MAJORITY VOTE COMMUNITY DETECTION WITH DYNAMIC THRESHOLD AND BOOTSTRAPPED ROUNDS

Guilherme da Costa Sales

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Daniel Ratton Figueiredo
Giulio Iacobelli

Rio de Janeiro
Março de 2019

MAJORITY VOTE COMMUNITY DETECTION WITH DYNAMIC THRESHOLD AND BOOTSTRAPPED ROUNDS

Guilherme da Costa Sales

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2019

Sales, Guilherme da Costa

Majority vote community detection with dynamic threshold and bootstrapped rounds/Guilherme da Costa Sales. – Rio de Janeiro: UFRJ/COPPE, 2019.

IX, 58 p.: il.; 29, 7cm.

Orientadores: Daniel Ratton Figueiredo
Giulio Iacobelli

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 48 – 53.

1. community detection. 2. stochastic block model. 3. majority vote. 4. label propagation. I. Figueiredo, Daniel Ratton *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

DETECÇÃO DE COMUNIDADES ATRAVÉS DO VOTO DA MAIORIA COM LIMIAR DINÂMICO E RODADAS BOOTSTRAP

Guilherme da Costa Sales

Março/2019

Orientadores: Daniel Ratton Figueiredo
Giulio Iacobelli

Programa: Engenharia de Sistemas e Computação

Detecção de comunidades é um problema fundamental em Ciência de Redes, onde os vértices de uma dada rede devem ser particionados de maneira que vértices num mesmo grupo sejam estruturalmente relacionados. Este problema encontra aplicações em diversas áreas e tem atraído muita atenção a seus aspectos práticos e teóricos. Algoritmos de propagação de rótulos (*label propagation algorithms*) se baseiam num procedimento que iterativamente atualiza a classificação de cada nó através do voto da maioria dos rótulos de comunidade de seus vizinhos. Estes algoritmos são conhecidos por serem simples e rápidos, e são muito utilizados em aplicaçoes práticas. Nesta dissertação, estudamos variações de um algoritmo de propagação de rótulos aplicado ao problema da recuperação de duas comunidades intrínsecas a uma rede (*majority vote algorithm*, ou MVA), e propomos as seguintes novas contribuições: (i) um limiar dinâmico que generaliza o limiar fixo utilizado pelo MVA, (ii) um critério de parada que resolve o problema de oscilação das soluções produzidas por algoritmos de propagação de rótulos, e (iii) estratégias de *bootstrapping* que reutilizam soluções para alcançar melhores resultados. Estas modificações dão origem a novos algoritmos de propagação de rótulos que chamamos *Global Average Majority* (GAM) e *Global Average Majority with Bootstrapping* (GAMB). Finalmente, o comportamento e a perfomance dos novos algoritmos são avaliados através de experimentos numéricos com redes sintéticas geradas pelo *stochastic block model* (SBM) e redes do mundo real com comunidades conhecidas.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MAJORITY VOTE COMMUNITY DETECTION WITH DYNAMIC THRESHOLD AND BOOTSTRAPPED ROUNDS

Guilherme da Costa Sales

March/2019

Advisors: Daniel Ratton Figueiredo
Giulio Iacobelli

Department: Systems Engineering and Computer Science

Community detection is a fundamental problem in network science, where the vertices of a given network are to be partitioned such that vertices in the same group are structurally related. This problem finds applications in a wide range of areas and has attracted much attention towards both its theoretical and practical aspects. Label propagation algorithms are based on a procedure that iteratively updates the classification of each node by a majority vote of its neighbors' community labels. These algorithms are known to be simple and fast, and are widely used in practical applications. In this dissertation, we study variations of a label propagation algorithm applied to the problem of recovering two communities embedded in a network (majority vote algorithm, or MVA), and propose the following new contributions: (i) a dynamic threshold that generalizes the fixed threshold used by the majority vote algorithm, (ii) a stopping criterion that solves the oscillation problem displayed by the solutions produced by label propagation, and (iii) bootstrapping strategies that re-utilize solutions to achieve better results. These modifications give rise to new label propagation algorithms which we call Global Average Majority (GAM) and Global Average Majority with Bootstrapping (GAMB). Finally, the behavior and performance of the new algorithms are evaluated by numerical experiments with synthetic networks generated by the stochastic block model (SBM) and real world networks with known communities.

Contents

List of Figures **viii**

List of Tables **ix**

1 Introduction **1**

1.1 Contributions 3

1.2 Document organization 4

2 Community structure **5**

2.1 What is a community? 5

2.2 Modularity 7

2.3 Random Graph Models 8

2.3.1 Erdős-Rényi (ER) model 8

2.3.2 General Stochastic Block Model (SBM) 9

2.3.3 Planted Bisection Model (PBM) 9

3 Community detection **11**

3.1 Theoretical approach 11

3.1.1 Exact recovery 13

3.1.2 Weak recovery 16

3.2 Practical Approach 17

3.2.1 Traditional approaches 17

3.2.2 Modularity optimization 17

3.2.3 Spectral methods 18

3.2.4 Dynamics based methods 18

3.2.5 Statistical inference based 18

3.2.6 Label propagation 19

4 Majority voting **20**

4.1 Majority vote algorithm 20

4.2 Global Average Majority 22

4.3 Complexity 24

4.4 Numerical evaluation 25
4.4.1 Accuracy 25
4.4.2 Other metrics 25

5 Bootstrapping 32
5.1 Complexity 34
5.2 Numerical evaluation 35
5.3 Bootstrapping MVA 38

6 Evaluation on real networks 40
6.1 Karate club 40
6.2 Political books 42
6.3 Political blogs 42

7 Conclusion 46
7.1 Future Directions 47

Bibliography 48

A Detailed statistics of GAMB rounds 54

List of Figures

1.1 Collaboration network of scientists working at the Santa Fe Institute. Source: [1] 2

4.1 Accuracy of MVA and GAM for different values of $p - q$. Each value is computed as the average over $M = 10$ instances with $R = 30$ repetitions each. 26
4.2 Number of iterations for different values of $p - q$. Each value is computed as the average over $M = 10$ instances with $R = 30$ repetitions each. 27
4.3 Additional metrics of MVA and GAM iterations. ($n = 1000$, $p = 0.01$) 29
4.4 Additional metrics of MVA and GAM iterations. ($n = 1000$, $p = 0.02$) 30
4.5 Additional metrics of MVA and GAM iterations. ($n = 1000$, $p = 0.05$) 31
4.6 Threshold value $\bar{f}$ of GAM across its iterations. The trajectories in color were chosen at random from 100 graph instances, and the thick black curve shows the average over all instances. ($n = 1000$, $p = 0.01$, $p - q = 0.006$). 31

5.1 Bootstrapping accuracy with $n = 1000$, $p = 0.01, 0.02, 0.05$ 36
5.2 Accuracy as a function of the bootstrap rounds ($n = 1000$, $p = 0.01$). 37
5.3 Accuracy and fraction of fixed vertices by bootstrap rounds ($n = 1000$, $p = 0.01$). 38
5.4 Number of iterations by bootstrap rounds ($n = 1000$, $p = 0.01$). . . . 39
5.5 Accuracy of bootstrapped MVA 39

6.1 Zachary's karate club network. Source: [1] 41
6.2 Political books network 43
6.3 Political blogs network. Source: [2] 44

A.1 Hard bootstrapping, $p = 0.01$, $p - q = 0.006$ 55
A.2 Soft bootstrapping, $p = 0.01$, $p - q = 0.006$ 56
A.3 Hard bootstrapping, $p = 0.01$, $p - q = 0.007$ 57
A.4 Soft bootstrapping, $p = 0.01$, $p - q = 0.007$ 58

List of Tables

6.1 Summary information of benchmark real world networks 40
6.2 Karate club network 41
6.3 Political books network 42
6.4 Political blogs network 45

Chapter 1

Introduction

In various domains of science and technology, one wants to understand systems composed of many interacting elements. Often times, not only the elements themselves and the nature of their relations are important, but also the pattern of interactions between them. Out of concerted efforts to better understand the role of structure in this kind of phenomena was born the discipline now called *Network Science*.

In the simplest form, a network (or a graph) is just a collection of points called vertices or nodes joined together by edges. The power of this abstract definition lies on the fact that these nodes and edges can represent many different entities and relations between them, and thus the study of this mathematical object can lead to widely applicable results.

An important example of system studied under this framework is the Internet, in which vertices represent computers and other devices like servers and routers, and the edges represent physical connections capable of transmitting data between those devices. Since the Internet was built and is maintained until the present day by many independent groups of people, with little centralized control, its structure and how it affects its function are not fully understood. To give a concrete example of how the structure influences function, consider how two computers in different parts of the network (perhaps in different continents) communicate. To establish and maintain a connection, the two computers exchange data packets, which are transmitted through a series of routers and servers that compose the infra-structure of the network. A first question can be easily posed: is the network connected? That is, given any two vertices A and B, is there a sequence of intermediary nodes connected by edges (also called a path) linking point A to point B? If not, then communication between them is impossible. Are there multiple paths connecting A to B? If not, then a the communication between A and B will be succeptible to failures happening in the nodes and edges between them. A less trivial question involves the dynamics of packets sent through the network. Given that multiple devices want to send and receive data at the same time, and that middle equipment cannot always support the volume of traffic received, how can communications protocols be engineered to ensure fast and reliable transmission of data without congesting too much the network?

A very different kind of network that is intensely studied are social networks. Although nowadays the term social network is more frequently associated with online services such as *Facebook* and *Instagram*, researchers interested in the dynamics of social interactions of groups of people have been acquiring and analyzing related data since 1930s [3]. In this context, nodes can represent anything from individuals to institutions and markets, and edges may correspond to professional or intimate relationships, financial transactions, and many others. The questions posed on these kinds of networks can involve the diffusion of ideas and behaviors, spread of diseases, formation and evolution of associations, and robustness of the financial system, to give a few examples.

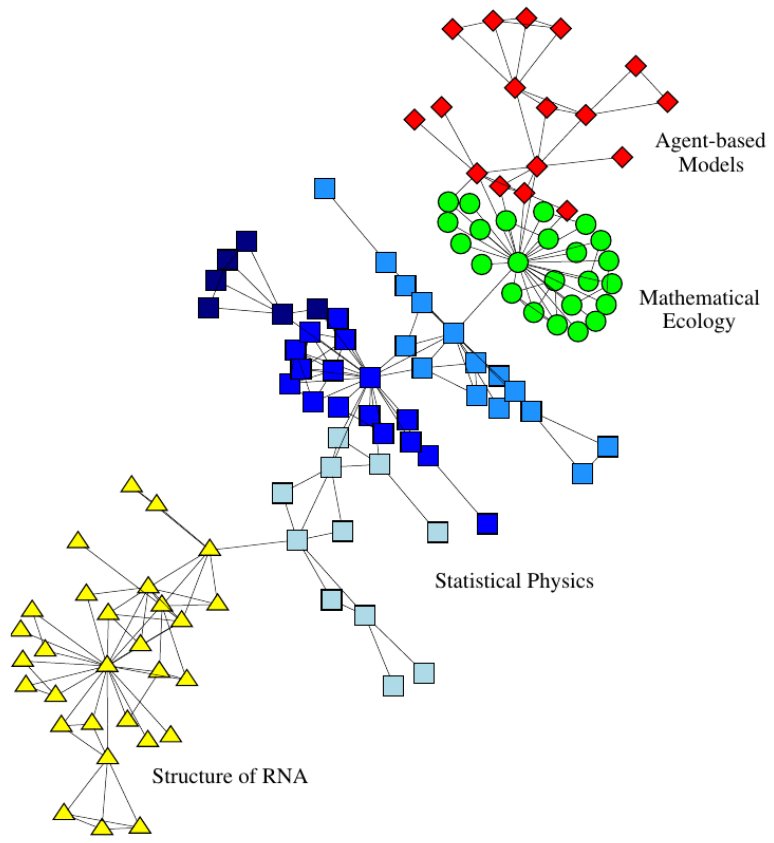


Figure 1.1: Collaboration network of scientists working at the Santa Fe Institute. Source: [1]

A prominent property in many networks is the presence of groups of vertices that are more densely connected among themselves than with the rest, which are called *communities* or, depending on the context, also *modules* or *clusters*. This organization often correlates with important information about the network, such as identity, spatial location, and function of the vertices. Consider for instance the collaboration network of scientists working at the Santa Fe Institute (SFI) in New Mexico, United States, depicted in Figure 1.1. In this network vertices represent scientists, an edges links two scientists if they co-authored an article, and the different node labels indicate the research areas of the scientists. It is clearly seen that co-authorships tend to involve scientists working in the same area, with only a few edges joining different areas, which illustrates the concept of community mentioned above.

An important example of application of community structure knowledge is in the context of biochemical networks. Many systems in the body are adequately represented as networks, such as interacting proteins, genes, or cells. Mechanisms such as protein production, metabolism, and cell regulatory behavior are fundamentally dependent on the patterns of interactions between these components, and knowledge of the network structure facilitates in searching for genes that are related to a disease, proteins that can be combined together to form new molecular products, and even in designing experiments to test unknown interactions. Since these networks are often too large to investigate exhaustively and related components tend to form clusters, community structure can serve as a very useful guide [4].

Other applications of community detection include product recommendation systems [5], medical prognosis [6], image segmentation [7], product-customer segmentation [8], and many others. Detecting the presence and recovering the composition of communities is therefore a fundamental problem in the study of networks and its applications.

As the networks in each domain of study may possess very different properties, there is no single clear-cut definition of what constitutes a community. In consequence, many models and algorithms have been proposed, and choosing the "right" one is largely a matter of adequacy to the task at hand. Even then, understanding which approaches are adequate for any specific task is still very much an open problem, and the subject of ongoing research.

One very popular type of algorithm used to recover communities known for its simplicity and speed, is the label propagation algorithm [9], which starts from an arbitrary initial configuration of community labels, and iteratively updates the label of each node based on the majority vote of the neighbors' labels. This algorithm provides the basis and inspiration for this work.

1.1 Contributions

In this dissertation, we study variations of a label propagation algorithm applied to the problem of recovering two communities embedded in a network (majority vote algorithm, or MVA), sometimes called *graph bisection*. We identify in this work the following contributions:

- A generalization of the fixed threshold that MVA uses to determine label assignments. In particular, the basic algorithm that we propose, Global Average Majority (GAM), uses a time varying threshold given by the global average of local fractions at each iteration. Besides giving a formal definition, some theoretical properties are established, connecting it to the MVA under some restrictions. This method also introduces a new stopping criterion that deals with the cycling problem displayed by solutions produced by label propagation.

- Two techniques to improve classification accuracy by re-utilizing the solutions obtained in successive GAM rounds, which we call *hard* and *soft bootstrapping*. This gives rise to a new method Global Average Majority with Bootstrapping (GAMB).

- A thorough evaluation of GAM and GAMB under the planted bisection random graph model highlighting important properties, as well as indicating its superiority in comparison to other approaches, such as higher accuracy and comparable running times. We also evaluate the performance of the algorithms on a couple of real world networks in which a meaningful two-community structure is known.

The first two contributions are independent from each other, and show significant advantages over the simple majority voting. These ideas could in principle be applied to other popular community detection algorithms, and we believe that they provide fresh ideas to community detection in general.

The research in this work originated an article submitted for publication in the 2019 ACM SIGMETRICS / IFIP Performance conference.

1.2 Document organization

The remainder of this dissertation is organized as follows. Chapter 2 covers the concept of community structure and related definitions and models. Chapter 3 reviews theoretical and practical approaches to the problem of community detection. Chapter 4 presents GAM along with some basic properties, and numerical evaluation on artificial networks. In Chapter 5, GAM is enhanced with a bootstrapping procedure and its performance evaluated, and in Chapter 6 the algorithms are then evaluated on three real world networks. Finally, conclusions and future perspectives are presented in chapter 7.

Chapter 2

Community structure

In this work we shall focus on networks represented by *unweighted, undirected simple graphs.* That means a network is represented by a graph $G = (V, E)$ composed by a finite set of vertices V – which unless specified otherwise can be taken to be the set of integers $[n] := \{1, 2, \dots, n\}$ for some $n \in \mathbb{N}$ – and a set of edges E consisting of pairs of distinct elements of V. We shall denote (u, v) the edge between $u, v \in V$, noting that the order in which the vertices are written is irrelevant, i.e (u, v) is the same as (v, u) (undirected), only one edge may exist between two distinct vertices and no self-loops are allowed (simple), and no weight is given to the edges (unweighted). Throughout this work, we will generally assume that the graphs under consideration are *connected.*

Letting $n = |V|$ be the number of vertices in G, we shall speak of the *adjacency matrix* of G as the binary $n \times n$ matrix A such that $A_{ij} = 1$ when $(i, j) \in E$, and $A_{ij} = 0$ otherwise. Note that since the edges are undirected, the adjacency matrix is symmetric. The *degree* of a vertex $i \in V$ is defined as $d_i := \sum_{j \in V} A_{ij}$, and it is easy to see that if $m = |E|$ is the number of edges in G, then $\sum_{i \in V} d_i = 2m$.

If C is a subset of V and $i \in V$, we also define the *internal* and *external degrees* of i with respect to C as $d_i^{int} := \sum_{j \in C} A_{ij}$ and $d_i^{ext} := \sum_{j \in V \setminus C} A_{ij}$.

2.1 What is a community?

Traditionally, the idea behind the concept of community is that of a subset of vertices more densely connected within itself than with the rest of the network. This notion is sufficiently vague so as to allow many different formalizations. One of the first to be proposed relied on the concept of a *clique*, which is a subset of vertices $K \subset V$ such that there is an edge between any pair of vertices in K. This definition is evidently too strict, since many groups of vertices that may be considered communities do not display such regular connections. Other definitions based on similar concepts such as n-clans, n-clubs, and k-plexes have also been put forward (see [1] and references therein). RADICCHI *et al.* [10] proposed the following two more flexible definitions, which we describe below.

- **A community in a strong sense** is a subset C of V such that

  $$d_i^{int} > d_i^{ext} \quad \text{for all } i \in C$$

  That is, every vertex in C has more connections to other vertices in C than to the rest of the graph.

- **A community in a weak sense** is a subset C of V such that

  $$\sum_{i \in C} d_i^{int} > \sum_{i \in C} d_i^{ext}$$

  In other words, the sum of all degrees within C is greater than the sum of degrees toward the rest of the network.

Even this definition has tangible drawbacks: for instance, a sufficiently large subset of vertices may be considered a community in the strong or weak sense regardless of internal density, and the union of disjoint communities is also a community, even when there are no edges between them [11]. A similar definition which has gathered some interest is due to HU *et al.* [12]. The issue of properly defining communities has also been brought up by COSCIA *et al.* [13].

Going in the opposite direction, more recent efforts have avoided altogether the question of exactly defining a community, and have instead relied on *quality functions* that assess how “community-like” a given subset of vertices is, or on the comparison with known clusters in real or artificial benchmark networks (often called *ground truth* communities) by *similarity metrics* [1, 14].

One of the most popular quality functions is undoubtedly the *modularity* due to NEWMAN and GIRVAN [15, 16]. This concept has been explored from many different viewpoints, and inspired the development of various algorithms and related metrics. It also displays a prominent characteristic of modern community detection methods, which is the reduced focus on counting and comparing the number of internal and external edges in subgraphs, and instead focusing on the probability of two vertices sharing a link with respect to a probabilistic model of how the edges are formed.

This leads us to the *stochastic block model* (SBM). It first appeared in social network literature [17], in mathematics as the inhomogeneous random graph [18], in computer science as the planted partition model [19, 20], and is arguably the simplest and most studied probabilistic network model with embedded community structure.

In the next sections, we provide more details on these two concepts which are central to community detection.

2.2 Modularity

The concept of *modularity* was proposed by NEWMAN and GIRVAN [16] as a way to quantify the community strength of a partition of vertices found by an algorithm, something particularly important in practical scenarios where community detection algorithms are applied to networks with unknown community structure. Modularity is based on the property of *assortative mixing* [15], that is, if vertices v in a network belong to a class or possess a characteristic c_v, and if the network displays assortative mixing, then vertices of the same class or characteristic tend to be more connected than would be if connections were made at random, independently of the vertex classes.

To be more precise, let $c_i \in [k] = \{1 \dots k\}$ denote the class of a vertex $i \in V$. The total number of edges between vertices of the same class can be written as

$$\sum_{(i,j)\in E} \mathbb{1}(c_i = c_j) = \frac{1}{2}\sum_{i,j} A_{ij}\, \mathbb{1}(c_i = c_j)$$

The expected number of edges if connections were random is computed following a random graph model called *configuration model.* In this model, the degree d_i of each vertex is fixed, and thus so is the number of edges in the network, $m = \frac{1}{2}\sum_{i\in V} d_i$. Each vertex i then has d_i "stubs" of edges, and the edge set is realized by connecting pairs of stubs uniformly at random, until all the stubs have been used. Thus every possible matching of stubs is generated with the same probability. Technically, this may produce networks with self-loops and multiple edges between two vertices. However, when degrees are bounded, the probability of multiple edges and self-loops goes to zero as the number of edges grows, which in practice means that for large networks these effects can in general be ignored. The probability of an edge between vertices i and j can be computed as

$$k_i \frac{k_j}{2m-1} \approx \frac{k_i k_j}{2m}$$

Modularity (Q) is then the difference between the number of edges between vertices of the same class in the network and in the configuration model, divided by the total number of edges:

$$Q = \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\mathbb{1}(c_i = c_j)$$

The quantity Q takes positive values when there are more edges between vertices of the same class than what would be expected by chance, and negative otherwise. It is strictly less than 1, even in a "perfectly mixed" network, and depending on sizes of classes and degrees it might even be considerably less. It is possible then to normalize by the modularity of this perfectly mixed value, though this is not usually done in practice [3].

Modularity is often used to evaluate communities found by various algorithms [21, 22], and also the basis of a popular class of *modularity maximization* algorithms [8, 23, 24].

2.3 Random Graph Models

The main benchmark graphs used in this work are stochastic block models with two symmetric commmunities, also known as planted bisection models. Stochastic block models are naturally understood as a generalization of Erdős-Rényi random graphs with an embedded community structure. In this section we define and review relevant properties of these models.

2.3.1 Erdős-Rényi (ER) model

This is one of the simplest random graph models one can conceive of, which is both a precursor and a particular case of the general SBM. It was first proposed by Edgar Gilbert in 1959 [25], but received the name of Erdős and Rényi who at the time studied a similar model. It is useful to us as a *null model* of a network that has no underlying community structure and as building block of the SBM.

Definition 1 (Erdős-Rényi random graph $G(n, p)$)**.** *Let n be a positive integer, and $p \in [0, 1]$. A graph G is said to be drawn according to $G(n, p)$ if it has n vertices and each pair of distinct vertices has probability p of being connected by an edge.*

A few important properties of the $G(n, p)$ model are collected below

Proposition 2. *Let $G \sim G(n, p)$. Then*

1. *The degree distribution of a fixed vertex i in G is given by $d_i \sim Binom(n-1, p)$. Thus, the expected degree of every vertex is $(n-1)p$, which is also the expected average degree in the graph.*

2. *The number of edges $m = |E|$ is also a binomial random variable, with distribution $\text{Binom}(\binom{n}{2}, p)$. Therefore the expected number of edges in G is $\frac{n(n-1)}{2}p$, and the expected graph density (ratio of edges present to the total of possible edges) is p.*

3. *If* $p_n = c\frac{\log n}{n}$ *and* $G_n \sim G(n, p_n)$*, then* $\lim_{n\to\infty} \Pr\left[G_n \textit{ is connected}\right] = 1$ *if and only if* $c > 1$*. We say in this case that the property of* G *being connected happens asymptotically almost surely (aas), and that* $p(n) = n^{-1}\log(n)$ *is a threshold function for this property.*

2.3.2 General Stochastic Block Model (SBM)

In the general SBM, each of the n vertices in the graph is randomly assigned to one of k communities (k fixed) according to a specified probability vector $p = (p_1, \ldots, p_k)$. Then, given the vector of community labels $\sigma \in [k]^n$, each edge (i, j) has probability W_{c_i,c_j} of appearing in the graph, where the matrix of probabilities W is also a parameter of the model. Below we give the formal definition as presented in [26].

Definition 3 (General Stochastic Block Model (SBM))**.** *Let* $n, k \in \mathbb{N}$*,* $p = (p_1, \ldots, p_k)$ *a probability vector over* $[k]$*, and* W *a* $k \times k$ *symmetric matrix with coefficients in* $[0, 1]$*. We say that the pair* (G, σ) *is drawn under* $\mathrm{SBM}(n, p, W)$ *if* σ *is an* n*-dimensional random vector with iid components distributed under* p*, and* G *is an* n*-vertex simple graph such that the vertices* i, j *are connected with probability* W_{σ_i,σ_j}*.*

The SBM is called *symmetric* when the probability vector p is uniform and W takes a constant value q_{in} on the diagonal, and a constant value q_{out} outside the diagonal, denoted $\mathrm{SSBM}(n, k, q_{in}, q_{out})$. This means that two vertices that belong to the same community are adjacent with probability q_{in}, and vertices in any two distinct communities are adjacent with probability q_{out}.

Another possibility commonly considered in the SBM is to require σ to be drawn such that $\frac{1}{n}|\{v \in [n] : \sigma_v = i\}| = p_i$, which in the symmetric case amounts to $|\{v \in [n] : \sigma_v = i\}| = n/k$ (for suitable values of n, k) and is referred to as being *strictly balanced.*

2.3.3 Planted Bisection Model (PBM)

By far the most studied instance of the SBM is the strictly balanced SSBM with $k = 2$ communities, and it is the one on which we focus on this work. This model will be used both to provide intuition as well as for numerical evaluation of the algorithms. For completeness, a formal definition is given below, where we note that since the probability vector p is not used, we use p to denote the intra-community edge probability (q_{in}), and q for the inter-community edge probability (q_{out}).

Since we are interested in clusters of vertices more densely connected inside the cluster than with the rest of the network, we will in general assume that $p > q$ (although the model imposes no restriction other than $p, q \in [0, 1]$). For simplicity of notation, we denote n the number of vertices in each community, thus the total number of vertices in the graph is $2n$.

Definition 4 (Planted Bisection Model (PBM)). *Given $n \in \mathbb{N}$ and $p, q \in [0,1]$, we define a random graph with $2n$ nodes $(G, \sigma) \sim \mathrm{PBM}(2n, p, q)$, by first choosing a balanced labelling σ (uniformly at random from the set $\{\tau \in \{0,1\}^{2n} : \sum_{i=1}^{2n} \tau_i = n\}$) and then randomly drawing the set of edges of $G = (V, E)$, where $V = \{1, \ldots, 2n\}$, according to*

$$\Pr\big[\{i,j\} \in E \mid \sigma\big] = \begin{cases} p, & \text{if } \sigma_i = \sigma_j \\ q, & \text{if } \sigma_i \neq \sigma_j \end{cases} \quad \text{for all } i, j \in V,\ i \neq j$$

We now remark a few properties of this model.

Proposition 5. *Let $(G, \sigma) \sim \mathrm{PBM}(2n, p, q)$.*

1. *The subgraph induced by the vertices in each community is an ER random graph with parameters n, p.*
2. *The number of edges inside each community is distributed as a $\mathrm{Binom}(\binom{n}{2}, p)$ random variable, thus the total of intra-community edges has distribution $\mathrm{Binom}(n(n-1), p)$, with expected value $n(n-1)p$.*
3. *Since there are n vertices in each community, the number of inter-community edges has distribution $\mathrm{Binom}(n^2, q)$. Thus the expected number of edges between communities is $n^2 q$. The expected number of edges in the whole graph is therefore $n(n-1)p + n^2 q$.*

It is easy to see that, according to the notion of community we have specified, the parameter $p - q$ is roughly proportional to the difference of intra-communty and inter-community edges (for fixed n), and may be taken as a measure of the "strength" of the community structure in the PBM model. Notice that in the degenerate case $p = q$, the probability of an edge between two vertices is independent of community assignment, and there is effectively no community structure in the model, which is then equivalent to a $G(2n, p)$ random graph.

Chapter 3

Community detection

As hinted towards in the previous chapters, the general goal of community detection is to find groups of vertices in the network that are somehow more connected within the group than with the rest of the network. There are many different approaches to this task. Some are more principled, such as the ones which define exactly the conditions under which a subset of vertices is considered a community (e.g clique) and then attempt to find such subsets in the network, or the ones which define a quality function for the strength of community structure and then finds vertex clusters by optimization. Others are heuristic: a procedure is defined that outputs clusters of vertices based on the features of the network, and the quality of such groups is assessed *a posteriori* by quality functions and/or by comparison to known communities in benchmark networks via *similarity metrics*.

Next, we review some relevant theoretical results and practical approaches to community detection.

3.1 Theoretical approach

On the theoretical side of community detection, we are mainly interested in results that ascertain under which conditions the following two kinds of problems can be solved:

1. *Detection (also called distinguishability or testing).* Given a network (which may be an instance of a random graph model), determine if there exists non-trivial community structure present in the network (or model).

2. *Recovery (or reconstruction).* Given a network with non-trivial community structure, find a partition of the vertex set that maximizes the agreement with the planted communities.

These questions have been more mainly studied in the stochastic block model. As said before, the reason for this is that the SBM is simple enough so that it allows analytical treatment, yet it captures very well our intuitive notion of community and is flexible enough to display many interesting phenomena. In this work we are only concerned with the recovery problem.

To provide a formal definition of the problem in the SBM, we first define how we compare two different vectors of community labels. We first state the definition in the case of the general SBM, and then what it amounts to in the PBM. The similarity between two vectors of community labels (community vectors) is computed simply by the fraction of vertices on which the two community labels agree. This is considered up to any permutation of the community labels, since what matters is the subset of vertices that compose each community and not the assigned labels.

Definition 6. *Let $\sigma, \sigma' \in \{0,1\}^{[2n]}$ be two strictly balanced community vectors. We define the* agreement *between σ and σ' as*

$$\mathcal{A}(\sigma, \sigma') = \max \left\{ \frac{1}{2n} \sum_{i=1}^{2n} \mathbb{1}(\sigma_i = \sigma'_i),\ \frac{1}{2n} \sum_{i=1}^{2n} \mathbb{1}(\sigma_i = 1 - \sigma'_i) \right\}.$$

Let σ be a community vector and denote $\mathcal{P}(\sigma) = \{\mathcal{P}_i(\sigma)\}_{i \in [k]}$ with $\mathcal{P}_i(\sigma) = \{v \in [n] : \sigma_v = i\}$ the partition encoded by σ. Then we can see that

$$\mathcal{A}(\sigma, \sigma') = 1 \quad \text{if and only if} \quad \mathcal{P}(\sigma) = \mathcal{P}(\sigma') \tag{3.1}$$

thus corresponding to equality of the encoded partitions. If $\hat{\sigma}$ is an estimate of the ground truth labels σ of any SBM instance, we shall refer to the quantity $\mathcal{A}(\hat{\sigma}, \sigma)$ as the *accuracy* of $\hat{\sigma}$.

We will now state formally and then describe some of the most important reconstruction problems in the PBM.

Definition 7. *Let $\{p_n\}, \{q_n\}$ be sequences in $[0,1]$, and $(G_n, \sigma_n) \sim \mathrm{PBM}(2n, p_n, q_n)$. Omitting the subscript n for clarity, we say that the following recovery requirements are solved if there exists an algorithm that takes G as an input and outputs $\hat{\sigma}$ such that*

- Exact recovery: $\Pr\{\mathcal{A}(\sigma, \hat{\sigma}) = 1\} = 1 - o_n(1)$
- Almost exact recovery: $\Pr\{\mathcal{A}(\sigma, \hat{\sigma}) = 1 - o_n(1)\} = 1 - o_n(1)$
- Weak recovery: $\Pr\{\mathcal{A}(\sigma, \hat{\sigma}) \geq 1/2 + \Omega_n(1)\} = 1 - o_n(1)$

Moreover, for any of the above conditions, when all that is known is that there exists an algorithm that satisfies it, we say that the condition is *information-theoretically* achived; when there is an algorithm polynomial in n that satisfies the condition, then we say that it is *efficiently* achieved.

Put in words, the above definitions mean that exact recovery requires that exact reconstruction of the partition; almost exact recovery allows for a vanishing fraction of misclassified vertices; and weak recovery only asks for an estimator performing better than random guesses. All of the above must be satisfied only asymptotically *with high probability*, i.e with probability tending to 1 as n goes to infinity. Note that there are also analogous versions of the above statements in the more general SBM [26].

Let us comment on the motivation for the high probability requirement. If $(G, \sigma) \sim \mathrm{PBM}(2n, p, q)$, then given any realization of σ (which we will call the ground truth labels), and assuming that p, q are strictly between 0 and 1, it is easy to see that any graph on $2n$ vertices can be generated, and reciprocally, a realization G of the graph can correspond to any one of the strictly balanced ground truth community vectors. Thus, a procedure that outputs an estimate $\hat{\sigma}$ of the ground truth labels using only the graph G cannot be always accurate. In this situation, the best that we can hope for is to maximize the agreement between σ and $\hat{\sigma}$ in probability.

Many other related problems and approaches have been studied. We refer to [26] for a comprehensive survey of recent results.

3.1.1 Exact recovery

Before stating the main result concerning exact recovery in the planted bisection model, let us remark an easy necessary condition. Adapting property 3 of the ER model in Proposition 2, it is easily concluded that PBM is connected aas if and only if $p_n, q_n \geq (1+\varepsilon)\log(n)/n$ for some $\varepsilon > 0$. Since it is not possible to exactly recover the partition of a disconnected graph using the topology alone, this gives necessary condition for exact recovery. When $p_n = a\log(n)/n$, $q_n = b\log(n)/n$ for constant $a, b > 0$, we say that the PBM is in the *logarithmic degree regime* (recall from Proposition 5 the expected average degree in the PBM).

Theorem 8. *[27] Consider sequences* $\{p_n\}$, $\{q_n\}$ *in* $[0,1]$ *such that* $p_n = a_n \log(n)/n$ *and* $q_n = b_n \log(n)/n$*, where* a_n, $b_n \in \Theta(1)$*. There exists an algorithm that achieves exact recovery for* $\mathrm{PBM}(2n, p_n, q_n)$ *if and only if*

$$(a_n + b_n - 2\sqrt{a_n b_n} - 1)\log n + \frac{1}{2}\log\log n \to \infty \tag{3.2}$$

From the result above we can recover the result in the logarithmic regime by ABBE *et al.* [28]:

Theorem 9. *Exact recovery in* $\mathrm{PBM}(2n, a\log(n)/n, b\log(n)/n)$ *is solvable and efficiently so if* $(a+b)/2 - \sqrt{ab} > 1$ *and unsolvable if* $(a+b)/2 - \sqrt{ab} < 1$.

MAP estimation and minimum-bisection

Let $\mathcal{P}$ stand for the ground truth partition corresponding to $(G, \sigma) \sim \mathrm{PBM}(2n, p, q)$ and $\hat{\mathcal{P}}(G)$ the partition corresponding to estimated community labels $\hat{\sigma}(G)$. To achieve exact recovery, we want the estimator $\hat{\sigma}(G)$ to maximize $\Pr\{\mathcal{A}(\sigma, \hat{\sigma}) = 1\} = \Pr\{\mathcal{P} = \hat{\mathcal{P}}(G)\}$. By the law of total probability, we may write

$$\Pr\{\mathcal{P} = \hat{\mathcal{P}}(G)\} = \sum_g \Pr\{\mathcal{P} = \hat{\mathcal{P}}(G) \,|\, G = g\} \Pr\{G = g\}$$

Therefore the best we can do is to find an estimator that maximizes $\Pr\{\mathcal{P} = \hat{\mathcal{P}}(G) \,|\, G = g\}$ for every possible realization g. This is the *maximum a posteriori* (MAP) estimator of $\mathcal{P}$.

Let us emphasize that by its very definition, the MAP estimator maximizes the probability of recovering the entire partition. Thus *if MAP fails, then no other algorithm can achieve exact recovery.* This is true not only for the PBM, but for the general SBM with k communities.

In a strictly balanced symmetric SBM, it is easy to see that $\Pr\{\mathcal{P} = \pi\}$ is uniform over the set of permissible partitions π. Using Bayes' Theorem, we can write

$$\Pr\{\mathcal{P} = \pi \,|\, G = g\} \propto \Pr\{G = g \,|\, \mathcal{P} = \pi\} \Pr\{\mathcal{P} = \pi\},$$

thus MAP is equivalent to *maximum likelihood estimation.*

Now we consider specifically the planted bisection case. Observing that for any two distinct $i, j \in [n]$ we can write

$$\Pr\{(i,j) \in E(G) \,|\, \mathcal{P} = \pi\} = \begin{cases} p, & \text{if } \{i,j\} \subset \pi_\ell \text{ for } \ell = 1 \text{ or } \ell = 2 \\ q, & \text{otherwise} \end{cases}$$

and letting N_{in}, N_{out} be respectively the numbers of intra-community and inter-community edges, then $M := |E(G)| = N_{in} + N_{out}$, and

$$\begin{aligned} \Pr\{G = g \,|\, \mathcal{P} = \pi\} &= \prod_{1 \le i < j \le M} \Pr\{(i,j) \in E(G) \,|\, \mathcal{P} = \pi\} \\ &= p^{N_{in}}(1-p)^{M - N_{in}} q^{N_{out}} (1-q)^{M - N_{out}} \\ &= p^M (1-q)^M \left[\frac{(1-p)q}{p(1-q)}\right]^{N_{out}} \end{aligned} \qquad (3.3)$$

M depends only on G, and N_{out} is a function of G and $\mathcal{P}$. In a rather cumbersome way, we can explicitly write

$$N_{out}(G, \mathcal{P}) = \sum_{1 \le i < j \le 2n} \mathbb{1}\left((i,j) \in E(G)\right) \mathbb{1}\left(\{i,j\} \not\subset \mathcal{P}_1 \text{ and } \{i,j\} \not\subset \mathcal{P}_2\right)$$

Since $0 < q < p < 1$ implies $0 < \frac{(1-p)q}{p(1-q)} < 1$, for given $G = g$ the likelihood (3.3) is maximized when the number of inter-community edges $N_{out}(g, \pi)$ is minimized. With the restriction of balanced partitions, this is the classical *min-bisection problem.* We know that in terms of worst-case complexity, the minimum bisection problem is NP-hard [29]. However, for (efficiently) achieving exact recovery, it is sufficient that there exist a polynomial time algorithm that solves min-bisection with high probability.

The spectral algorithm

Let $(G, \sigma) \sim \mathrm{PBM}(2n, p, q)$ and A be the adjacency matrix of G. Without loss of generality, we work with community vectors in $\{-1,1\}^{2n}$. The restriction to strictly balanced partitions imposes $\sum_{v \in [2n]} \sigma_v = 0$, or equivalently $\sigma^\top \mathbf{1} = 0$, with $\mathbf{1} = (1, \dots, 1)$. Also note that the number of edges $m = |E(G)|$ is fixed, and

$$A_{ij}\sigma_i\sigma_j = \begin{cases} 1, & \text{if } (i,j) \in E \text{ and } \sigma_i = \sigma_j \\ -1, & \text{if } (i,j) \in E \text{ and } \sigma_i \neq \sigma_j \\ 0, & \text{if } (i,j) \notin E \end{cases}$$

Letting $N_{in}(\sigma), N_{out}(\sigma)$ be the number of inter and intra-community edges according to σ, it is easy to see that

$$\sigma^\top A \sigma = \sum_{i,j} A_{ij}\sigma_i\sigma_j = N_{in}(\sigma) - N_{out}(\sigma) = m - 2N_{out}(\sigma),$$

so that the min-bisection problem is equivalent to

$$\begin{aligned} \max \quad & \sigma^\top A \sigma \\ \text{subject to} \quad & \sigma^\top \mathbf{1} = 0 \\ & \sigma \in \{-1,1\}^{2n} \end{aligned} \tag{3.4}$$

The integer constraint implies $\|\sigma\|_2^2 = \sigma_1^2 + \ldots + \sigma_{2n}^2 = 2n$, which suggests the following *spectral relaxation*

$$\begin{aligned} \max \quad & \sigma^\top A \sigma \\ \text{subject to} \quad & \sigma^\top \mathbf{1} = 0 \\ & \|\sigma\|_2^2 = 2n \end{aligned} \tag{3.5}$$

Since A is symmetric, without the linear constraint $\sigma^\top \mathbf{1} = 0$, the above problem amounts to the variational characterization of the largest eigenvalue of A [30].

The solution of the problem with the linear constraint is the eigenvector corresponding to the largest eigenvalue of the projection of A on the subspace orthogonal to the vector $\mathbf{1}$. Notice that $A\mathbf{1}$ is the vector containing the degrees of the vertices, and since the binomial distribution of degrees concentrates very strongly around its mean $\mu := (n-1)p + nq$, the vector $A\mathbf{1}$ will be close to $\mu\mathbf{1}$ with high probability. Thus it is reasonable to suppose that the solution to the spectral relaxation will be close to the eigenvector $v_2(A)$ corresponding to the second largest eigenvalue of A instead. Finally, we obtain a solution with integer coefficients by truncating $v_2(A)$:

$$\hat{\sigma}_i^{\text{spec}} = \begin{cases} -1, & \text{if } (v_2(A))_i < 0 \\ 1, & \text{if } (v_2(A))_i \geq 0 \end{cases}$$

3.1.2 Weak recovery

The recent interest in weak recovery in the SBM was sparked by the following conjecture by DECELLE *et al.* [31], based on non-rigorous ideas from statistical physics.

Conjecture 10. *Let* $(G, \sigma) \sim \mathrm{SSBM}(n, k, a/n, b/n)$, *and define* $\mathrm{SNR} = \frac{(a-b)^2}{k(a+(k-1)b)}$. *Then for every* $k \leq 2$, *it is possible to solve weak recovery efficiently if and only if* $\mathrm{SNR} > 1$. *This is called the* Kesten-Stigum *threshold.*

In the case $k = 2$, the threshold is reduced to

$$(a-b)^2 > 2(a+b)$$

This conjecture was first established for the case $k = 2$ by [32, 33], who proved achievability above the threshold, and [34], that showed the impossibility below the threshold. A slightly more flexible characterization was achieved by [35]:

Theorem 11. *Weak recovery is solvable in* $\mathrm{SSBM}(n, 2, a_n/n, b_n/n)$ *when* a_n, $b_n = \pi(1)$ *and* $(a_n - b_n)^2/(2(a_n + b_n)) \to \lambda$ *if and only if* $\lambda > 1$.

3.2 Practical Approach

On the practical side, there are a great number of possible approaches to community detection, exploring different notions of community, of similarity metrics between partitions, and heuristic principles, each combination leading to different algorithms and methods of evaluation. In particular, one remarkable characteristic of practical community detection is that in general the number of clusters in a network is not known a priori [3], are often varied in connectivity and size [36], and display hierarchical structures [37, 38]. So far there is no consensus in the community on the most appropriate, although many previous works have evaluated and compared the performance of such algorithms and metrics under various conditions, and some principles for choosing a suitable algorithm have been put forward [1, 39–41].

Let us now see some different algorithms that have been proposed for practical community detection.

3.2.1 Traditional approaches

The first algorithmic approaches to community detection appeared in social network analysis literature and can be broadly divised into two types, *agglomerative*, and *divise methods* [16]. The first proceed by grouping together vertices that are similar according to some given metric. This process could be halted at any given point, and the resulting clusters would be taken to be the communities in the network. Divise methods, on the other direction, start with the full network and proceeds by removing edges between least similar pairs of vertices, eventually segmenting the network into smaller components, which are declared communities. Adapting this idea, Newman and Girvan [16] proposed their popular *Edge-betweenness* algorithm. One remarkable feature of these algorithms is the production of *dendrograms*, which display a division of networks into clusters at multiple levels. Because of this, these methods are also referred to as *hierarchical clustering algorithms.* They, however, lack a bit of justification for the significance of their results, since the network dendrograms are always constructed, regardless of the actual presence of communities.

3.2.2 Modularity optimization

To address this issue, later came algorithms based on the optimization of modularity (cf. Section 2.2). This optimization problem was shown to be NP-hard by BRANDES *et al.* [42], and several ways of approximating its solution have been proposed, such as the original greedy algorithm proposed by NEWMAN [43], *Fast-greedy* algorithm by CLAUSET *et al.* [8], *Louvain* by BLONDEL *et al.* [23], and many others. Although the concept of modularity has been shown to not always capture significant community structure [44–46], it remains very popular due to its principled nature and simplicity.

3.2.3 Spectral methods

Methods employing the eigenvectors of matrices such as the adjacency and the Laplacian matrix of a graph have been known for a long time, and go back to the problem of *graph partitioning* encountered in applications such as distribution of parallel computing workload, and VLSI circuit design. [47].

Noticing that the partitions found using the Laplacian matrix did not really correspond to natural subdivisions found in networks, NEWMAN [24, 48, 49] proposed methods based on the *modularity matrix* which have since become very popular.

3.2.4 Dynamics based methods

As a way to capture more fluid notions of community, and higher-level community structures, some methods have explored dynamic network processes like random walk diffusion, spin dynamics, and synchronization [1].

Walktrap [50] measures the similarity of two vertices using the transition probability of a random walker moving from one to the other in a fixed number of steps, and then applies traditional hierarchical clustering to detect community structure. ROSVALL *et al.* [51] proposed *Infomap*, which finds communities by minimizing the description length of a random walk. *Spinglass* [52] is based on ideas from statistical mechanics, and maps the community detection problem to energy minimization in a model of ferromagnetic and antiferromagnetic interactions (spin glass model), by simulated annealing. An interesting feature of this method is that it not only favors intra-community edges and penalizes inter-community edges, but also favors inter-community non edges and penalizes intra-community non edges.

3.2.5 Statistical inference based

Another broad class of methods finds communities by fitting statistical network models to the data. HASTINGS [53], used planted partition models with multiple communities and a *belief propagation* algorithm to perform maximum likelihood estimation of the unknown paramaters. NEWMAN *et al.* [54] use a mixture model with directed edges and the *expectation-maximization* (EM) algorithm. Similar approaches were undertaken by [55–57].

3.2.6 Label propagation (LP)

Finally, starting with the work of RAGHAVAN *et al.* [9], a new class of simple and fast methods emerged, based on the idea of *label propagation*. In this type of method, each vertex is initially given some label, and then iteratively updated according to a function of their neighbors' labels, such as the majority among them. [58, 59]

The working of the original LP algorithm can be described as follows. At first, each vertex $i \in V$ is initialized with a community label $c_i(0) = i$. Then each one has its label updated, assuming the label of the majority of its neighbors, with ties broken uniformly at random. In a synchronous version of the algorithm, all vertices have their labels updated in parallel for each iteration $t = 1, 2, \ldots$, thus $c_i(t)$ is a function of $\{c_j(t-1)\}_{j \in N_i}$, where N_i is the neighborhood of i. In the asynchronous version, at each iteration t, the vertices are randomly ordered, and their labels updated sequentially. The algorithm stops when each vertex has a label that reaches the majority among their neighbors. That is, letting $d_i^{C_\ell}$ be the number of neighbors vertex i has in community C_ℓ, and C_k be the community assigned to i at the current iteration, the process stops if each vertex i satisfies

$$d_i^{C_k} \geq d_i^{C_\ell} \quad \text{for all } \ell$$

Chapter 4

Majority voting

In this chapter we motivate and present one of the main contributions of this work, which is a generalization of the majority vote idea used by label propagating algorithms.

4.1 Majority vote algorithm

Since the original label propagation algorithm proposed [9] started with one different label for each vertex, allowing for an arbitrary number of communities, we shall start here with a modified version of the synchronous algorithm, specialized for the case of two communities, which we call *majority vote algorithm* (MVA). With two balanced communities, we suppose the graph has $2n$ vertices, and the labels are specified by a vector in $\{0,1\}^{2n}$. The initial assignment is made uniformly at random for each vertex. As does the label propagation algorithm, MVA assigns to each vertex the label assumed by the majority of its neighbors at the current iteration, breaking ties arbitrarily at random.

It will be useful to formalize the algorithm in the following manner: let A denote the adjacency matrix of G and $D = \operatorname{diag}(d_1, \dots, d_{2n})$, where d_i is the degree of node i. Given $\sigma \in \{0,1\}^{2n}$, we denote the fraction of 1-labelled neighbors of i in σ as

$$f_i(\sigma) := (D^{-1}A\sigma)_i = \frac{1}{d_i}\sum_{j\in V} A_{ij}\sigma_j \ , \tag{4.1}$$

The complementary fraction of 0-labelled neighbors of i is then given by $\frac{1}{d_i}\sum_{j\in V} A_{ij}\,(1-\sigma_j)$. Thus the majority of vertex i's neighbors has label 1 when

$$\sum_{j\in V} A_{ij}\sigma_j > \sum_{j\in V} A_{ij}\,(1-\sigma_j) \iff f_i(\sigma) > 1/2 \ .$$

Letting $\sigma(t)$ denote the community labels assigned at iteration t, the updating scheme can then be written as

$$\sigma_i(t) = \begin{cases} 1, & \text{if } f_i\left(\sigma(t-1)\right) > 1/2 \\ 0, & \text{if } f_i\left(\sigma(t-1)\right) < 1/2 \\ \sim \mathrm{Ber}(1/2), & \text{otherwise} \end{cases} \quad \text{for every node } i \in V. \tag{4.2}$$

As the LP algorithm, MVA may oscillate between states when a vertex has the same number of 1 and 0 neighbors. Instead of proceeding as in the original version of the algorithm, the MVA stops when it finds a cycle (or fixed point) on its sample path. This condition may be implemented by storing the previously visited states in a list or hash table, and checking if the new state has already been visited or not at each iteration. As the state space of all possible label assignment is finite, this stoppage criterion will always be eventually satisfied. Although in theory cycles can be very long and appear only after a very large number of iterations, in practice they are short and appear quite early (as we show in Section 4.4). The MVA pseudocode is shown in Algorithm 1.

Algorithm 1: *Majority Vote Algorithm (MVA)*

```
   Input: Graph $G = (V, E)$, initial nodes label $\sigma(0) \in \{0,1\}^V$
   Output: Estimated community labels $\widehat{\sigma}$
 1 Let $A$ be the adjacency matrix of $G$
 2 Let $D = \mathrm{diag}(d_1, \dots, d_{|V|})$
 3 HT.add($\sigma(0)$) /* hash table with key $\sigma$ */
 4 $t = 1$
 5 repeat
 6     $f = D^{-1} A \sigma(t-1)$
 7     for $i \in V$ do
 8         $\sigma_i(t) = \begin{cases} 1, & \text{if } f_i > 1/2 \\ 0, & \text{if } f_i < 1/2 \\ \sim \mathrm{Ber}(1/2), & \text{if } f_i = 1/2 \end{cases}$
 9     end
10     if HT.exists($\sigma(t)$) then break
11     HT.add($\sigma(t)$)
12 until True
13 $\widehat{\sigma} = \sigma(t)$
14 return $\widehat{\sigma}$
```

A couple of remarks about this algorithm. Note first that the initial label assignment $\sigma(0)$ is generated uniformly at random, thus no prior information concerning node labels is assumed. Also, the algorithm is synchronous in the sense that the label of all nodes are updated before the start of a next iteration. Last and most importantly, note that the MVA has a fixed threshold of 1/2 to decide label assignments, as indicated in (4.2) and line 8 of Algorithm 1. This threshold seems too rigid as it does not depend on the network structure nor the labels currently assigned to nodes. This provides the motivation for an improved version of this algorithm.

4.2 Global Average Majority

In order to overcome the stiffness in the threshold of the MVA, we propose a thresholding technique that is dynamic (across the iterations) and leverages the labels assigned in a given iteration as well as the network structure. Intuitively, the structure of a given graph instance has particularities that should be leveraged to define a more appropriate threshold.

We propose using as threshold the *average fraction of 1-labelled neighbors* (across all nodes), defined as follows. Given a label assignment $\sigma \in \{0,1\}^{2n}$ in G,

$$\bar{f}(\sigma) := \frac{1}{2n} \sum_{i \in V} f_i(\sigma) \ , \tag{4.3}$$

where $f_i(\sigma)$ is given in Equation (4.1). Note that $\bar{f}(\sigma)$ depends on the label assignment and on the graph structure via the definition of $f_i(\sigma)$, leveraging "global" information relative to this graph instance. Moreover, the averaging across all nodes provides robustness to small changes in σ.

The *Global Average Majority* (GAM) algorithm is the MVA modified to use $\bar{f}$ as threshold across its iterations. In particular, given a label assignment $\sigma(t-1)$, it computes $\sigma(t)$ recursively as follows:

$$\sigma_i(t) = \begin{cases} 1, & \text{if } f_i\left(\sigma(t-1)\right) > \bar{f}(\sigma(t-1)) \ , \\ 0, & \text{if } f_i\left(\sigma(t-1)\right) < \bar{f}(\sigma(t-1)) \ , \\ \sim \text{Ber}(1/2), & \text{otherwise} \ , \end{cases} \tag{4.4}$$

for every node $i \in V$. Note that this new threshold condition means that if the fraction of 1-labelled neighbors of i in $\sigma(t-1)$ is greater than the average fraction of 1-labelled neighbors in the graph, then assign label 1 to node i. Otherwise assign label 0, or break a tie by flipping a fair coin. For completeness, GAM is shown in Algorithm 2. Note that lines $14--15$ will be be used later, as information needed to drive the next algorithm, and can be ignored for now. Last, GAM will surely terminate since it must revisit a label assignment, given that the state space (i.e., all possible label assignments) is finite.

Some observations about the GAM and the threshold function $\bar{f}$ are stated below.

Algorithm 2: *Global average majority (GAM)*

```
Input: Graph $G = (V, E)$, initial nodes label $\sigma(0) \in \{0,1\}^V$
Output: Estimated community labels $\widehat{\sigma}$
        Subset of nodes $S$
1  Let $A$ be the adjacency matrix of $G$
2  $D = \mathrm{diag}(d_1, \ldots, d_{|V|})$
3  HT.add($\sigma(0), 0$) /* hash table with key-value pair */
4  $k = 1$
5  repeat
6      $f = D^{-1}A\sigma(t-1)$
7      $\bar{f} = \frac{1}{|V|}\sum_{i \in V} f_i$
8      for $i \in V$ do
9          $\sigma_i(t) = \begin{cases} 1, & \text{if } f_i > \bar{f} \\ 0, & \text{if } f_i < \bar{f} \\ \sim \mathrm{Ber}(1/2), & \text{if } f_i = \bar{f} \end{cases}$
10     end
11     if HT.exists($\sigma(t)$) then break
12     HT.add($\sigma(t), k$)
13 until True
14 $j =$ HT.get($\sigma(t)$)
15 $S = \{i \in V : \sigma_i(\ell) = \sigma_i(t) \text{ for all } j \le \ell < k\}$
16 $\widehat{\sigma} = \sigma(t)$
17 return $\widehat{\sigma}$, $S$
```

Proposition 12. *Let $G = (V, E)$ be graph on $2n$ vertices and $\sigma \in \{0,1\}^{2n}$ be a label assignment on G. Then, it holds that*

$$\frac{d_{\min}}{d_{\max}}\,\bar{\sigma} \le \bar{f}(\sigma) \le \frac{d_{\max}}{d_{\min}}\,\bar{\sigma}\ ,$$

where $\bar{\sigma} := \frac{1}{2n}\sum_{i \in V}\sigma_i$ and $d_{\min}, d_{\max}$ denote the minimum and maximum degrees of G, respectively.

In particular, if G is regular, then for all label assignment σ it holds that $\bar{f}(\sigma) = \bar{\sigma}$. Thus, if we only consider balanced label assignment on regular graphs, $\bar{f}(\sigma) = 1/2$, i.e., the GAM threshold reduces to MVA.

Proof of Proposition 12. Let us write $i \sim j$ if i and j are neighbors.

$$\bar{f}(\sigma) = \frac{1}{2n}\sum_{i \in V}\sum_{j \sim i}\frac{\sigma_j}{d_i} \ge \frac{1}{2n}\sum_{i \in V}\sum_{j \sim i}\frac{1}{d_{\max}}\sigma_j = \frac{1}{(2n)\,d_{\max}}\sum_{i \in V}\sum_{j \sim i}\sigma_j\ .$$

Since, each j contributes $d_j\sigma_j$ in the sums on the RHS, we have that

$$\bar{f}(\sigma) \geq \frac{1}{(2n)\,d_{\max}} \sum_{j\in V} d_j\sigma_j \geq \frac{d_{\min}}{(2n)\,d_{\max}} \sum_{j\in V} \sigma_j = \frac{d_{\min}}{d_{\max}}\bar{\sigma}\ .$$

The proof of the upper bound is similar. □

GAM enjoys an interesting property: if a label assignment is a fixed point of (4.4) (no ties occur), then the swapped assignment (i.e., exchanging 1 for 0, and vice-versa) is also a fixed point. This is important because the algorithm should also stop with a swapped label assignment, as this is irrelevant from the perspective of assessing the accuracy of the assignment returned by the algorithm.

The following proposition shows that the fixed points of (4.4) have this property. In passing, we remark that the MVA also has the same property.

Proposition 13. *Let $G = (V, E)$ be a graph on $2n$ vertices and $\sigma \in \{0,1\}^{2n}$ be a label assignment. Let $\sigma^{-1} \in \{0,1\}^{2n}$ denote the assignment obtained by swapping 0's and 1's in σ. Then,*

$$\sigma \text{ is a fixed point of (4.4)} \iff \sigma^{-1} \text{ is a fixed point of (4.4)}\ .$$

Proof of Proposition 13. If σ is a fixed point of (4.4) and we set $\sigma(t) = \sigma$, we must have that $\sigma(t+1) = \sigma$. Let $i \in V$ and let us assume, without loss of generality, $\sigma_i = 1$. Since $\sigma_i(t+1) = 1$ (and ties do not occur in the fixed point) it must be the case that $f_i(\sigma) > \bar{f}(\sigma)$. Given that $f_i(\sigma^{-1}) = 1 - f_i(\sigma)$ and $\bar{f}(\sigma^{-1}) = 1 - \bar{f}(\sigma)$ (both properties can be derived from Definition 4.1 and 4.3, using that $\sigma_i^{-1} = 1 - \sigma_i$), we obtain that $f_i(\sigma^{-1}) < \bar{f}(\sigma^{-1})$. From the assumption $\sigma_i(t) = \sigma_i = 1$ we have that $\sigma_i^{-1}(t) = 0$, which together with $f_i(\sigma^{-1}) < \bar{f}(\sigma^{-1})$ assures that $\sigma_i^{-1}(t+1) = 0$, and the result follows. □

4.3 Complexity

The computational complexity of GAM can be determined as follows. The computation of $f(\sigma)$ can be performed visiting the neighbors of every node, and thus requires $\sum_{i\in V} d_i = 2m$, twice the number of edges in the graph. Computing $\bar{f}(\sigma)$ and $\sigma(t)$ requires $2n$ operations with the later updating the labels for every node, each of which in constant time (this may require the generation of a random number). Last, cycle detection can be performed in constant time with the usage of hash table, where the hash key is the label assignment $\sigma(t)$ and the hash value is k. This procedure repeats until a cycle is found. If η denotes the number of iterations until a cycle is found, then the complexity of GAM until stopping is $\Theta\left(\eta(m+n)\right)$.

Note that recovering the set S (set of fixed nodes) can be performed in time proportional to the cycle length. Moreover, since every node must be considered, the complexity of this operation is $\Theta(\eta n)$. Thus, the final computation complexity of GAM is $\Theta\left(\eta(n+m)\right)$ which is considered *almost linear* in the case η scales slowly with the size of the graph. Indeed, empirical evaluations indicate that η is rather small, depending mostly on the hardness of detecting communities.

4.4 Numerical evaluation

To evaluate the performance of the GAM algorithm, we performed numerical simulations using the planted bisection model PBM$(2n, p, q)$ as benchmark, and compared the results to the MVA and a vanilla spectral method. The algorithms were implemented in the Python language, using the packages `networkx` to facilitate network generation and manipulation, and `scipy` and `numpy` for numerical computations, including the eigenvector computation for the spectral method. All experiments were repeated multiple times independently (including generation of the graph instance), and using various parameter choices for the PBM. For every graph instance, the algorithms were executed a number of times, each one with a new random initial state. Results indicate the sample average and the 95% confidence interval shown as error bars in the plots, over all repetitions.

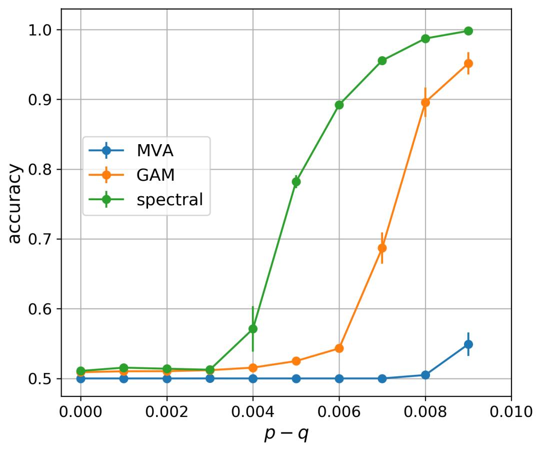
4.4.1 Accuracy

Figure 4.1 shows the average accuracy of MVA, GAM and spectral method as a function of $p-q$ for fixed $n = 1000$ and $p = 0.01, 0.02, 0.05$, over 10 graph instances and 30 repetitions each. As $p-q$ goes from 0 (where there is no planted community structure in the network) to $p-0.01$, we observe a phase transition as the algorithms perform no better than random guesses, to the point where almost all vertices are labelled correctly. This is in agreement with the idea that $p-q$ quantifies the strength of the planted community structure.

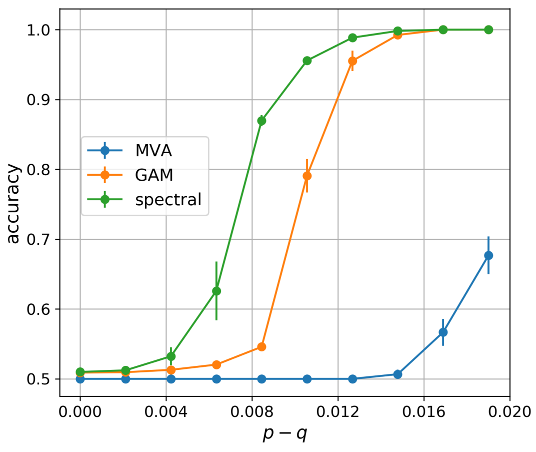
Also, the location of the phase transition differs between the algorithms, but keeping the relative order: first the spectral algorithm, then GAM, and MVA. Furthermore, as p increases, i.e the network becomes denser, the beginning of the phase transition starts closer to $p-q=0$.

4.4.2 Other metrics

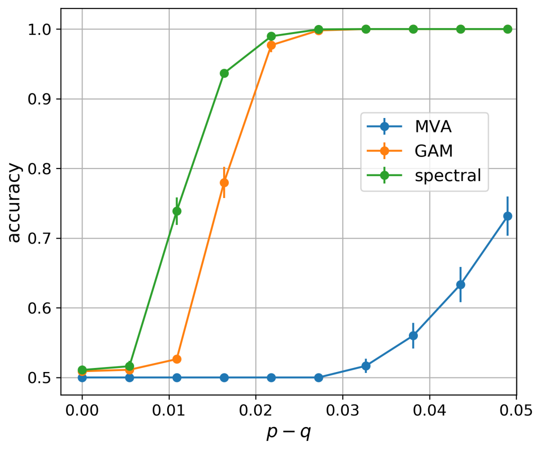
To quantify and analyze the behavior of MVA and GAM, a number of other behavior metrics were employed.



(a) $n = 1000$, $p = 0.01$

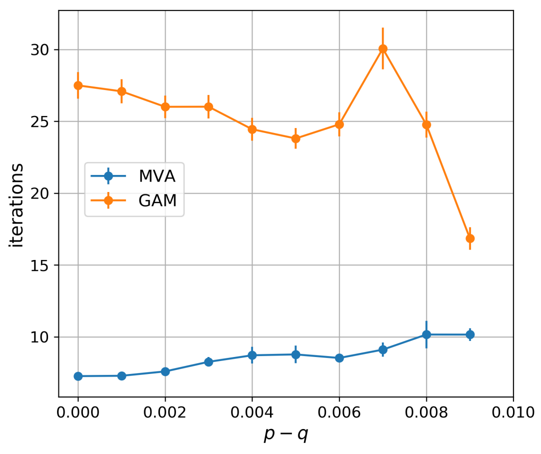


(b) $n = 1000$, $p = 0.02$

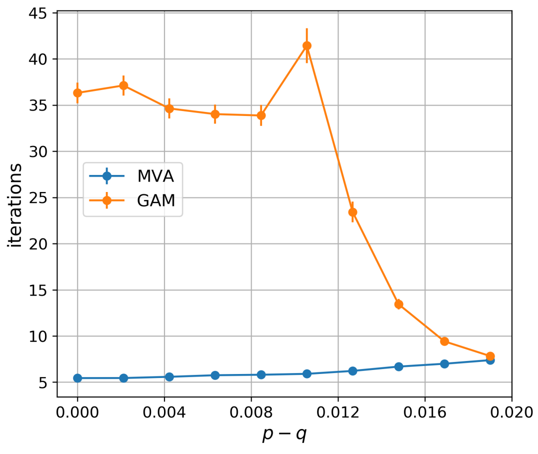


(c) $n = 1000$, $p = 0.05$

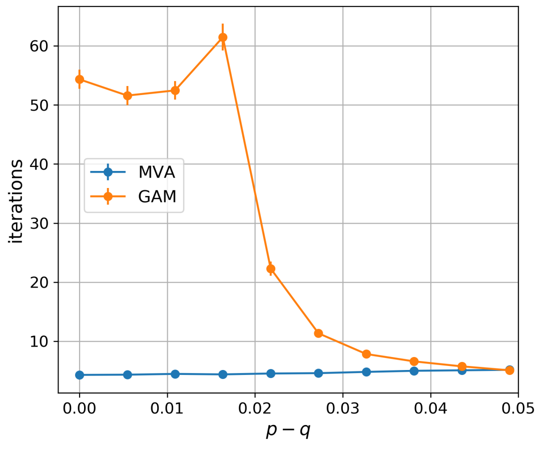
Figure 4.1: Accuracy of MVA and GAM for different values of $p - q$. Each value is computed as the average over $M = 10$ instances with $R = 30$ repetitions each.



(a) $n = 1000$, $p = 0.01$



(b) $n = 1000$, $p = 0.02$



(c) $n = 1000$, $p = 0.05$

Figure 4.2: Number of iterations for different values of $p-q$. Each value is computed as the average over $M = 10$ instances with $R = 30$ repetitions each.

Figure 4.2 shows the average number of iterations until each algorithm stops, with fixed $n = 1000$, $p = 0.01$, 0.02, 0.05, and varying $p-q$ between 0 and $p-0.001$. It can be seen that the number of iterations performed by MVA increases very slowly with $p - q$, with decreasing slope as p becomes larger. In sharp contrast, GAM displays a very interesting phase transition, where the number of iterations starts relatively high, decreasing slowly with $p - q$ until it peaks out, and then decreases rapidly to the same number of iterations as MVA. This suggests that GAM explores the label assignment space more effectively than MVA, converging faster to a solution when there is more structural evidence (larger $p - q$), and spending more time when it is scarce.

Another interesting figure is the cycle length, seen in the upper left of Figures 4.3, 4.4, and 4.5 for $p = 0.01$, 0.02, 0.05. MVA almost always stops by reaching a fixed point, that is, a cycle of length 1. On the other hand, GAM again shows a sort of phase transition, going from length 2 near $p - q = 0$ to 1 (fixed point) when

$p - q$ gets larger. As with the accuracy, the beginning of the transition approaches $p - q = 0$ with increasing p, reinforcing the idea that GAM recognizes that the problem gets easier as $p - q$ and p grow, while the behavior of MVA is less affected by these changes. The fraction of 1-labelled nodes on the upper right of Figures 4.3, 4.4, and 4.5 show that GAM is always finding perfectly balanced partitions, while MVA's behavior is more erratic.

Perhaps the most striking feature of GAM, which is also the motivation for the next chapter, the fraction and accuracy of fixed vertices, on bottom left and right of Figures 4.3, 4.4, and 4.5, respectively. Recall that both algorithms stop when they arrive at a previously found state, i.e a previously found community vector. While for MVA this usually happens at fixed point, i.e an exact repetition of the last state, in GAM the cycle length of 2 indicates the appearance of a different state before repeating. We say that nodes whose labels do not change during the full cycle are *fixed nodes* (or vertices). Naturally, the *accuracy of fixed vertices* is the agreement of community vector with the ground truth when the coordinates are restricted to the fixed vertices. Evidently, the fraction of fixed labels for MVA is essentially 1. GAM, on the other hand, displays a phase transition in which the fraction of fixed vertices goes from 0.5 to 1 as $p-q$ goes from 0 to $p-0.001$, with a similar one occurring with the accuracy. This behavior is very interesting because it shows that, as the problem instance gets easier, GAM is fixing more nodes in the cycle, and also the accuracy of the fixed nodes is increasing. Thus, an idea to improve this algorithm would be to re-initialize the iterations keeping the labels of fixed vertices, and randomizing the rest. This idea is explored in the next chapter.

Last, Figure 4.6 shows the dynamic threshold value $\bar{f}$ used in GAM (see Equation 4.3) across the iterations where each curve corresponds to a different network instance. One hundred instances were generated, and 8 trajectories were sampled at random (in color). The thick black line is the average over all trajectories. We observe that in all scenarios $\bar{f}$ changes across the iterations due to particularities of the graph instance and current label assignment. However, $\bar{f}$ eventually converges (when a cycle of length one is found), or oscillates between two values (when a cycle of length two is found). Interestingly, the average of $\bar{f}$ for each iteration across the graph instances (red line in the plot) is around 0.5, indicating that it indeed generalizes the 1/2 threshold of MVA by considering structural features of the graph instance at hand.

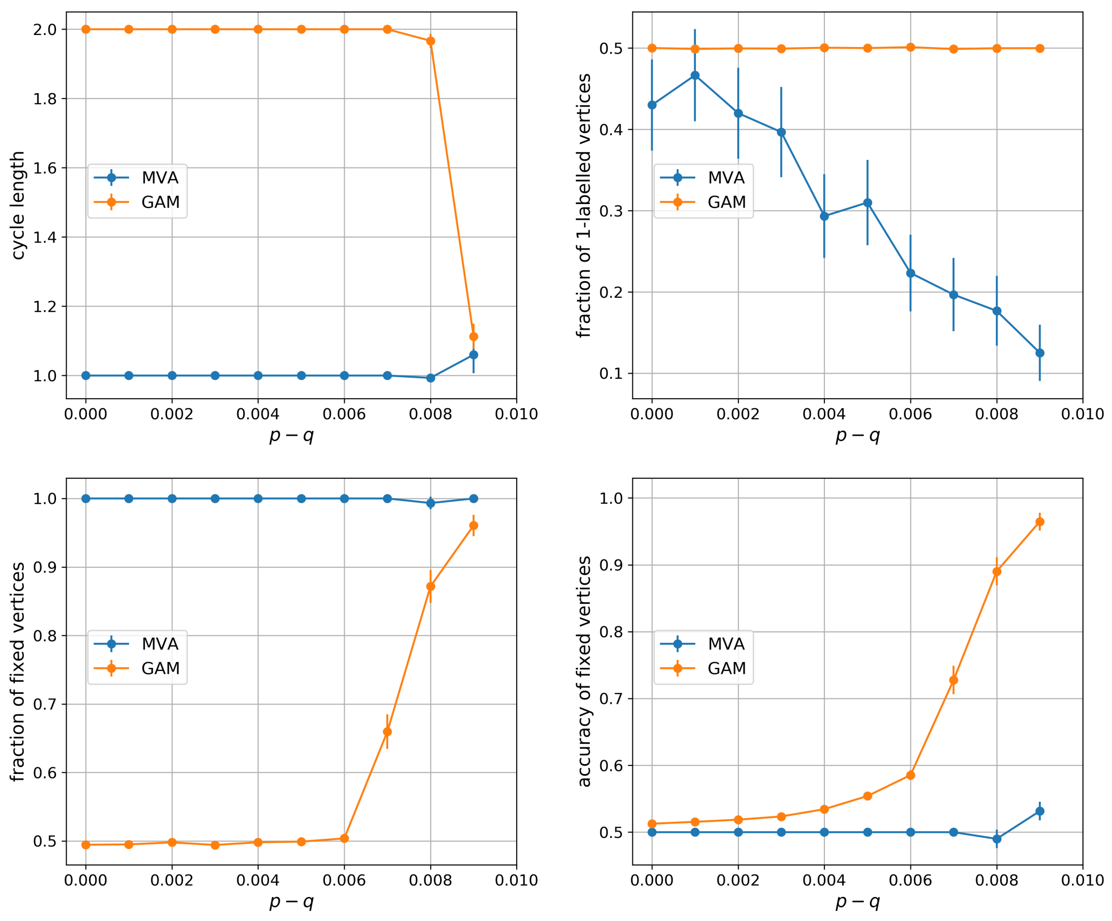


Figure 4.3: Additional metrics of MVA and GAM iterations. ($n = 1000$, $p = 0.01$)

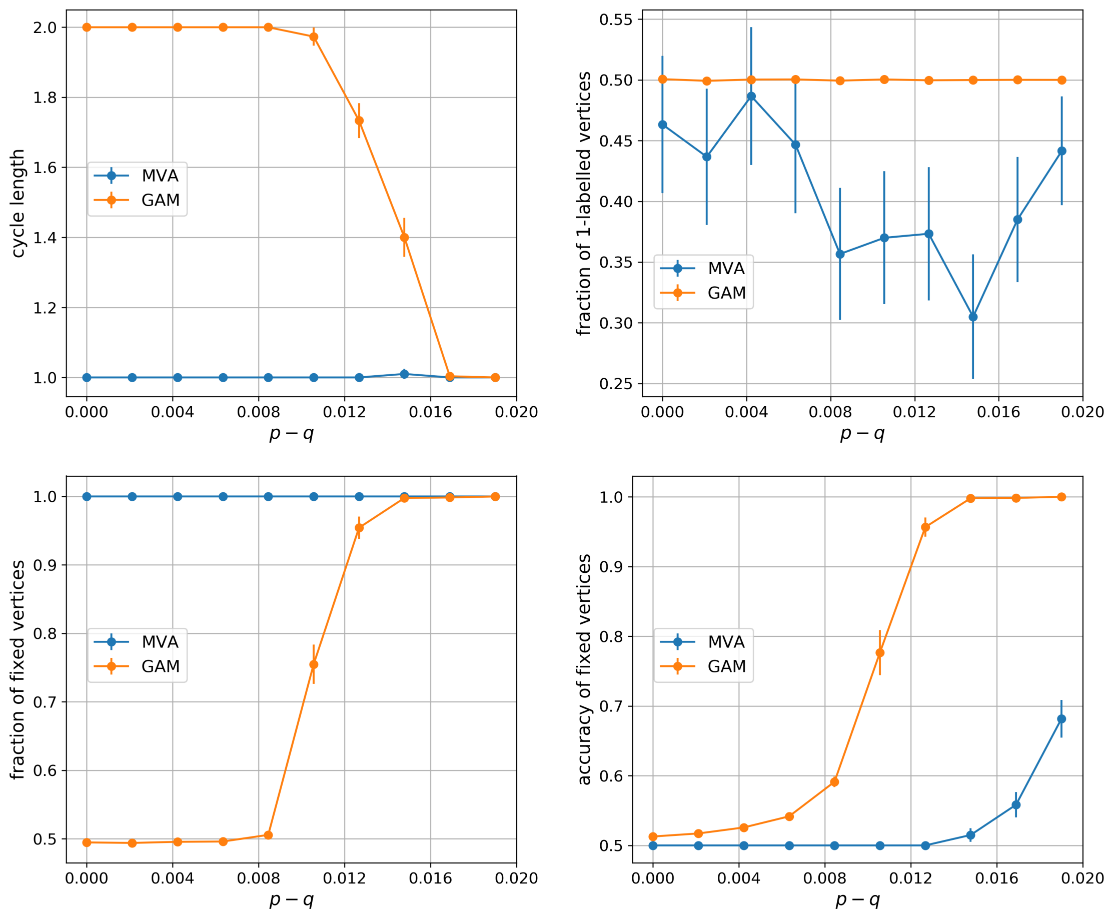


Figure 4.4: Additional metrics of MVA and GAM iterations. ($n = 1000$, $p = 0.02$)

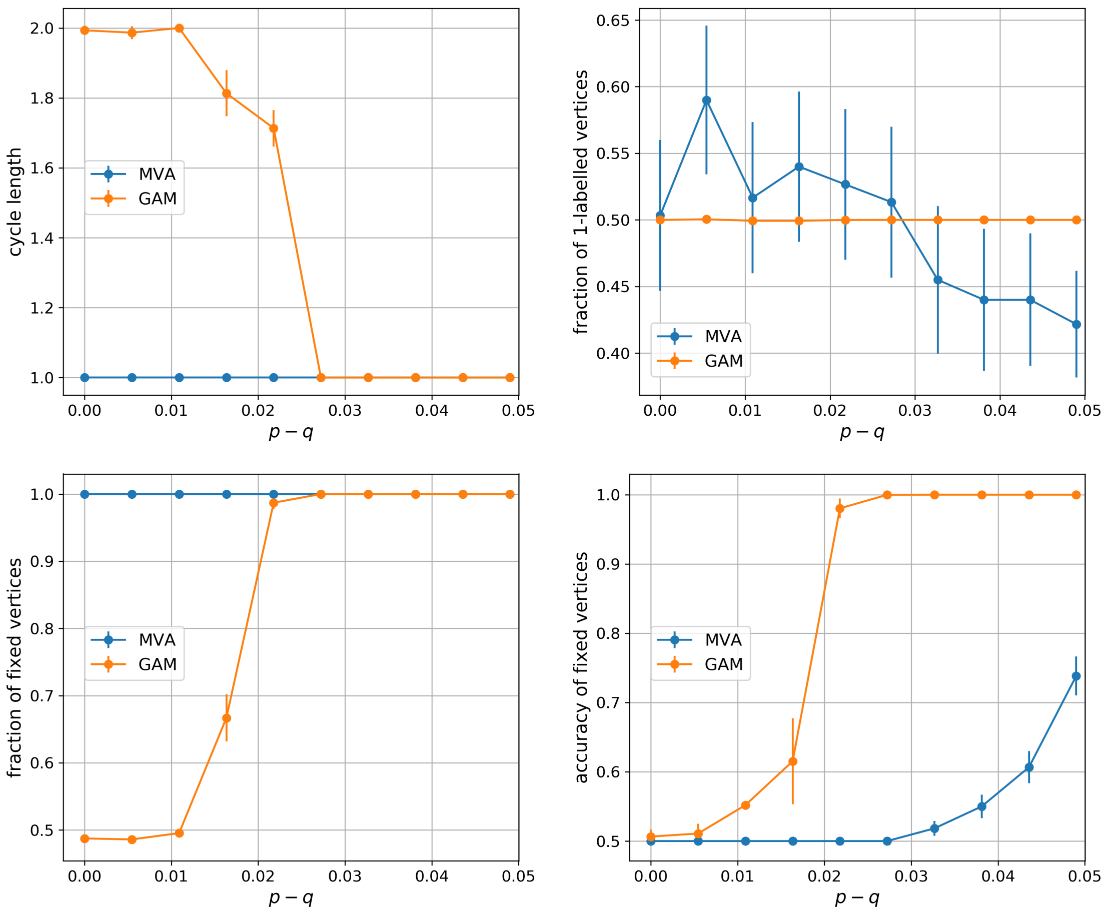


Figure 4.5: Additional metrics of MVA and GAM iterations. ($n = 1000$, $p = 0.05$)

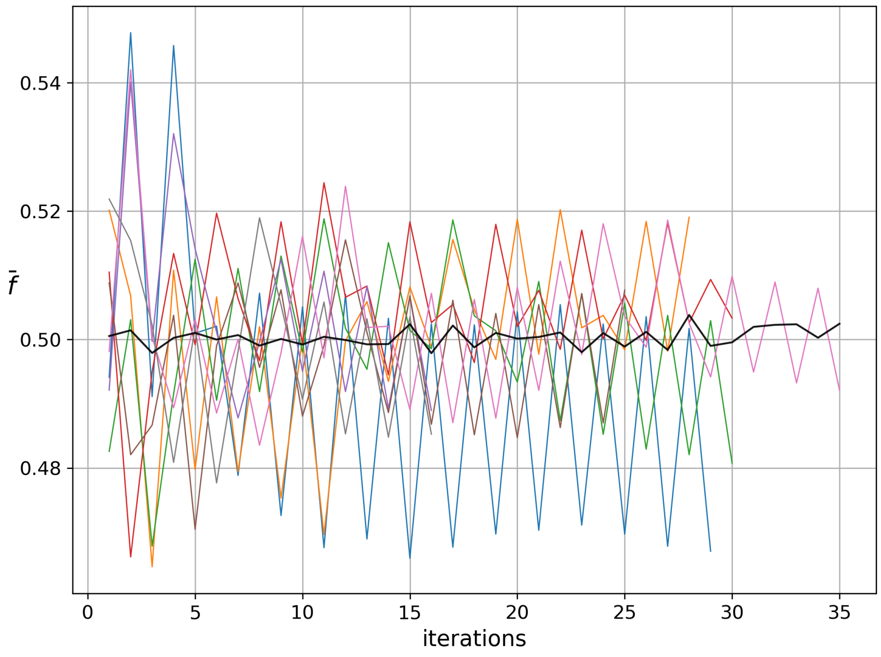


Figure 4.6: Threshold value $\bar{f}$ of GAM across its iterations. The trajectories in color were chosen at random from 100 graph instances, and the thick black curve shows the average over all instances. ($n = 1000$, $p = 0.01$, $p - q = 0.006$).

Chapter 5

Bootstrapping

Recall that GAM stops when it finds a cycle (or fixed point) after a sufficiently large number of iterations. Consider the sequence of label assignments in a given cycle found by the algorithm: $\sigma(1), \sigma(2), \ldots, \sigma(\eta) = \sigma(1)$ with $\eta \geq 2$ denoting the cycle length. Consider a node i that retains its label throughout the cycle, i.e., such that $\sigma_i(1) = \sigma_i(2) = \cdots = \sigma_i(\eta)$, and a node j that changes its label several times in the cycle. Intuitively, the label of node i is much more likely to be correct than the label of node j in configuration $\sigma(\eta)$. Indeed, retaining its label throughout the cycle is an indication that this label is robust to other assignments, and thus more likely to be correct.

This intuition can be confirmed by simulations which show that fixed nodes exhibit a much larger accuracy, specially in regimes where it is more difficult to recover the correct labels (to be shown). This observation will drive the next contribution.

Since fixed nodes can be easily identified in a cycle and their label assignment is more likely to be correct, this information can be used to re-initialize GAM with the expectation that it will find cycles with an even larger number of fixed nodes, and possibly more accurate assignment. We call this idea *bootstrapping*. Moreover, bootstrapping can be applied in successive rounds, pushing GAM towards producing cycles with ever more fixed nodes and more accurate label assignments.

One important consideration is how to exploit the information concerning fixed nodes to drive the initialization of the next round. Before proceeding, some instrumental concepts need to be introduced.

Consider running GAM for a sequence of rounds, $r = 1, 2, \ldots$ Before starting a round, node labels must be initialized. In round $r = 1$ labels are initialized uniformly at random, while in other rounds labels are initialized using the cycle found by the algorithm in the previous round.

For any given round, let η denote the first time the algorithm produces an assignment that has already been observed in this round, thus forming a cycle in the sample path, i.e.,

$$\eta := \min\{k > 0 : \exists\, l < k : \sigma(l) = \sigma(k)\} \ .$$

Note that $\eta < +\infty$ since GAM explores states in a finite state space. In fact, GAM generates a sample path in the corresponding finite state space Markov chain (where the chain states are label assignments and most transitions have probability 1). Moreover, for a given η there exists only one $\theta < \eta$ such that $\sigma(\theta) = \sigma(\eta)$, thus θ is the time of the first assignment that is identical to $\sigma(\eta)$.

For a given execution of GAM, consider the following definitions

- $\widehat{\sigma} := \sigma(\eta)$; the label assignment produced by GAM
- $\mathcal{C} := \{\sigma(l) : \theta \leq l \leq \eta\}$; the cycle encountered by GAM that terminated its execution
- $S := \{i \in V : \sigma_i(l) = \sigma_i(\eta) \text{ for all } \theta \leq l \leq \eta\}$; the set of nodes that maintain their labels in the cycle (fixed nodes)

Clearly, $\widehat{\sigma}$, η, θ, $\mathcal{C}$ and S depend on the round which will be emphasized using the superscript r. Furthermore, for a node $i \in S^r$, we denote

- $N_i^r := |\{j \in S^r : A_{ij} = 1\}|$; the number of neighbors of i that are also fixed nodes (in round r)
- $M_i^r := |\{j \in S^r : A_{ij} = 1 \text{ and } \widehat{\sigma}_j^r = \widehat{\sigma}_i^r\}|$; the number of nodes in N_i^r that have the same label as node i (in round r)

We propose two different bootstrapping procedures. The first determines the initial assignments of a round considering only if the node was a fixed node in the previous round. More precisely,

$$\sigma_i^{r+1}(0) = \begin{cases} \widehat{\sigma}_i^r \ , & \text{if } i \in S^r \ , \\ \sim \text{Ber}(1/2) \ , & \text{if } i \notin S^r \ . \end{cases} \tag{5.1}$$

This procedure is denoted by *hard* bootstrapping as it is more rigid, assigning fixed nodes to their labels in the previous round and randomizing non-fixed nodes. Intuitively, hard bootstrapping trusts the fixed nodes produced by GAM, since it maintains their labels in the next round, and randomizes the labels of the others.

The second bootstrapping procedure leverages not only information about the node itself (fixed or not fixed), but also about its neighbors. Moreover, it will also assign labels at random (but not necessarily uniform) to all nodes.

More specifically, a fixed node will retain its label in the next round with a certain probability, called *acceptance probability*, and it will swap its label with the complementary probability. The acceptance probability of a fixed node i is determined by the number of neighbors of i that are also fixed nodes. Thus, the larger the number of fixed neighbors that have the same label of node i, the higher the probability that the label of node i is correct. More precisely,

$$\begin{aligned} &\sigma_i^{r+1}(0) \sim \mathrm{Ber}(1/2)\ , && i \notin S^r \\ &\sigma_i^{r+1}(0) = \begin{cases} \widehat{\sigma}_i^r\ , & \text{with prob. } \rho_i^r \\ 1-\widehat{\sigma}_i^r\ , & \text{with prob. } 1-\rho_i^r \end{cases}\ , && i \in S^r \end{aligned} \tag{5.2}$$

where ρ_i^r is the acceptance probability for i obtained from the output of round r and used in the initialization of round $r+1$, defined as

$$\rho_i^r := \frac{1}{2} + \frac{1}{2}\frac{M_i^r}{N_i^r}\ .$$

In contrast, we denote this procedure by *soft* bootstrapping, as it allows for a fixed node to have a different label in a subsequent round. Intuitively, soft bootstrapping trusts less the immediate output of GAM and uses more information (about the nodes neighbors) to regain trust.

In particular, whenever a fixed node i has the same label as all of its fixed neighbors (i.e., $M_i^r = N_i^r$), the acceptance probability $\rho_i^r = 1$ and consequently $\sigma_i^{r+1}(0) = \widehat{\sigma}_i^r$ with probability one. Under this condition, fixed nodes retain their labels in the next round. On the other hand, if none of the neighbors of i are fixed, then $\rho_i^r = 1/2$ and $\sigma_i^{r+1}(0) = \widehat{\sigma}_i^r$ with probability 1/2. Thus, the fact that node i is fixed is not enough evidence for soft bootstrapping, if node i has no fixed neighbors with the same label of itself.

The algorithm enhancing GAM with bootstrapping rounds is called GAM (Global Average Majority with Bootstrapping) and is shown in Algorithm 3. Note that GAM is initialized with a uniform random label assignment $\sigma^{r=0}(0) \sim \mathrm{Ber}(1/2)$ and stops after a maximum number of rounds is reached, denoted by R. Last, the bootstrapping procedures described here is not dependent on GAM and can be applied to any other MVA, as long as the algorithm produces a cycle with its output.

5.1 Complexity

The computational complexity of GAMB is proportional to the number of rounds R. For every round, GAM is executed and σ is updated, with the later having complexity $\Theta(n)$, as all nodes must be updated. Thus, if η^* is the maximum number of iterations of GAM across all R rounds, then GAMB has complexity $\Theta(R\eta^*(n+m))$. Again, this can be considered almost linear time if R does not grow with the graph size. Indeed, as numerical experiments will soon indicate, R can be relatively small for GAMB to produce more accurate results.

Algorithm 3: *Global Average Majority with Bootstrapping (GAMB)*

```
Input:  Graph $G = (V, E)$, initial nodes label $\sigma \in \{0,1\}^V$
        number of bootstrap rounds R
Output: Estimated community labels $\widehat{\sigma}$

/* hard bootstrapping                                                */
1  for $r = 1, \dots, R$ do
2      $(\widehat{\sigma}^r, S) = \texttt{GAM}(G, \sigma)$
3      for $i \in V$ do
4          if $i \in S$ then
5              $\sigma_i = \widehat{\sigma}_i^r$
6          else
7              $\sigma_i \sim \mathrm{Ber}(1/2)$
8          end
9      end
10 end
11 return $\widehat{\sigma}^R$

/* soft bootstrapping                                                */
1  for $r = 1, \dots, R$ do
2      $(\widehat{\sigma}^r, S) = \texttt{GAM}(G, \sigma)$
3      for $i \in V$ do
4          $N_i = |\{j \in S : A_{ij} = 1\}|$
5          $M_i = |\{j \in S : A_{ij} = 1 \text{ and } \widehat{\sigma}_j^r = \widehat{\sigma}_i^r\}|$
6          if $i \in S$ then
7              $\sigma_i = \begin{cases} \widehat{\sigma}_i^r, & \text{with prob. } \frac{1}{2} + \frac{1}{2}\frac{M_i}{N_i} \\ 1 - \widehat{\sigma}_i^r, & \text{with prob. } \frac{1}{2} - \frac{1}{2}\frac{M_i}{N_i} \end{cases}$
8          else
9              $\sigma_i \sim \mathrm{Ber}(1/2)$
10         end
11     end
12 end
13 return $\widehat{\sigma}^R$
```

5.2 Numerical evaluation

Figure 5.1 includes hard and soft GAMB in the same comparison of accuracy as seen in Figure 4.1). We notice that both bootstrapping strategies display the fast transition to full accuracy, but with an improved accuracy over MVA and GAM.

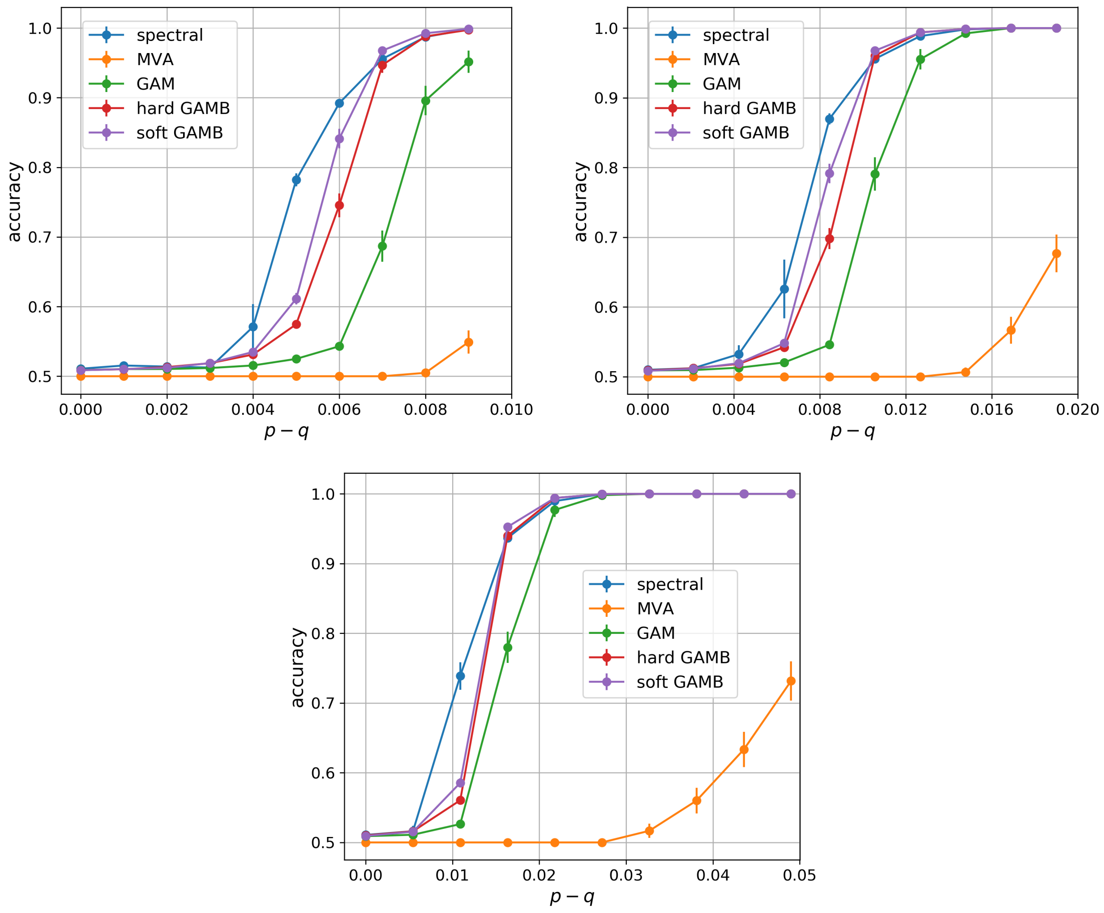
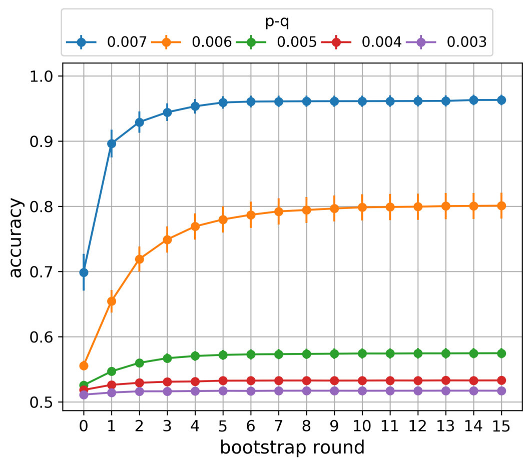


Figure 5.1: Bootstrapping accuracy with $n = 1000$, $p = 0.01, 0.02, 0.05$

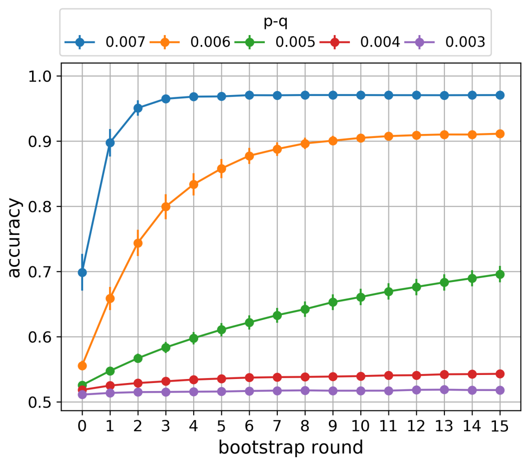
Their performance is quite comparable to that of the spectral method, and even slightly superior for some points in the transition, clearly indicating its superiority over MVA for the accuracy, and spectral method for running time.

Figure 5.2 shows the accuracy of the two strategies as a function of bootstrap rounds, with various values of $p - q$. Each curve shows the average over 10 graph instances with 20 repetitions each. We can clearly see the increase in accuracy with the bootstrap rounds. Note that round 0 corresponds to the accuracy of the initial GAM, where the labels are initialized at random. For $p-q = 0.007$, we remark that with the first bootstrap round, in both hard and soft bootstrapping, the accuracy goes from 0.7 to 0.9, an increase of 28%. With more iterations, accuracy stays near 0.97, corresponding to an accumulated increase of around 38%. The curve corresponding to $p - q = 0.006$ shows a similar behavior, even more pronounced in soft bootstrapping, going from 0.56, almost as bad as random guessing, to 0.9 after 9 rounds, an impressive increase of 60% in accuracy. Lower values of $p - q$ show a more subtle increase, if at all.

In Figure 5.3, we see the evolution of accuracy and fraction of fixed vertices for hard (left column) and soft (right) bootstrapping. We observe a similar increase in



(a) Hard bootstrapping

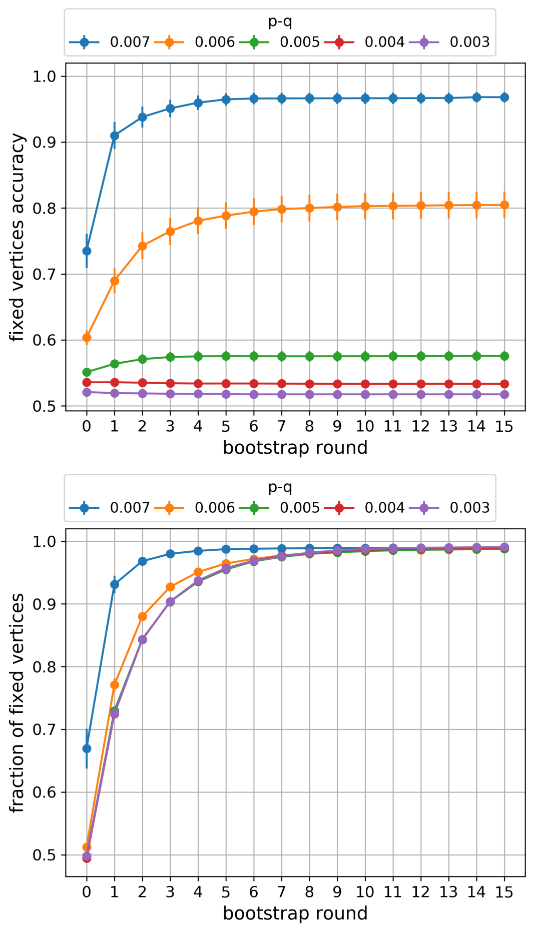


(b) Soft bootstrapping

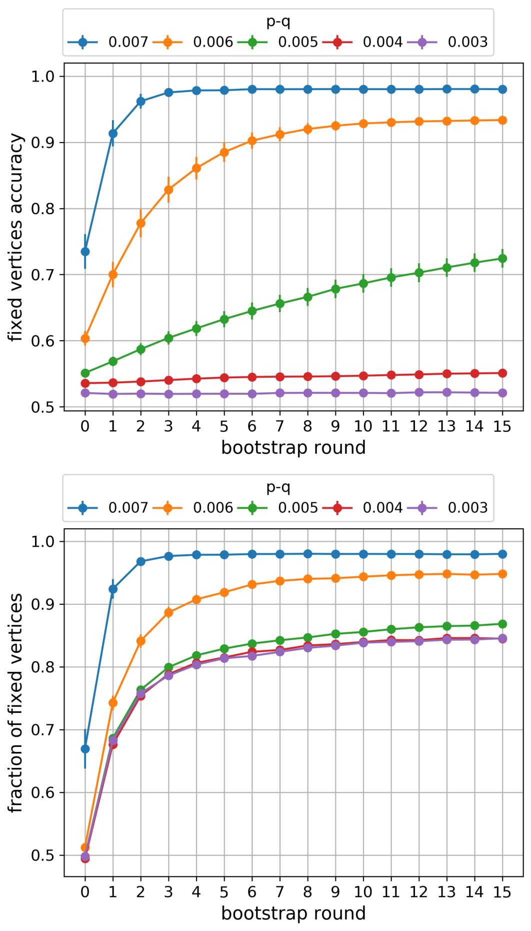
Figure 5.2: Accuracy as a function of the bootstrap rounds ($n = 1000$, $p = 0.01$).

accuracy as the ones in Figure 5.2, and also rapidly growing fraction of fixed vertices, converging to 1 in hard bootstrapping, and generally increasing in soft bootstrapping. This is to be expected from their definitions, since hard bootstrapping keeps all of the fixed vertices' labels, while soft bootstrapping allow for some randomization of fixed vertices' labels. These figures confirm the phenomenom hinted at at the end of the last chapter, that the fixed nodes in a GAM cycle are better classified than the rest. Also, comparing hard and soft bootstrapping, we notice that soft bootstrapping performs better in terms of accuracy, even achieving a pronounced increase in accuracy for $p - q = 0.005$ whereas hard bootstrapping does not, thus suggesting that the more flexible approach, which does not fully "trust" the label of every fixed vertex and uses fixed neighbors to perform a biased randomization, is superior.

Figure 5.4 shows the average number of iterations performed by the GAM at each round of hard and soft bootstrapping. For hard bootstrapping, the number of iterations decreases monotonically with the rounds and does not depend strongly on the $p - q$ gap (with the larger gap requiring slightly less iterations). The picture is quite different for soft bootstrapping where for smaller gaps, the number of iterations stabilizes at a higher value which also seems to be dependent on $p - q$. In general, soft bootstrapping requires more iterations than hard at any given round, justified by the difference in initialization. Last, all cycles found in all the rounds shown either by hard or soft bootstrapping had length at most two. Thus, cycles are very short.



(a) Hard bootstrapping

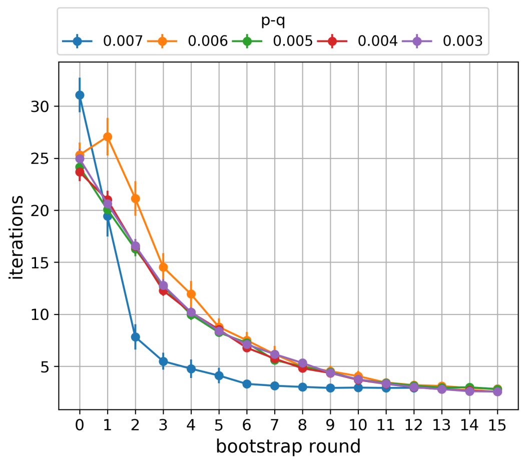


(b) Soft bootstrapping

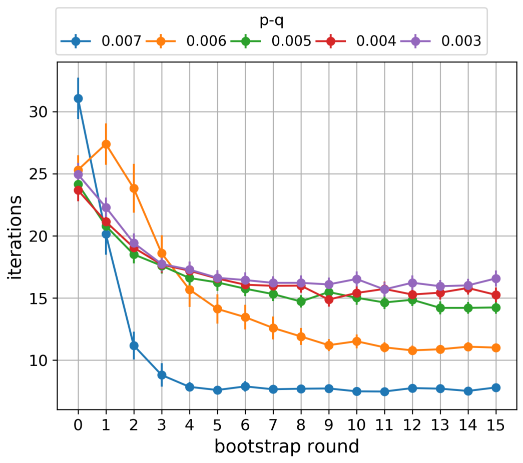
Figure 5.3: Accuracy and fraction of fixed vertices by bootstrap rounds ($n = 1000$, $p = 0.01$).

5.3 Bootstrapping MVA

In Figure 5.5, we see the effect of trying to apply the bootstapping strategy to MVA iterations. It is clear that there is no improvement on the accuracy. This is easily justifiable, since hard bootstrapping does not break out of fixed points (which are the usual configuration produced by MVA), and soft bootstrapping cannot take advantage of the biased sampling, once every vertex is a fixed. Thus, the gains achieved with the bootstrapping strategy are closely tied to the behavior of the states produced by GAM iterations, in which fixed vertices are more likely to be correctly classified.

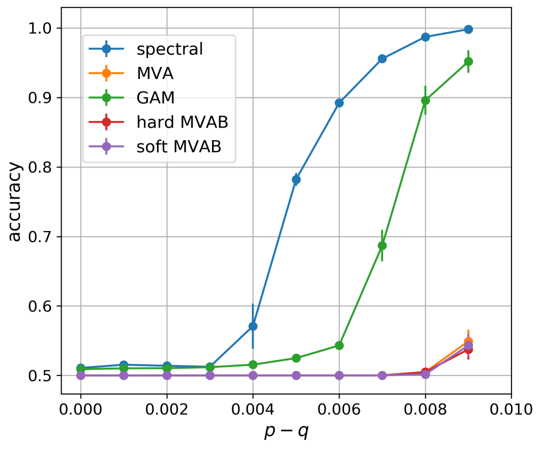


(a) Hard bootstrapping



(b) Soft bootstrapping

Figure 5.4: Number of iterations by bootstrap rounds ($n = 1000$, $p = 0.01$).



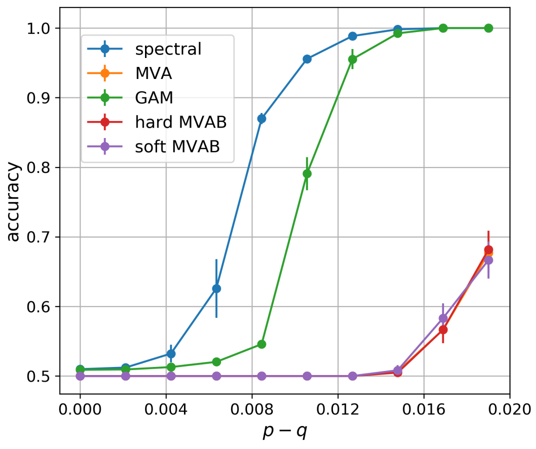


Figure 5.5: Accuracy of bootstrapped MVA

Chapter 6

Evaluation on real networks

In this chapter we evaluate the previously discussed methods when applied to some real world networks for which a meaninful ground truth with two communities is known. As said previously, real networks often have no clear-cut community ground truth, and vary greatly in number and composition of communities. Thus, the application of our methods without further adaptation is somewhat limited. For every network, the algorithms were executed 100 times, and the accuracy with respect to the ground truth partition, and the wall clock execution time were recorded. Summary information about each one can be found in Table 6.1. All of the datasets used can be found on Mark Newman's Network Data repository [60].

Network	# nodes	# edges	source
karate	34	78	[61]
polbooks*	92	374	-
polblogs*	1222	16714	[2]

*after pre-processing

Table 6.1: Summary information of benchmark real world networks

6.1 Karate club

One of the first and perhaps the most popular real world network with known ground truth communities is the karate club network studied by ZACHARY [61]. It has 34 vertices corresponding to the members of a karate club in the United States, who were observed for 3 years. The edges in the graph link the indiduals that participated in activities outside the karate club. When a conflict happened between the president of the club (node 34 in Figure 6.1) and the instructor (vertex 1), the members of the club split up into two groups, whose members supported either the instructor (shown in red), or the club president (in red). In Figure 6.1,

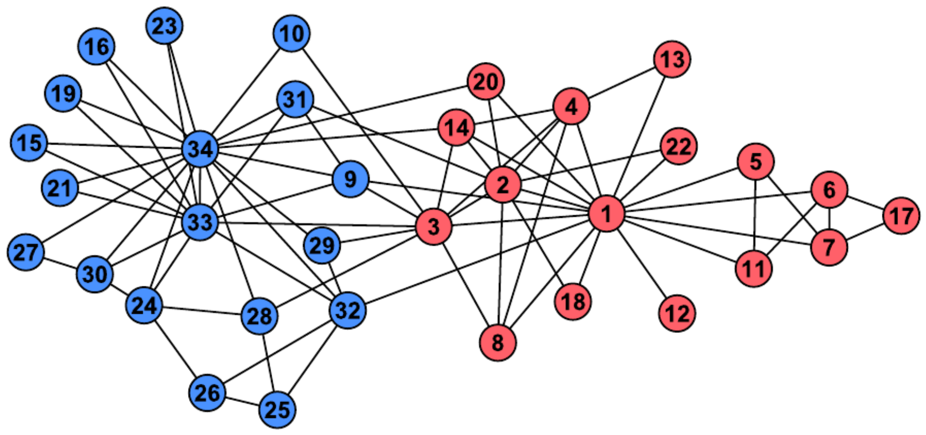


Figure 6.1: Zachary's karate club network. Source: [1]

we can observe that this division of the group is well reflected in the topology of the network: almost every node in the graph is connected to either one of nodes 1 or 34, and these connections are strongly correlated to the group in which the club member stayed after the division.

To assess the performance of the algorithms, we took as ground truth the partition of the members after the split up of the club, used as a reference to compute the accuracy of the estimated community vectors for each of the algorithms. The results are displayed in Table 6.2. We can see that the algorithms' average accuracy is ordered as seen before: from lowest to highest, MVA, GAM, hard GAMB, soft GAMB, Spectral. It is interesting to note that the maximum value of the accuracy for each algorithm coincides with the accuracy of the spectral method, and the minimum value attains the minimum accuracy possible, 0.5. In terms of wall clock time, the spectral method is clearly more advantageous, since only MVA and GAM are faster, but offer less accurate solutions.

Method	Accuracy				Time (s)			
	min	max	avg.	std.	min	max	avg.	std.
Spectral	0.97	0.97	0.97	0.00	0.0006	0.0741	0.0014	0.0074
MVA	0.50	0.97	0.63	0.16	0.0005	0.0016	0.0008	0.0002
GAM	0.50	0.97	0.70	0.16	0.0006	0.0009	0.0007	0.0001
hard GAMB	0.50	0.94	0.84	0.14	0.0035	0.0045	0.0039	0.0002
soft GAMB	0.50	0.97	0.87	0.14	0.0037	0.0056	0.0051	0.0003

Table 6.2: Karate club network

6.2 Political books

The network of political books was compiled by Valdis Krebs from a list of best-selling books and recommendations from the Amazon bookseller website that are based on co-purchasings [62]. In other words, the nodes of this network consist of 105 book titles, and an edge links two books if one has been suggested for a client viewing the other, which happens with higher probability if many previous clients bought both books. The books were all related to north-american politics, and top sellers in a period close to the 2004 presidential elections. They are labelled in the network according to partisanship as liberal, conservative, or neutral.

Since there are 3 possible ground truth values for each vertex, to apply the algorithms we took the subgraph induced by the nodes corresponding to liberal and conservative books, shown in Figure 6.2. This projected network contains 92 vertices and 374 edges.

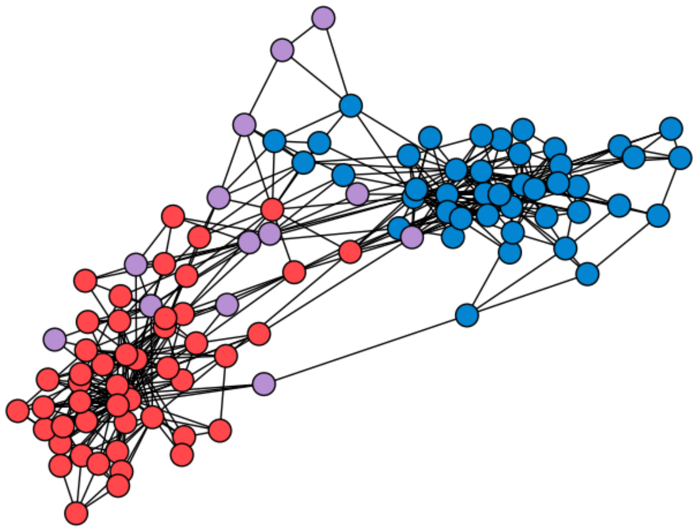
In Table 6.3, we can see the results of the algorithms applied to the induced network mentioned above. It is noticeable that the accuracy obtained by GAM with and without bootstrapping is much closer to that of the spectral method, and the latter is even outperformed by soft GAMB. Again, the wall clock execution times are very low because of the size of the network, and the spectral method is still a good option for its high accuracy and low execution time. This time, however, GAM is also competitive, displaying results almost as good as the spectral method, and taking half the time on average.

Method	Accuracy				Time (s)			
	min	max	avg.	std.	min	max	avg.	std.
Spectral	0.97	0.97	0.97	0.00	0.0032	0.0038	0.0032	0.0001
MVA	0.50	0.98	0.77	0.21	0.0012	0.0022	0.0015	0.0002
GAM	0.62	0.98	0.97	0.04	0.0013	0.0023	0.0016	0.0002
hard GAMB	0.92	0.98	0.97	0.01	0.0074	0.0091	0.0081	0.0003
soft GAMB	0.96	0.98	0.98	0.01	0.0112	0.0128	0.0117	0.0002

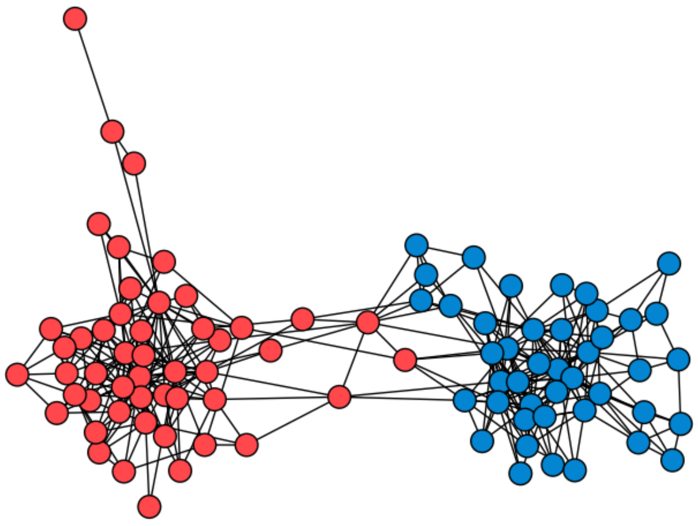
Table 6.3: Political books network

6.3 Political blogs

Last, with the same theme of the last network, the political blogs network was compiled from links between blogs related to US politics around the 2004 presidential elections, by ADAMIC and GLANCE [2]. Each vertex corresponds to a blog labelled as liberal or conservative, based on classifications made by political weblog lists and on the blog's contents. A visualization of this network is displayed on Figure 6.3, in



(a) Original network



(b) Projected network on conservative and liberal books

Figure 6.2: Political books network

which red nodes and edges correspond to conservative blogs, and the color blue to liberals. Orange edges point from liberal to conservative, and purple edges in the opposite direction. Also, the size of each node corresponds to the number of other blogs linking to it.

The original network consists of 1490 vertices and 19090 directed edges, and also contains multiple edges linking two vertices, and self-loops. In order to apply our algorithms, we first pre-process this network, removing directionality, self-loops, and repeated edges. As the resulting graph is not connected, we take the largest connected component, which has 1222 vertices and 16714 edges.

In Table 6.4 we see the results of the algorithms applied to this network. It is in this case that GAM and its bootstrapped versions really shine, attaining accuracies near 0.95, a higher value than the 0.93 accuracy obtained by the spectral method. In addition, the execution times of all of these are considerably lower than the time taken by the spectral method, which is to be expected from the complexity of the algorithms and the larger size of the network.

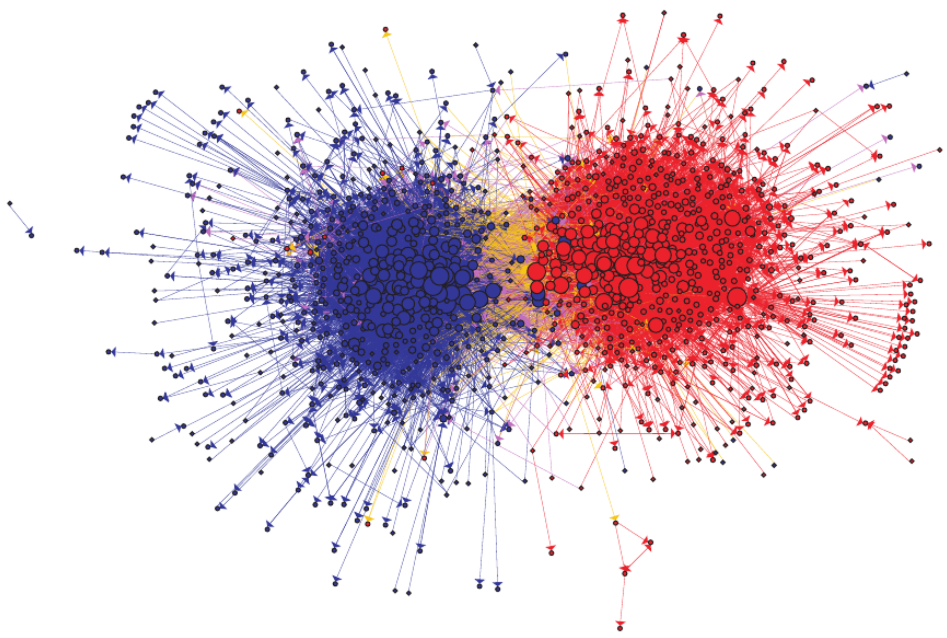


Figure 6.3: Political blogs network. Source: [2]

Method	Accuracy				Time (s)			
	min	max	avg.	std.	min	max	avg.	std.
Spectral	0.93	0.93	0.93	0.00	2.5872	2.5872	2.5872	0.00
MVA	0.52	0.95	0.66	0.20	0.0785	0.2889	0.1436	0.0779
GAM	0.95	0.96	0.95	0.00	0.0779	0.1233	0.0924	0.0127
hard GAMB	0.95	0.96	0.95	0.00	0.4938	0.5220	0.5042	0.0051
soft GAMB	0.95	0.96	0.95	0.00	0.5983	0.6354	0.6083	0.0055

Table 6.4: Political blogs network

Chapter 7

Conclusion

As we have seen, community detection is a fundamental problem in the area of network science, involving also insights from mathematics, computer science, statistics, and even physics. There are many avenues of investigation, some focusing on theoretical properties of models and algorithms, and other more practical, attempting to understand community structures present in real world networks and applying this knowledge to other tasks.

In this work we have focused on the problem of recovering two planted communities in a network, and proposed new algorithms based on the popular label propagation method. In particular, Global Average Majority or GAM introduces a dynamic threshold in substitution of the fixed threshold used in simple majority voting, and we have shown by theoretical analysis and numerical experiments that this modification does not incur on substantial penalties in terms of complexity, and achieves much higher accuracy nearing the spectral method used as reference, which on the other hand displays worse asymptotic complexity. This method also introduces a new stopping criterion that deals with the cycling problem displayed by solutions produced by label propagation.

Furthermore, by investigating other performance metrics of the GAM algorithm such as fraction and accuracy of fixed vertices, we proposed two bootstrapping strategies that re-utilize the solutions found by successive GAM rounds in order to amplify their accuracy, giving rise to the methods which we called hard and soft GAMB (Global Average Majority with Bootstrapping). Again, using synthetic networks generated by the planted bisection model and real world networks with known communities, we performed numerous experiments which allowed us to better understand the behavior and confirm the perfomance of the methods proposed.

7.1 Future Directions

Many development directions are left open for this work. We highlight some of the most interesting below:

- The dynamic threshold, stopping criterion based on cycles, and bootstrapping of solutions are general ideas that might be explored in other algorithms such as general label propagation (which allows the detection of an arbitrary number of communities) and belief propagation. Given the popularity of these algorithms and the quality of results obtained here, this seems to be a very promising research direction in practical community detection.

- Another related approach extending this work in the context of practical community detection would be the direct adaptation of GAM and GAMB to the problem of detecting an arbitrary number of communities. With such an algorithm, one could repeat the experiments we performed using as benchmarks the general stochastic block model and real world networks with multiple ground truth communities.

- On the theoretical side, it would be especially interesting to demonstrate more properties of the dynamic threshold used by GAM, and to obtain results further explaining the superiority of GAM over MVA, perhaps even under other probabilistic network models.

Bibliography

[1] FORTUNATO, S., HRIC, D. "Community detection in networks: A user guide", *Physics Reports*, v. 659, pp. 1–44, nov. 2016.

[2] ADAMIC, L. A., GLANCE, N. "The political blogosphere and the 2004 US election: divided they blog". In: *Proceedings of the 3rd international workshop on Link discovery*, pp. 36–43. ACM, 2005.

[3] NEWMAN, M. E. *Networks: An Introduction*. Oxford University Press, Inc., 2010.

[4] BARABÁSI, A.-L., GULBAHCE, N., LOSCALZO, J. "Network medicine: a network-based approach to human disease", *Nature reviews genetics*, v. 12, n. 1, pp. 56, 2011.

[5] LINDEN, G., SMITH, B., YORK, J. "Amazon. com recommendations: Item-to-item collaborative filtering", *IEEE Internet computing*, , n. 1, pp. 76–80, 2003.

[6] SØRLIE, T., PEROU, C. M., TIBSHIRANI, R., et al. "Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications", *Proceedings of the National Academy of Sciences*, v. 98, n. 19, pp. 10869–10874, 2001.

[7] SHI, J., MALIK, J. "Normalized Cuts and Image Segmentation", *IEEE Transactions on pattern analysis and machine intelligence*, v. 22, n. 8, 2000.

[8] CLAUSET, A., NEWMAN, M. E., MOORE, C. "Finding community structure in very large networks", *Physical review E*, v. 70, n. 6, pp. 066111, 2004.

[9] RAGHAVAN, U. N., ALBERT, R., KUMARA, S. "Near linear time algorithm to detect community structures in large-scale networks", *Physical review E*, v. 76, n. 3, pp. 036106, 2007.

[10] RADICCHI, F., CASTELLANO, C., CECCONI, F., et al. "Defining and identifying communities in networks", v. 101, n. 9, pp. 2658–2663, 2004.

[11] REICHARDT, J., BORNHOLDT, S. "Statistical mechanics of community detection", *Physical Review E*, v. 74, n. 1, pp. 016110, 2006.

[12] HU, Y., CHEN, H., ZHANG, P., et al. "Comparative definition of community and corresponding identifying algorithm", *Physical Review E*, v. 78, n. 2, pp. 026121, 2008.

[13] COSCIA, M., GIANNOTTI, F., PEDRESCHI, D. "A classification for community discovery methods in complex networks", *Statistical Analysis and Data Mining: The ASA Data Science Journal*, v. 4, n. 5, pp. 512–546, 2011.

[14] FORTUNATO, S. "Community detection in graphs", *Physics reports*, v. 486, n. 3-5, pp. 75–174, 2010.

[15] NEWMAN, M. E. "Mixing patterns in networks", *Physical Review E*, v. 67, n. 2, pp. 026126, 2003.

[16] NEWMAN, M. E., GIRVAN, M. "Finding and evaluating community structure in networks", *Physical review E*, v. 69, n. 2, pp. 026113, 2004.

[17] HOLLAND, P. W., LASKEY, K. B., LEINHARDT, S. "Stochastic blockmodels: First steps", *Social networks*, v. 5, n. 2, pp. 109–137, 1983.

[18] SÖDERBERG, B. "General formalism for inhomogeneous random graphs", *Physical review E*, v. 66, n. 6, pp. 066121, 2002.

[19] JERRUM, M., SORKIN, G. B. "The Metropolis algorithm for graph bisection", *Discrete Applied Mathematics*, v. 82, n. 1-3, pp. 155–175, 1998.

[20] CONDON, A., KARP, R. M. "Algorithms for graph partitioning on the planted partition model", *Random Structures & Algorithms*, v. 18, n. 2, pp. 116–140, 2001.

[21] YANG, J., LESKOVEC, J. "Defining and evaluating network communities based on ground-truth", *Knowledge and Information Systems*, v. 42, n. 1, pp. 181–213, 2015.

[22] LESKOVEC, J., LANG, K. J., MAHONEY, M. "Empirical comparison of algorithms for network community detection". In: *Proceedings of the 19th international conference on World wide web*, pp. 631–640. ACM, 2010.

[23] BLONDEL, V. D., GUILLAUME, J.-L., LAMBIOTTE, R., et al. "Fast unfolding of communities in large networks", *Journal of statistical mechanics: theory and experiment*, v. 2008, n. 10, pp. P10008, 2008.

[24] NEWMAN, M. E. “Finding community structure in networks using the eigenvectors of matrices”, *Physical review E*, v. 74, n. 3, pp. 036104, 2006.

[25] GILBERT, E. N. “Random graphs”, *The Annals of Mathematical Statistics*, v. 30, n. 4, pp. 1141–1144, 1959.

[26] ABBE, E. “Community Detection and Stochastic Block Models: Recent Developments”, *Journal of Machine Learning Research*, v. 18, pp. 1–86, 2018.

[27] MOSSEL, E., NEEMAN, J., SLY, A., et al. “Consistency thresholds for the planted bisection model”, *Electronic Journal of Probability*, v. 21, 2016.

[28] ABBE, E., BANDEIRA, A. S., HALL, G. “Exact recovery in the stochastic block model”, *IEEE Transactions on Information Theory*, v. 62, n. 1, pp. 471–487, 2016.

[29] KARP, R. M. “Reducibility among combinatorial problems”. In: *Complexity of computer computations*, Springer, pp. 85–103, 1972.

[30] HORN, R. A., HORN, R. A., JOHNSON, C. R. *Matrix analysis*. Cambridge university press, 1990.

[31] DECELLE, A., KRZAKALA, F., MOORE, C., et al. “Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications”, *Physical Review E*, v. 84, n. 6, pp. 066106, 2011.

[32] MASSOULIÉ, L. “Community detection thresholds and the weak Ramanujan property”. In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 694–703. ACM, 2014.

[33] MOSSEL, E., NEEMAN, J., SLY, A. “A proof of the block model threshold conjecture”, *Combinatorica*, v. 38, n. 3, pp. 665–708, 2018.

[34] MOSSEL, E., NEEMAN, J., SLY, A. “Reconstruction and estimation in the planted partition model”, *Probability Theory and Related Fields*, v. 162, n. 3-4, pp. 431–461, 2015.

[35] DESHPANDE, Y., ABBE, E., MONTANARI, A. “Asymptotic mutual information for the balanced binary stochastic block model”, *Information and Inference: A Journal of the IMA*, v. 6, n. 2, pp. 125–170, 2016.

[36] LESKOVEC, J., LANG, K. J., DASGUPTA, A., et al. “Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters”, *Internet Mathematics*, v. 6, n. 1, pp. 29–123, 2009.

[37] SALES-PARDO, M., GUIMERA, R., MOREIRA, A. A., et al. "Extracting the hierarchical organization of complex systems", *Proceedings of the National Academy of Sciences*, v. 104, n. 39, pp. 15224–15229, 2007.

[38] CLAUSET, A., MOORE, C., NEWMAN, M. E. "Hierarchical structure and the prediction of missing links in networks", *Nature*, v. 453, n. 7191, pp. 98, 2008.

[39] YANG, Z., ALGESHEIMER, R., TESSONE, C. J. "A comparative analysis of community detection algorithms on artificial networks", *Scientific reports*, v. 6, pp. 30750, 2016.

[40] MA, J., WANG, J., GHORAIE, L. S., et al. "A Comparative Study of Cluster Detection Algorithms in Protein–Protein Interaction for Drug Target Discovery and Drug Repurposing", *Frontiers in pharmacology*, v. 10, 2019.

[41] WILLIAMS, N., ARNULFO, G., WANG, S., et al. "Comparison of methods to identify modules in noisy or incomplete brain networks", *Brain connectivity*, , n. ja.

[42] BRANDES, U., DELLING, D., GAERTLER, M., et al. "On modularity clustering", *IEEE transactions on knowledge and data engineering*, v. 20, n. 2, pp. 172–188, 2008.

[43] NEWMAN, M. E. "Fast algorithm for detecting community structure in networks", *Physical review E*, v. 69, n. 6, pp. 066133, 2004.

[44] GOOD, B. H., DE MONTJOYE, Y.-A., CLAUSET, A. "Performance of modularity maximization in practical contexts", *Physical Review E*, v. 81, n. 4, pp. 046106, 2010.

[45] LANCICHINETTI, A., FORTUNATO, S. "Limits of modularity maximization in community detection", *Physical review E*, v. 84, n. 6, pp. 066122, 2011.

[46] GUIMERA, R., SALES-PARDO, M., AMARAL, L. A. N. "Modularity from fluctuations in random graphs and complex networks", *Physical Review E*, v. 70, n. 2, pp. 025101, 2004.

[47] SPIELMAN, D. A., TENG, S.-H. "Spectral partitioning works: Planar graphs and finite element meshes", *Linear Algebra and its Applications*, v. 421, n. 2-3, pp. 284–305, 2007.

[48] NEWMAN, M. E. "Modularity and community structure in networks", *Proceedings of the national academy of sciences*, v. 103, n. 23, pp. 8577–8582, 2006.

[49] NEWMAN, M. E. “Spectral methods for community detection and graph partitioning”, *Physical Review E*, v. 88, n. 4, pp. 042822, 2013.

[50] PONS, P., LATAPY, M. “Computing communities in large networks using random walks”. In: *International symposium on computer and information sciences*, pp. 284–293. Springer, 2005.

[51] ROSVALL, M., BERGSTROM, C. T. “Maps of random walks on complex networks reveal community structure”, *Proceedings of the National Academy of Sciences*, v. 105, n. 4, pp. 1118–1123, 2008.

[52] REICHARDT, J., BORNHOLDT, S. “Statistical mechanics of community detection”, *Physical Review E*, v. 74, n. 1, pp. 016110, 2006.

[53] HASTINGS, M. B. “Community detection as an inference problem”, *Physical Review E*, v. 74, n. 3, pp. 035102, 2006.

[54] NEWMAN, M. E., LEICHT, E. A. “Mixture models and exploratory analysis in networks”, *Proceedings of the National Academy of Sciences*, v. 104, n. 23, pp. 9564–9569, 2007.

[55] KARRER, B., NEWMAN, M. E. “Stochastic blockmodels and community structure in networks”, *Physical review E*, v. 83, n. 1, pp. 016107, 2011.

[56] LANCICHINETTI, A., RADICCHI, F., RAMASCO, J. J., et al. “Finding statistically significant communities in networks”, *PloS one*, v. 6, n. 4, pp. e18961, 2011.

[57] PEIXOTO, T. P. “Hierarchical block structures and high-resolution model selection in large networks”, *Physical Review X*, v. 4, n. 1, pp. 011047, 2014.

[58] BARBER, M. J., CLARK, J. W. “Detecting network communities by propagating labels under constraints”, *Physical Review E*, v. 80, n. 2, pp. 026129, 2009.

[59] XIE, J., SZYMANSKI, B. K. “Towards linear time overlapping community detection in social networks”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 25–36. Springer, 2012.

[60] NEWMAN, M. “Network data”. url: `http://www-personal.umich.edu/~mejn/netdata`, Accessed March 13, 2019.

[61] ZACHARY, W. W. “An information flow model for conflict and fission in small groups”, *Journal of anthropological research*, v. 33, n. 4, pp. 452–473, 1977.

[62] KREBS, V. “Political patterns on the WWW”. url: `http://www.orgnet.com/leftright.html`, Accessed March 13, 2019.

Appendix A

Detailed statistics of GAMB rounds

Figures A.1, A.2, A.3, and A.4 show detailed statistics of each round in GAMB executions. As explained in the numerical evaluation section of Chapter 5, each figure shows the results obtained from 10 different network instances produced by the planted partition model, and for each instance the algorithms were executed 20 times, independently. The results are displayed as boxplots, in which boxes extend from the lower to upper quartile values of the data, with an orange line indicating the median and a green triangle marking the mean. The whiskers extend from the minimum to the maximum value of the distribution.

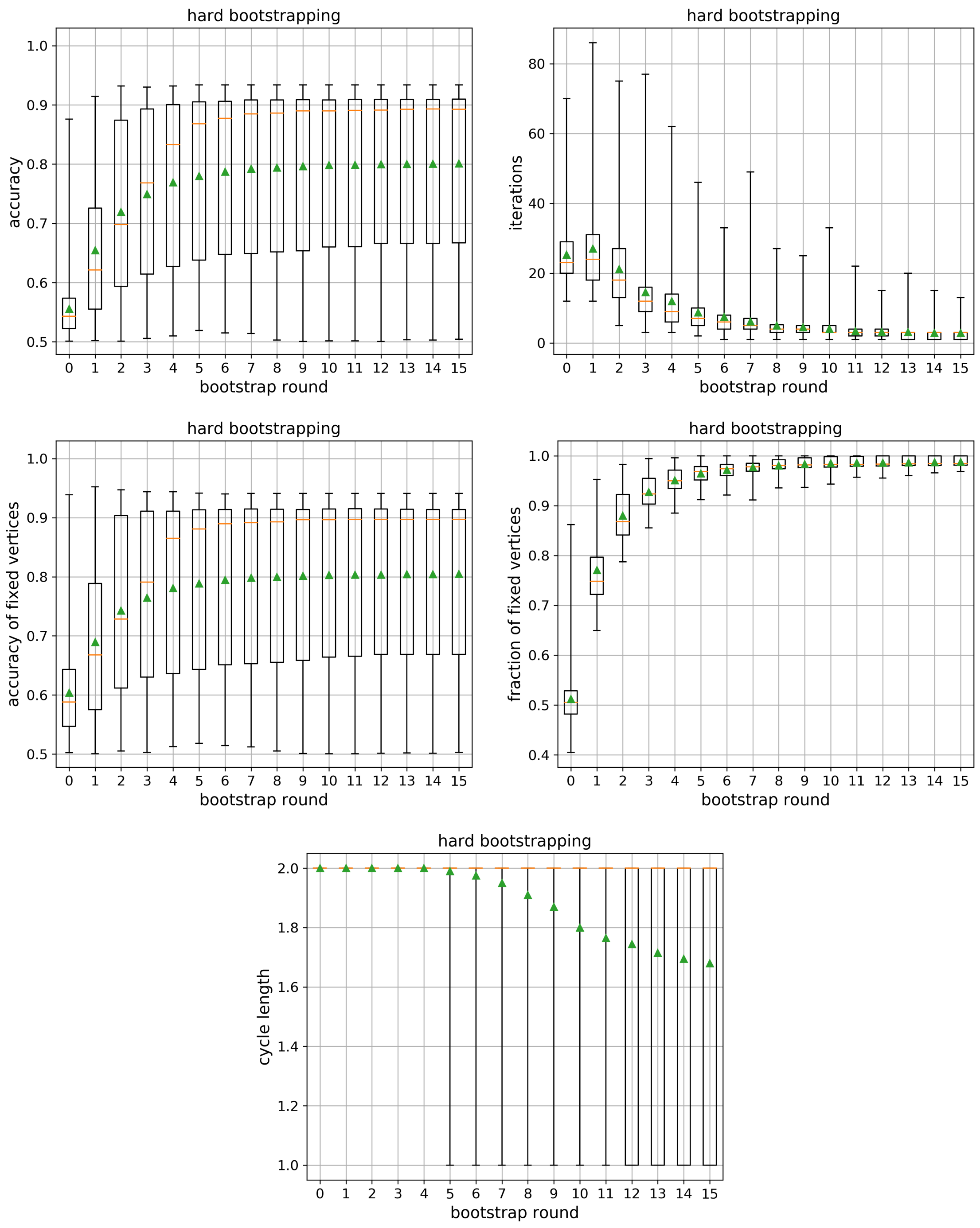


Figure A.1: Hard bootstrapping, $p = 0.01$, $p - q = 0.006$

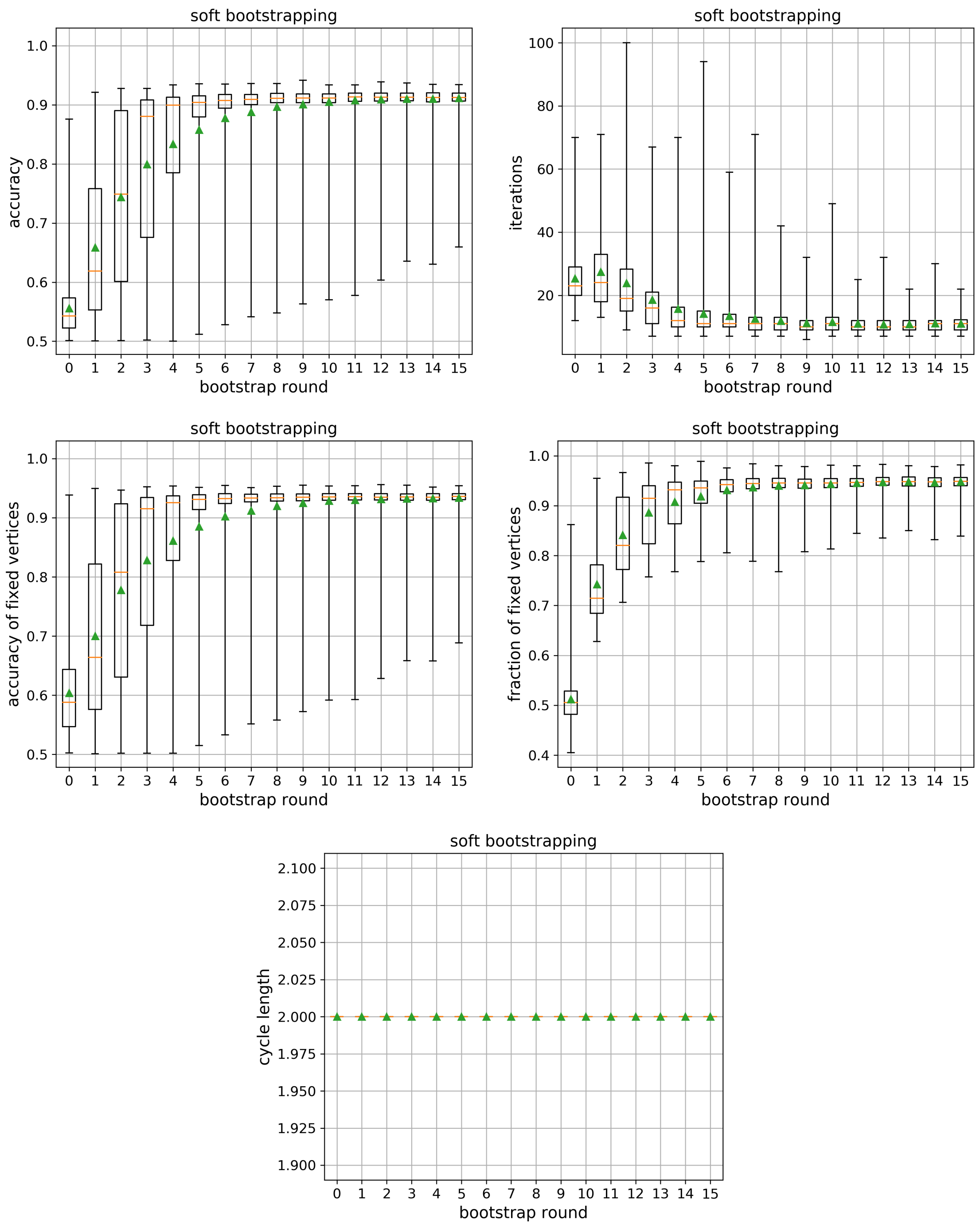


Figure A.2: Soft bootstrapping, $p = 0.01$, $p - q = 0.006$

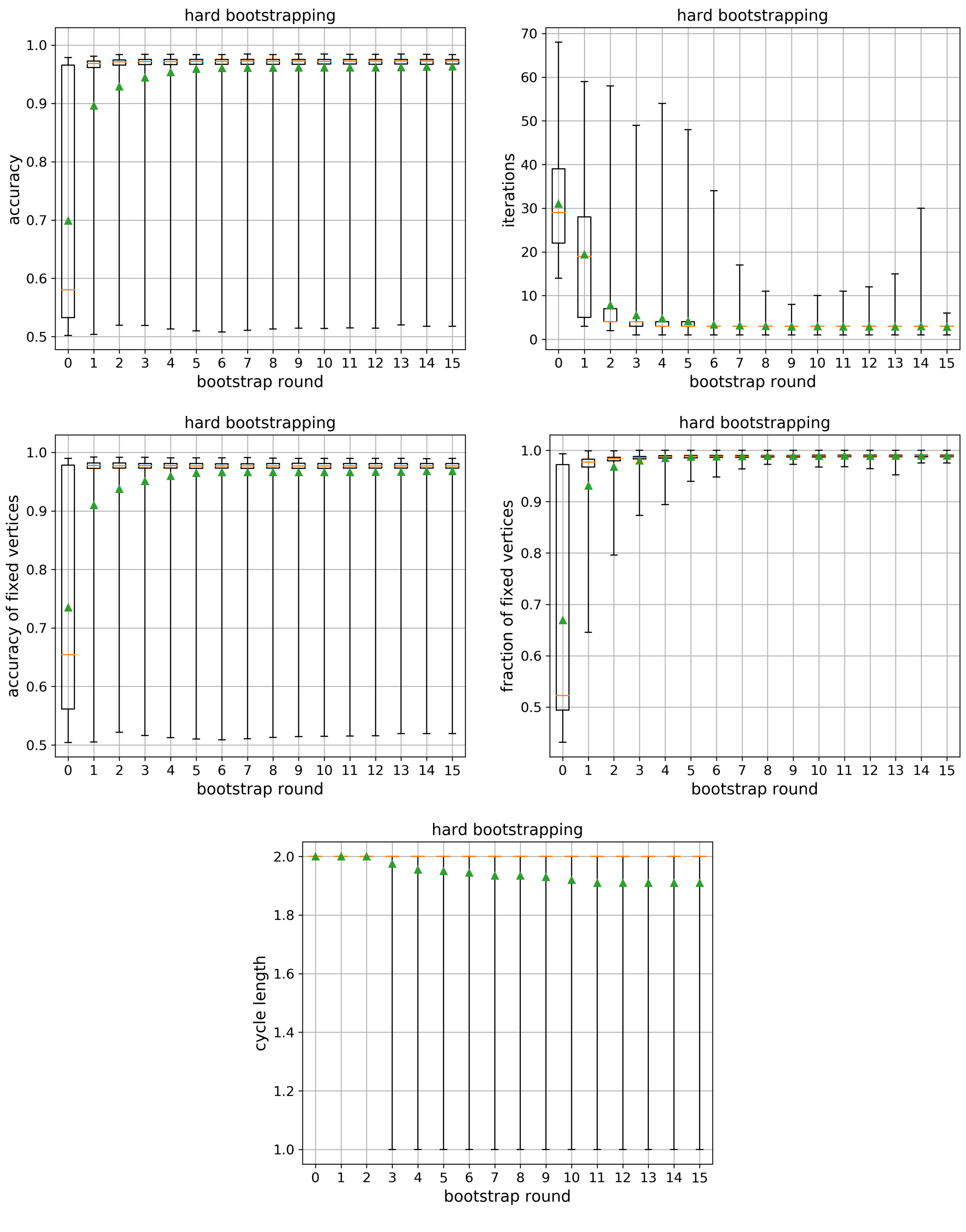


Figure A.3: Hard bootstrapping, $p = 0.01$, $p - q = 0.007$

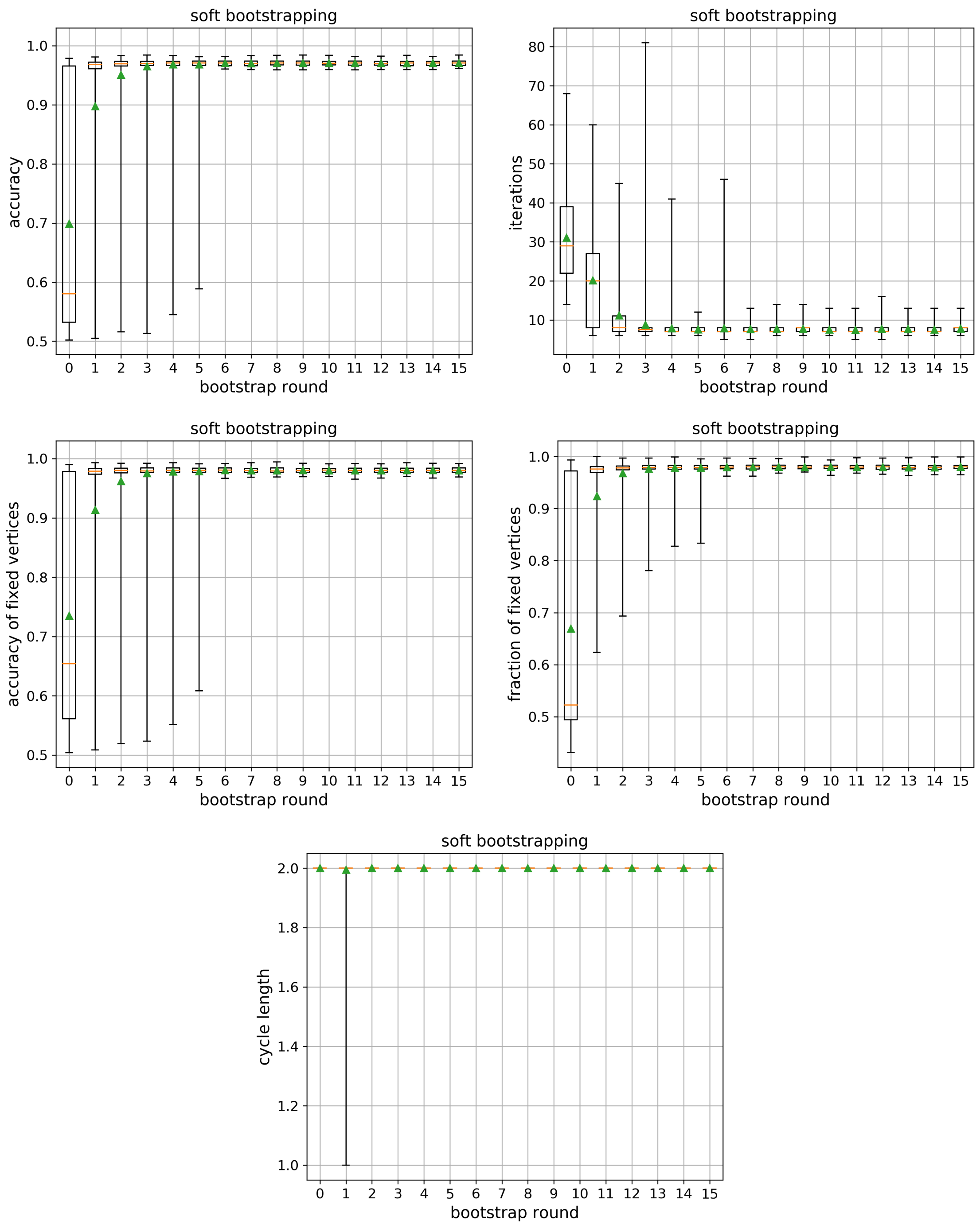


Figure A.4: Soft bootstrapping, $p = 0.01$, $p - q = 0.007$