_u \text{ if } \tau' \notin \partial\tau \end{cases}\end{aligned}$$

- **Dependence of one children set on another**: Given the labels of two vertices τ_1, τ_2, the distribution of $\partial\tau_1$ is independent of $\partial\tau_2$.

However, if one (or both) of the labels are not known, $\partial\tau_1$ may convey information on $\partial\tau_2$ (and vice versa). The exact dependence is seen via an appropriate use of Bayes' rule and the two points given above.

CHAPTER 5

SIMULATIONS

In this chapter, we present numerical results of the algorithms described in Chapters 3 and 4. For graphs with a single community with outliers, we run the following algorithms:

- Degree Thresholding (DT)
- Maximum Likelihood based on children (abbreviated as "Children" or C)
- Message Passing (MP)

For the case of two communities, we present simulation results of message passing alone. In this case, the algorithm requires initialization with the labels of few of the vertices. This aspect is also discussed.

5.1 Single community with outliers

We simulate the case of a single community with outliers with the following parameters:

- T = 10,000
- $\beta = \begin{pmatrix} 4 & 1 \\ 1 & 1 \end{pmatrix}$
- $m = 5$
- $\rho = (0.5, 0.5)$

The values of θ for these parameters are: $(\theta_1 = 0.61, \theta_2 = 0.37)$.

For each of the algorithms DT, C and MP, we display the probability of error in recovering the label of vertex τ, $P_e(\tau)$, as a function of τ (see Figures

5.1, 5.2 and 5.3 respectively). Each plot is an average over 1000 random graphs. From the analysis of Chapters 3 and 4, we expect that this quantity would depend only on τ/T; hence, we scale the axis accordingly. We also plot the probability of error of type 1 and type 0 (as discussed in Chapter 3) for each of the algorithms. Finally, we compare these three algorithms in Figure 5.4, from which we see two trends:

- Algorithm C performs significantly better than Algorithm DT for vertices that arrived earlier $\tau/T < 0.1$. This is expected because Algorithm C uses more information than Algorithm DT. For vertices that arrived later, the degree growth process (effectively, the $Z - process$) does not run long enough to extract extra information.

- Algorithm MP performs much better than Algorithm C for vertices that arrived later $\tau/T > 0.1$. This is because the extra information provided by the parent's label is very useful when the duration of the degree-growth process is short.

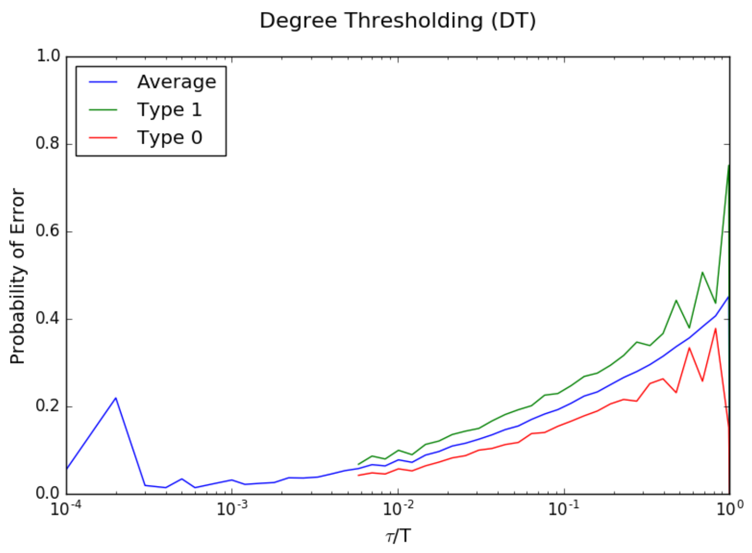

Figure 5.1: Performance of Algorithm DT

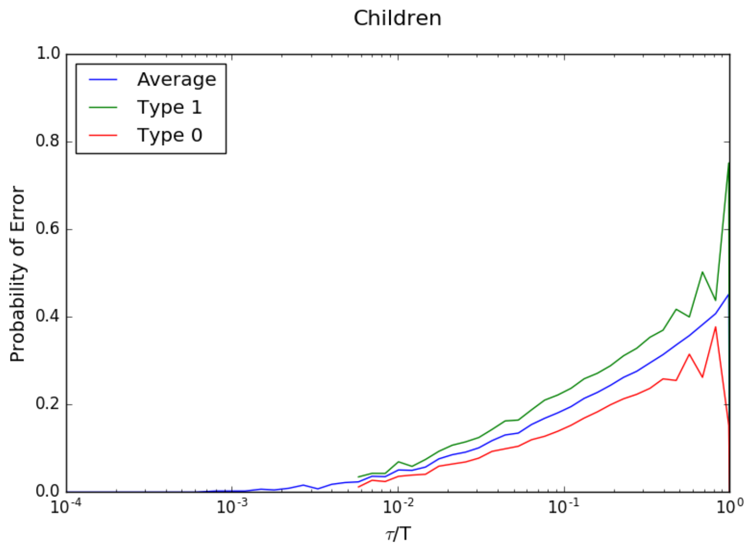

Figure 5.2: Performance of Algorithm C

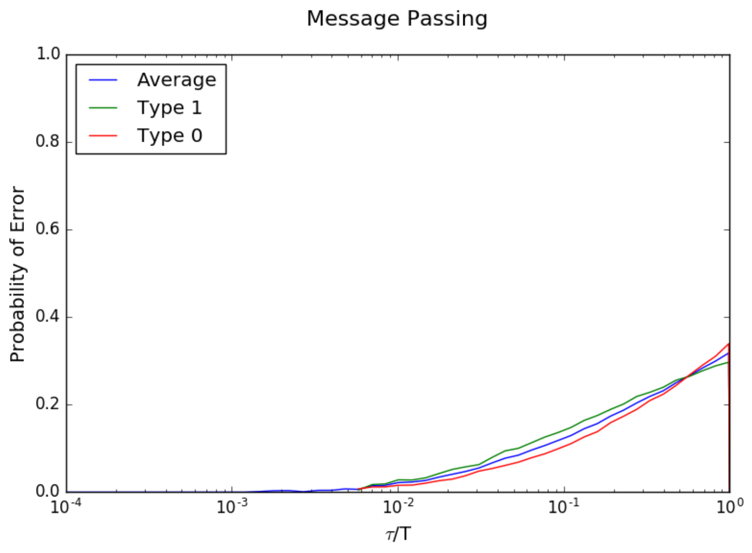

Figure 5.3: Performance of Algorithm MP

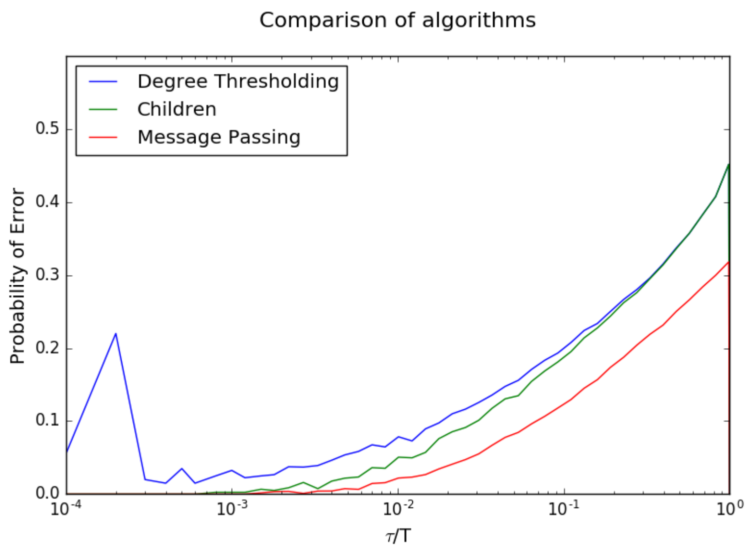

Figure 5.4: Comparison of different community detection algorithms

5.2 Two equal-sized communities

The case of two equal-sized communities poses an interesting case, as neither Algorithm C nor DT gives any information. This is because the degree growth process of every vertex has identical distribution, irrespective of their label. If we run the message passing algorithm with the initialization proposed above, we do not get any information about the communities either. This is because message passing builds upon information provided by Algorithm C.

One method to overcome this problem is to initialize message passing with the labels of a few of the vertices. This is known as semi-supervised learning, where the labels of some samples are known, but is unknown for most of the others. It is useful to initialize with the labels of the first few vertices. Since these vertices are likely to have very high degrees, the information about their label propagates to many vertices quickly. In subsequent iterations, the effect of this initial seed reduces, as every application of the function $g(\cdot)$ reduces the magnitude of the messages. Owing to the nature of the graph, most vertices are within a small distance from the initial vertices, and hence are affected by the seed.

In simulations, we find that initializing with 10 vertices work well. However, it is possible that a large fraction of these vertices are of a single type (say 8/10 are of type 1). In this case, we introduce a balancing factor to reduce the intensity of the messages of the dominant type. This ensures convergence to a solution with roughly equal number of labels of each type.

The simulations shown in Figure 5.5 are for the following parameters:

- T = 10,000
- $\beta = \begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix}$
- $m = 5$
- $\rho = (0.5, 0.5)$

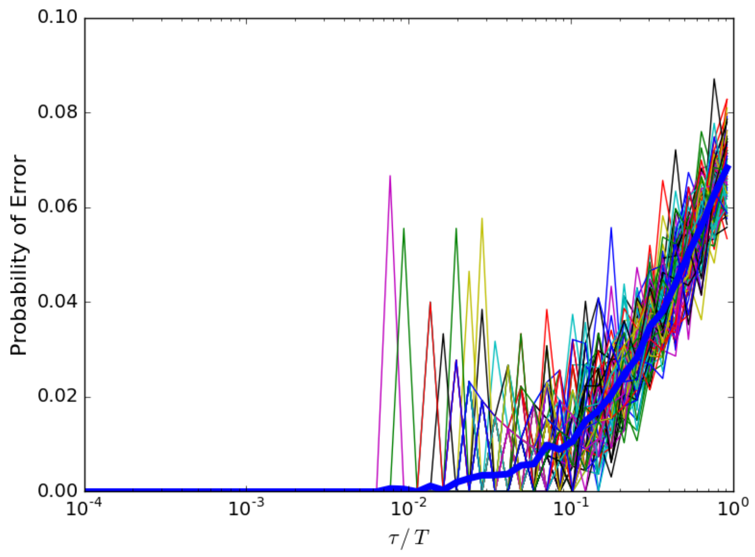

Figure 5.5: Performance of message passing in recovering two equal-sized communities. The performance plot of 100 random graphs are shown, and the average performance is depicted by the bold blue curve.

From Figure 5.5, we observe that message passing successfully recovers all the first 100 vertex labels 98% of the time, and on average, makes an error on 5%.

CHAPTER 6

CONCLUSION

In this thesis, we have presented a study of community detection in preferential attachment graphs. This study of a popular statistical problem on a new model proved to be an exciting area of research, as it provided a fertile ground for many new ideas, and also brought together interesting concepts from the SBM and the PA model. The random graph model was introduced, and some basic properties were examined. The degree-growth process, a novel viewpoint of tracking the time-evolution of the graph, was presented and a precise asymptotic characterization of the process was given. This analysis was then used to develop a hypothesis testing based algorithm for community recovery, for which an upper bound on the probability of error was given. The principles behind this algorithm were then extended to develop a message passing algorithm that runs on the graph to be clustered. The message passing algorithm was shown to have better performance for recovering a single planted community, and also gave a method to recover two equal-sized planted communities. These points were illustrated by simulations.

The thesis does not give a theoretical performance guarantee for the message passing algorithm. This remains an open question. The thesis also does not examine other algorithms for community detection, like SDP relaxations or spectral methods. We hope this shall be studied in the future. Finally, it is important to study when community recovery is and is not information-theoretically feasible.

REFERENCES

[1] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[2] E. Abbe, "Community detection and stochastic block models: Recent developments," *arXiv preprint arXiv:1703.10146*, 2017.

[3] "The graph of a social network," https://griffsgraphs.wordpress.com/2012/07/02/a-facebook-network/, accessed: 2018-05-23.

[4] C. Moore, "The computer science and physics of community detection: Landscapes, phase transitions, and hardness," *arXiv preprint arXiv:1702.00467*, 2017.

[5] B. Hajek, Y. Wu, and J. Xu, "Achieving exact cluster recovery threshold via semidefinite programming," *IEEE Transactions on Information Theory*, vol. 62, no. 5, pp. 2788–2797, 2016.

[6] E. Abbe, A. S. Bandeira, and G. Hall, "Exact recovery in the stochastic block model," *IEEE Transactions on Information Theory*, vol. 62, no. 1, pp. 471–487, 2016.

[7] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[8] B. Bollobás, O. Riordan, J. Spencer, and G. Tusnády, "The degree sequence of a scale-free random graph process," *Random Structures & Algorithms*, vol. 18, no. 3, pp. 279–290, 2001.

[9] R. Van Der Hofstad, *Random Graphs and Complex Networks*. Cambridge University Press, 2016, vol. 1.

[10] J. Jordan, "Geometric preferential attachment in non-uniform metric spaces," *Electronic Journal of Probability*, vol. 18, no. 8, pp. 1–15, 2013.

[11] F. McSherry, "Spectral partitioning of random graphs," in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on.* IEEE, 2001, pp. 529–537.

[12] B. Hajek, Y. Wu, and J. Xu, "Recovering a hidden community beyond the spectral limit in $O(|E|log^*|V|)$ time," *arXiv preprint arXiv:1510.02786*, 2015.

[13] B. Hajek and S. Sankagiri, "Community recovery in a preferential attachment graph," *arXiv preprint arXiv:1801.06818*, 2018.

[14] V. S. Borkar, *Stochastic Approximation.* Cambridge University Press, 2008.

[15] D. J. MacKay, *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, 2003.