Pontifícia Universidade Católica
do Rio de Janeiro

Breno Serrano de Araujo

A MIP approach for Community Detection in the Stochastic Block Model

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Thibaut Victor Gaston Vidal

Rio de Janeiro
September 2020

Pontifícia Universidade Católica
do Rio de Janeiro

Breno Serrano de Araujo

A MIP approach for Community Detection in the Stochastic Block Model

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee:

Prof. Thibaut Victor Gaston Vidal
Advisor
Departamento de Informática – PUC-Rio

Prof. Marcus Vinícius Soledade Poggi de Aragão
Departamento de Informática – PUC-Rio

Prof. Rafael Martinelli Pinto
Departamento de Engenharia Industrial – PUC-Rio

Rio de Janeiro, September 17th, 2020

Breno Serrano de Araujo

Obtained a bachelor's degree in Electronics and Computer Engineering (2017) at the Federal University of Rio de Janeiro (UFRJ). Worked at Accenture as a Data Science analyst from 2017 to 2018. Since 2018 works at TechnipFMC as a Research & Development engineer.

Bibliographic data

Serrano de Araujo, Breno

A MIP approach for Community Detection in the Stochastic Block Model / Breno Serrano de Araujo; advisor: Thibaut Victor Gaston Vidal. – 2020.

68 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2020.

Inclui bibliografia

1. Informática – Teses. 2. Detecção de Comunidades. 3. Stochastic Block Model. 4. Programação Inteira Mista. 5. Machine Learning. 6. Aprendizado Não-Supervisionado. 7. Busca Local. I. Vidal, Thibaut Victor Gaston. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

I would like to first thank my advisor Thibaut Vidal for all the great opportunities he has opened up for me and for his close guidance throughout this journey. I wish to thank Marcus Poggi for his insightful contributions and for encouraging me to pursue a Masters degree from the very beginning. I would also like to thank Rafael Martinelli for the valuable comments and suggestions.

I am very grateful to my colleagues Vinícius Ribeiro, Matheus Costa dos Santos and Mário Vignoles for all the good moments at TechnipFMC, where I have learned many important lessons for my professional and personal life.

I would like to thank my family for all the love and support they gave me, especially my parents Beatriz and Paulo, my brother Luiz Paulo and my little sister Giulia.

Finally, I would like to give a very special thanks to my loving girlfriend Juliana who was always on my side through these years.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Serrano de Araujo, Breno; Vidal, Thibaut Victor Gaston (Advisor). **A MIP approach for Community Detection in the Stochastic Block Model**. Rio de Janeiro, 2020. 68p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The Degree-Corrected Stochastic Block Model (DCSBM) is a popular model to generate random graphs with community structure given an expected degree sequence. The standard approach of community detection algorithms based on the DCSBM is to search for the model parameters which are the most likely to have produced the observed network data, via maximum likelihood estimation (MLE). Current techniques for the MLE problem are heuristics and therefore do not guarantee convergence to the optimum. We present mathematical programming formulations and exact solution methods that can provably find the model parameters and community assignments of maximum likelihood given an observed graph. We compare the proposed exact methods with classical heuristic algorithms based on expectation-maximization (EM). The solutions given by exact methods give us a principled way of recognizing when heuristic solutions are sub-optimal and measuring how far they are from optimality.

Keywords

Community detection; Stochastic Block Model; Mixed Integer Programming; Machine Learning; Unsupervised Learning; Local Search.

Resumo

Serrano de Araujo, Breno; Vidal, Thibaut Victor Gaston. **Uma abordagem de Programação Inteira Mista para Detecção de Comunidades no Stochastic Block Model**. Rio de Janeiro, 2020. 68p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O *Degree-Corrected Stochastic Block Model* (DCSBM) é um modelo popular para geração de grafos aleatórios com estrutura de comunidade, dada uma sequência de graus esperados. O princípio básico de algoritmos que utilizam o DCSBM para detecção de comunidades é ajustar os parâmetros do modelo a dados observados, de forma a encontrar a estimativa de máxima verossimilhança, ou *maximum likelihood estimate* (MLE), dos parâmetros do modelo. O problema de otimização para o MLE é comumente resolvido por meio de heurísticas. Neste trabalho, propomos métodos de programação matemática, para resolver de forma exata o problema de otimização descrito, e comparamos os métodos propostos com heurísticas baseadas no algoritmo de *expectation-maximization* (EM). Métodos exatos são uma ferramenta fundamental para a avaliação de heurísticas, já que nos permitem identificar se uma solução heurística é sub-ótima e medir seu *gap* de otimalidade.

Palavras-chave

Detecção de Comunidades; Stochastic Block Model; Programação Inteira Mista; Machine Learning; Aprendizado Não-Supervisionado; Busca Local.

Table of contents

List of figures

List of tables

List of Abreviations

ER – Erdős–Rényi (random graph model)
SBM – Stochastic Block Model
DCSBM – Degree-Corrected Stochastic Block Model
PPM – Planted Partition Model
MLE – Maximum Likelihood Estimation
MAP – Maximum a Posteriori Estimation
EM – Expectation-Maximization
LS – Local Search
MIP – Mixed Integer Programming
MILP – Mixed Integer Linear Programming
MINLP – Mixed Integer Non-Linear Programming
MIQP – Mixed Integer Quadratic Programming
IQP – Integer Quadratic Programming
SDP – Semidefinite Programming
SBC – Symmetry-breaking Constraints
B&B – Branch-and-Bound
sBB – Spatial Branch-and-Bound

1
Introduction

Many networks that occur in the real-world present some form of community structure, where vertices are organized in groups (also called communities). Vertices within the same community present a similar connection pattern. In particular, assortative communities represent groups of individuals that are more densely connected to each other than to individuals of other groups. The Stochastic Block Model (SBM) is a simple yet versatile model for the generation of networks with different types of community structure. Under the SBM, the probability that any two vertices are connected only depends on the community memberships of the corresponding vertices. Various extensions to the SBM have been proposed, for example the Degree-Corrected Stochastic Block Model (DCSBM) [1], in which the expected degrees of the vertices can be further specified.

One important application of the DCSBM is in the task of community detection. Identifying communities offers valuable information about the underlying structure of an observed network [2]. Community detection has important applications, in the study of social networks [3], networks of protein-protein interactions [4], gene expressions [5] and in DNA 3D folding [6], among others. The general principle of community detection methods based on such models is to search for the model parameters, namely the community assignments and the latent variables describing the connectivity matrix, that are the most likely to have generated the observed network data. The model parameters can be estimated via maximum likelihood estimation (MLE) given an observed graph.

Current methods for the MLE problem are heuristics which are not guaranteed to converge to the solution of maximum likelihood. Convergence guarantees focus on the probability of recovering the *true* underlying communities of a graph generated by the SBM in the asymptotic limit when the size of the network grows to infinity. Works in the literature [7, 8] study the community recovery problem from a statistical and information-theoretic point of view and provide thresholds and conditions under which different types of algorithms are able to recover the underlying community assignments with high probability for different asymptotic regimes.

We address the MLE problem, from an optimization standpoint, and

propose mathematical programming formulations which can be solved to optimality by general-purpose optimization solvers. The resulting methods can provably find the optimal solution (and bounds) of the maximum likelihood model for an observed graph. The main contribution of this work is to make a first step towards developing exact solution methods for the MLE problem of community detection in the DCSBM.

We formulate the MLE problem using mixed integer programming (MIP) techniques and propose exact solution methods. Firstly, a straightforward descriptive formulation is proposed, which results in a mixed integer non-linear program (MINLP). The model can be exactly solved with algorithms based on spatial branch-and-bound (sBB), using global optimization solvers. The resulting method can find the DCSBM parameters of maximum likelihood with a certificate of global optimality. However, it is intractable even for very small networks. Building upon it, we employ linearization techniques to produce a mixed integer linear programming (MILP) formulation. To make the formulation solvable in practice we make use of dynamic constraints generation and symmetry-breaking constraints. We carefully analyze the problem to derive tight bounds on the model variables. The MILP model drastically reduces the computation time required by the more naive approach.

The ability to solve machine learning models to optimality has important consequences, as having access to optimal solutions of a problem allows us to study the performance of heuristic approaches in a more principled way and to measure how far a heuristic solution is from the optimum. To illustrate this, we discuss three variants of the expectation-maximization (EM) algorithm and run computational experiments to compare the heuristic solutions with those found by the exact solution approaches in terms of likelihood and proximity to the ground truth.

This work is organized as follows: Chapter 2 introduces important background concepts on random graph models, including the SBM and some of its variants, such as the DCSBM. In Chapter 3 we formally state the MLE problem, briefly review related works and discuss our main contributions. Next, we present the mixed integer programming formulations and exact solution approaches in Chapter 4. As a means of comparison we describe some simple heuristic approaches based on the EM algorithm in Chapter 5. Then, Chapter 6 details the computational experiments and discusses their results and Chapter 7 concludes this work.

2
Theoretical Foundations

This chapter introduces the theoretical basis of the Stochastic Block Model. In order to provide some background on the methods proposed in this work, we first briefly present some classical models of random graphs [9, 10] and discuss their connections to the SBM (and its main variants).

2.1
Random Graphs

The term random graph is used to refer to probability distributions over graphs (i.e. a probability distribution of a graph-valued random variable) [11]. Whereas a random variable can be described by a probability distribution, a random graph can be described by a network model, in which the values of certain network properties are fixed, while other properties are random. Many random graph models with different characteristics have been proposed in the literature [9, 12, 13, 14], and the Stochastic Block Model is a particular case. This section describes a few fundamental network models which are useful for understanding the Stochastic Block Model and its variants.

2.1.1
Erdős–Rényi Random Graph Model

The Erdős–Rényi (ER) model of random graphs was first proposed by Paul Erdős and Alfréd Rényi in 1959 [15, 16]. The Erdős–Rényi model $\mathrm{ER}(n, p)$ can be specified by two parameters: the number of vertices n in the graph and the edge probability p between any pair of vertices. Under the ER model, the vertex set V is defined a priori and, for each pair of vertices in V, an edge is present with probability p, independent of any other pair of vertices. We write $G \sim \mathrm{ER}(n, p)$ to denote that the graph G is drawn under the ER model. The resulting graph is undirected and can be represented by a symmetric adjacency matrix A. It can be equivalently stated that each element A_{ij} follows a Bernoulli distribution with mean p, $A_{ij} \sim \mathrm{Bernoulli}(p)$, for $i < j$. Figure 2.1 illustrates some realizations of the ER model.

The expected number of edges on the graph G is equal to $\binom{n}{2}p = \frac{1}{2}n(n-1)p$. One important property of the ER model is that, for any vertex

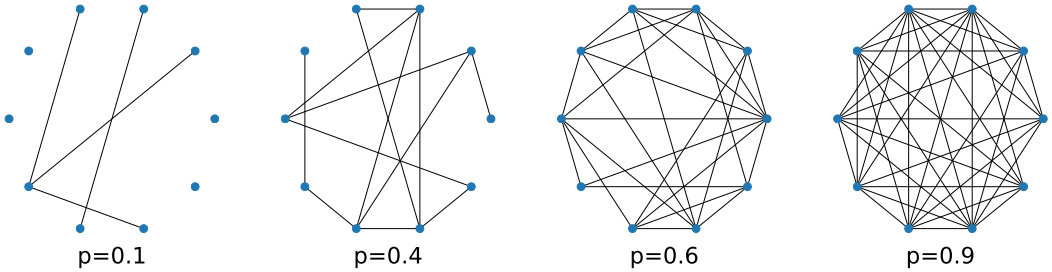


Figure 2.1: Realizations of the ER model for $n = 10$ and different values of p

i, the probability $P(k_i = k)$ that i has degree exactly k is given by a binomial distribution: $P(k_i = k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}$. For large enough n, the random variables k_i and k_j associated with $P(k_i = k)$ and $P(k_j = k)$ can be assumed to be independent for different vertices i and j.

The binomial distribution can be approximated by the Poisson distribution for large n if $np \to \lambda$, where λ is a constant. This means that the edge probability p remains small relative to the size of the network. Therefore for large networks, assuming $p = \frac{\lambda}{n}$, the degree distribution of the Erdős–Rényi random graph is approximately Poisson with mean $\lambda = np$:

$$P(k_i = k) \to \frac{\lambda^k e^{-\lambda}}{k!} = \frac{(np)^k e^{-np}}{k!} \tag{2-1}$$

For this reason the model is often also called the "Poisson random graph" in the literature.

The ER model was proposed as a model for studying the behavior of real-world networks and it has many important properties which make it a useful theoretical tool. Nevertheless, Albert and Barabási (2002) [14] show that the model has some properties which make it fundamentally different from real-world networks. In particular, it is empirically shown that many large networks have a right-skewed degree distribution, where most of the vertices have a low degree and only a small number of vertices in the tail of the distribution have high degree. The degree distribution of many real-world networks can thus be approximated by a power law distribution $P(k_i = k) \simeq k^{-\gamma}$, as opposed to the Poisson distribution produced by the model.

To address this shortcoming various extensions to the ER model have been proposed which allow for non-Poisson degree distributions. In the next section we present one such model which admits any desired degree distribution, called the configuration model.

2.1.2 Configuration Model

The configuration model [9] is an extension of the ER model which allows for the generation of networks with a given *degree sequence*. That is, for each vertex $i \in V$, its degree k_i is specified (and is a fixed value). The degree sequence can be directly specified or can be drawn from a theoretical distribution (such as the Poisson or power-law distribution). The generation of the network then works as follows.

Each vertex i receives a total of k_i "stubs" (also called "half-edges"). The sum of stubs in the graph is equal to $\sum_i k_i = 2m$, where m is the total number of edges. The sum of stubs must be an even number, and the value of m is therefore specified. Two stubs are then chosen uniformly at random and connected to form an edge. Another pair of stubs is chosen from the remaining $2m-2$ stubs in the graph and connected together, and this process is repeated until there are no more remaining stubs in the graph. As a result, every vertex of the graph has the degree exactly as specified. The resulting network may contain self-edges or multi-edges, however the expected number of self-edges and multi-edges tends to zero for large n (as n tends to infinity).

One special case of the configuration model is when the degree sequence is drawn from the Poisson distribution, in which case the random graph model is closely equal to the ER(n, p) model. Both models are not exactly equal since the configuration model allows for the creation of self-edges and multi-edges, as opposed to the ER model. However, this difference can be ignored for large networks.

An important property of the configuration model is the expected number of edges p_{ij} between vertices i and j. Consider a stub from vertex i. The total number of stubs in the network, excluding the one we are considering, is equal to $2m-1$. Out of the $2m-1$ stubs in the network, exactly k_j are assigned to vertex j. Since stubs are connected together uniformly at random, the probability that the stub from vertex i connects with any of the stubs from vertex j is given by $k_j/(2m-1)$. Since vertex i has k_i stubs, the expected number of edges between vertices i and j is:

$$p_{ij} = \frac{k_i k_j}{2m-1} \simeq \frac{k_i k_j}{2m} \tag{2-2}$$

where the second equality is approximately equal for large m.

The configuration model plays an important role in the field of Network Science, greatly due to the importance of the degree distribution in real-world networks. We refer the reader to [9] for a more detailed introduction to this network model and a discussion about its key properties. The next section

presents the Stochastic Block Model, which can be viewed as an extension of the ER model to produce graphs with block structure.

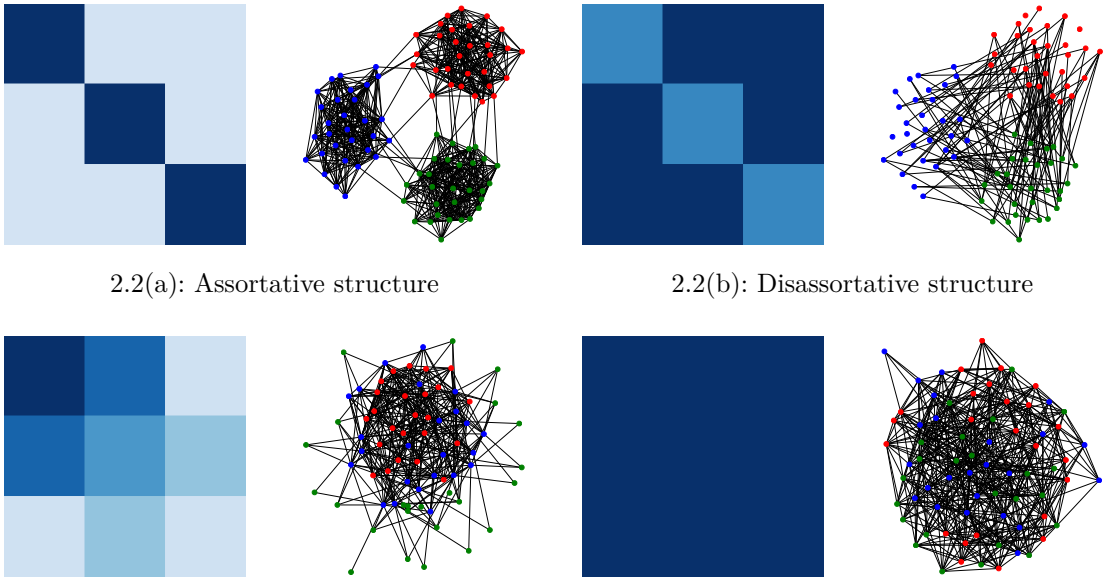
2.1.3 Stochastic Block Model

Many real-world networks present some form of *community structure*, where vertices are arranged in *groups* (also denoted *communities*, *clusters* or *modules* in the literature) [2]. The classic view of community structure can be expressed by the notion that the density of connections within a community is higher than between different communities. This notion captures the interaction pattern of some real-world networks, where members of a group interact more strongly with other members of the same group, than they do with members of other groups. There exists, however, no general consensus on the precise definition of a community, and this classical notion has often been challenged [17, 18]. The Stochastic Block Model provides one way of defining communities, in terms of connection probabilities.

The Stochastic Block Model (SBM) is a generative model of random graphs with community structure [3, 20, 21, 22]. The number of vertices n and the number of communities K is fixed and specified a priori. Each vertex in the network has a unique community assignment (each vertex belongs to exactly one community). The group of vertex i is assumed to be known and is given by $g_i \in \{1, \ldots, K\}$. Edges are then placed in the graph such that the probability that vertices i and j are connected only depends on their group memberships, and is given by $\omega_{g_i g_j} \in [0, 1]$. Hence, the SBM can be defined by a group membership vector $\boldsymbol{g} \in \{1, \ldots, K\}^n$, which encodes the community assignment of each vertex $i \in \{1, \ldots, n\}$, and by a $K \times K$ connectivity matrix $\boldsymbol{\Omega}$ (also called affinity matrix in the literature), which represents the edge probabilities for each pair of communities. We write $G \sim \text{SBM}(\boldsymbol{g}, \boldsymbol{\Omega})$ to denote that the graph G is drawn under the SBM with parameters $\boldsymbol{g}$ and $\boldsymbol{\Omega}$.

The SBM generalizes the classic notion of community structure, in that it allows for the generation of different types of block structure, as illustrated in Figures 2.2 and 2.3 for an SBM with three groups. If the connection probabilities are higher inside the diagonal blocks than elsewhere, namely $\omega_{qq} > \omega_{rs}, \forall q, r, s \in \{1, \ldots, K\}$, with $r \neq s$, then the SBM is said to have an *assortative structure* [23] (which represents the classic view of community structure). If $\omega_{qq} < \omega_{rs}, \forall q, r, s \in \{1, \ldots, K\}$, with $r \neq s$, then we have a *disassortative structure*, and edges are more likely between different communities than inside them. The special case in which all probabilities in the affinity matrix are equal, $\omega_{rs} = p, \forall r, s$, is equivalent to the classic Erdős-Rényi

random graph model ER(n, p).



2.2(a): Assortative structure

2.2(b): Disassortative structure

2.2(c): Core-periphery structure

2.2(d): Random graph

Figure 2.2: Four network realizations produced by the SBM with three communities and with different types of community structure (inspired by Figure 8 of Fortunato and Hric, 2016 [2]). Each figure depicts the underlying affinity matrix $\mathbf{\Omega}$ (left) and the resulting graph (right) drawn under the SBM. The affinity matrix is represented in a lightness scale where darker colors correspond to higher probability values and lighter colors correspond to lower values.

Amini et al. [24] further define two notions of assortativity: *strong* and *weak* assortativity. The former corresponds to our previous definition of assortativity, which can be equivalently stated as:

$$\min_q \omega_{qq} > \max_{r \neq s} \omega_{rs}. \tag{2-3}$$

The model is weakly assortative if each element in the diagonal of $\mathbf{\Omega}$ is greater than or equal to the other elements in its row:

$$\omega_{qq} > \omega_{qs}, \forall q, s \in \{1, \ldots, K\}. \tag{2-4}$$

Strong assortativity implies weak assortativity, but the converse is not true. The SBM also enables the generation of other types of structure, such as core-periphery, hierarchical, or multipartite structures, among others.

The SBM as described above produces simple graphs (without self-edges or multi-edges). In this case, each element of the adjacency matrix A is a random variable with a Bernoulli distribution: $A_{ij} \sim \text{Bernoulli}(\omega_{g_i g_j})$, for $i < j$ and $A_{ii} = 0, \forall i$. If the networks are allowed to contain multi-edges and self-edges, then $\omega_{g_i g_j}$ no longer represents the probability of an edge between vertices i and j, but rather the expected number of edges between them. The

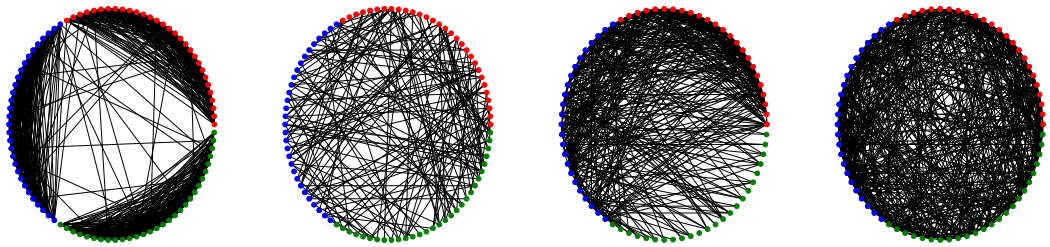


Figure 2.3: Graphs with different types of community structure produced by the SBM with three groups. The same networks of Figure 2.2 are drawn in this figure with the vertices arranged in a circle. This alternative visualization further highlights the differences in the resulting network structures. From left to right, the following structures are illustrated: *assortative*, *disassortative*, *core-periphery* and *random graph*.

number of edges between any pair of vertices is a Poisson-distributed random variable with mean $\omega_{g_i g_j} \in \mathbb{R}^+$. Hence, $A_{ij} \sim \text{Poisson}(\omega_{g_i g_j})$, for $i < j$. By convention, the diagonal element A_{ii} is equal to twice the number of self-edges from i to itself (and thus is always an even number).

In summary, two variants of the standard SBM were presented: the Bernoulli version which generates undirected simple graphs, and the Poisson version which produces undirected multi-graphs. In this work we assume that graphs are allowed to have self-edges and multi-edges and thus adopt the Poisson-distributed SBM, unless otherwise noted. The next section presents a popular extension of the SBM which allows for heterogeneous vertex degrees.

2.1.4 Degree-Corrected Stochastic Block Model

The degree-corrected SBM (DCSBM) was introduced by Karrer and Newman [1] in 2011. They argued that the standard SBM is not well suited for many applications to real-world networks, since the model cannot generate networks with structure similar to those found in empirical network data. They claim that a key reason is that the standard SBM does not take into account the variation in vertex degree, and thus proposed to address this problem by extending the SBM to consider the heterogeneity in the degrees of vertices. The DCSBM is an extension of the SBM in a similar way as the configuration model is an extension of the ER model, allowing for the generation of networks with arbitrary degree distributions.

The DCSBM introduces a new set of parameters θ_i that control the expected degree of vertex i. The number of edges between a pair of vertices i and j, with $i < j$, is drawn from a Poisson distribution with mean $\theta_i \theta_j \omega_{g_i g_j}$:

$$A_{ij} \sim \text{Poisson}(\theta_i \theta_j \omega_{g_i g_j}) = \frac{(\theta_i \theta_j \omega_{g_i g_j})^{A_{ij}}}{A_{ij}!} \exp\left(-\theta_i \theta_j \omega_{g_i g_j}\right), \tag{2-5}$$

which depends not only on the group memberships g_i and g_j, but also on the expected degrees θ_i and θ_j of the vertices. The diagonal elements A_{ii} are again equal to twice the number of self-edges from i to itself. As before, the number of vertices n and the number of groups K are fixed and specified a priori. The DCSBM is specified by the community assignments $\boldsymbol{g}$, the affinity matrix $\boldsymbol{\Omega}$ and the expected vertex degrees $\boldsymbol{\theta}$:

$$G \sim \text{DCSBM}(\boldsymbol{g}, \boldsymbol{\Omega}, \boldsymbol{\theta}). \tag{2-6}$$

In their work, Karrer and Newman analyze real-world networks and experimentally show that this minor extension to the SBM provides a much better fit when performing community detection.

2.1.5 Planted Partition Model

The Planted Partition Model (PPM) is a special case of the SBM where all diagonal elements of the affinity matrix $\boldsymbol{\Omega}$ are equal to p, and all off-diagonal elements are equal to q. In other words, $\omega_{rs} = p$ if $r = s$ and $\omega_{rs} = q$ otherwise if $r \neq q$. Thus, the affinity matrix is defined by just two parameters p and q via the equation:

$$\boldsymbol{\Omega} = q\boldsymbol{E}_K + (p - q)I_K \tag{2-7}$$

where $\boldsymbol{E}_K$ is the $K \times K$ matrix of all ones and I_K is the $K \times K$ identity matrix. We write $G \sim \text{PP}(\boldsymbol{g}, p, q)$ to denote that the graph G is drawn under the PPM with community assignments $\boldsymbol{g}$ and parameters p and q. One special case of the PPM is the balanced Planted Partition Model, $\text{PP}^{\text{bal}}(\boldsymbol{g}, p, q)$, which further assumes that the communities have equal sizes. For simplicity we may omit the dependence on $\boldsymbol{g}$ and write $\text{PP}(p, q)$ and $\text{PP}^{\text{bal}}(p, q)$.

3
Community Detection with the Stochastic Block Model

Community detection refers to the task of identifying communities in a network (also called *graph* or *network clustering* in the literature). Figure 3.1 illustrates the general idea of community detection: given a network, where the only information available is the presence of edges between vertices, the aim is to identify to which community each vertex belongs. This is, however, a loosely defined problem, since there exists no universal definition of a community. Consequently, there is no universal way of comparing different approaches to community detection and assessing their performance. Fortunato and Hric [2] argue that the lack of a universal definition of a community gives a lot of freedom for researchers to propose diverse approaches to the problem. This has lead to the development of a multitude of methods, for example based on consensus clustering [25], spectral clustering [26], modularity maximization [27] and methods based on statistical inference, among others [2].

This work focuses on the problem of community detection based on the degree-corrected SBM, which is the most popular approach among the methods based on statistical inference. The standard approach is to fit the generative model to the observed network data, where the number of communities K is assumed to be known. Fortunato and Hric [2] discuss some heuristic techniques for determining the value of K.

In this chapter, we describe the main concepts behind community detection based on the DCSBM. We then formally present the problem statement, briefly review related works in the literature and discuss the main contributions of this work.

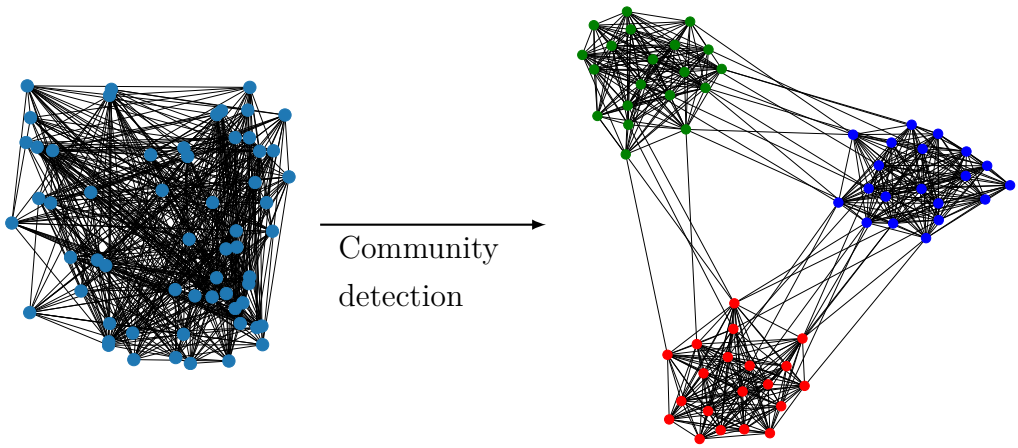


Figure 3.1: Given an observed network (left) the task of community detection consists in identifying the underlying assignments of vertices to communities (right).

3.1 Maximum Likelihood Estimation

In the community detection problem we observe a graph $G = (V, E)$ and aim to identify communities in it. As usual, the total number of vertices is $|V| = n$, the total number of edges is $|E| = m$, and the graph is represented by an adjacency matrix A. When the DCSBM is used for community detection, the standard approach is to search for the model parameters that provide the best fit to the observed graph G.

Maximum likelihood estimation (MLE) [28] is a common method for finding the parameter values of a model or probability distribution given observed data. The parameter values are estimated by maximizing a likelihood function, such that under the assumed statistical model the observed data is most likely. If the individual observations are assumed to be statistically independent, then the probability density function (PDF) of the data given the parameters can be expressed as a product of PDFs of the individual observations.

Assuming the Poisson version of the DCSBM, and given the values of the model parameters, the probability that the observed network was generated from the DCSBM can be expressed as:

$$P(A|\boldsymbol{g},\boldsymbol{\Omega},\boldsymbol{\theta}) = \prod_{i<j} \frac{(\theta_i\theta_j\omega_{g_ig_j})^{A_{ij}}}{A_{ij}!} \exp\left(-\theta_i\theta_j\omega_{g_ig_j}\right)\times \prod_i \frac{\left(\frac{1}{2}\theta_i^2\omega_{g_ig_i}\right)^{A_{ii}/2}}{\left(\frac{1}{2}A_{ii}\right)!} \exp\left(-\tfrac{1}{2}\theta_i^2\omega_{g_ig_i}\right) \tag{3-1}$$

which defines the likelihood function of the DCSBM.

In this work we adopt the special case where $\theta_i\theta_j = \frac{k_ik_j}{2m}$, as in Newman (2016) [19], where $\frac{k_ik_j}{2m}$ corresponds to the expected number of edges in the configuration model. With this definition, the parameter $\omega_{g_ig_j}$ now quantifies the expected number of edges between vertices i and j, relative to the configuration model.

As usual, it is more convenient to maximize the log-likelihood function. After removing constant terms which do not affect the position of the optimum, Equation (3-1) simplifies to:

$$\log P(A|\boldsymbol{g},\boldsymbol{\Omega}) = \tfrac{1}{2}\sum_{i,j}^{n}\left(A_{ij}\log\omega_{g_ig_j} - \tfrac{k_ik_j}{2m}\omega_{g_ig_j}\right) \tag{3-2}$$

The complete derivation is available in Appendix A. The solution to the MLE problem is given by an affinity matrix $\boldsymbol{\Omega}$ and group membership assignments $\boldsymbol{g}$, which maximize the log-likelihood function (3-2).

3.2 Community Recovery

The literature on community detection with the SBM mainly focuses on the problem of recovering the true underlying community labels of an observed network G, drawn under the SBM, in the asymptotic limit as the size of the network grows to infinity (i.e. as n tends to infinity). Three types of recovery requirements are described by Abbe (2018) [7]: *exact recovery*, *almost exact recovery* and *weak recovery* (also called *detection*). The model adopted in this work is based on Karrer and Newman (2011) [1] and Newman (2016) [19], and is slightly different from the one described by Abbe, in that the community assignments $\boldsymbol{g}$ are specified and deterministic, rather than drawn from a probability distribution over the community labels.

The recovery requirements rely on an agreement function, defined as follows:

Agreement. Let $\mathbf{x},\mathbf{y} \in \{1,\ldots,K\}^n$ be two community assignment vectors. The agreement between $\mathbf{x}$ and $\mathbf{y}$ is the maximum number of common elements between $\mathbf{x}$ and any permutation of the community labels of $\mathbf{y}$:

$$A(\mathbf{x}, \mathbf{y}) = \max_{\pi \in \Pi_K} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(x_i = \pi(y_i)) \tag{3-3}$$

where Π_K is the set of all permutations of the set of labels $\{1, \ldots, K\}$. In other words, the agreement function $A(\cdot, \cdot) \in [0, 1]$ measures the agreement between two membership vectors, considering all possible permutations of community labels.

Let G be a graph drawn under the SBM with true underlying community assignments $\boldsymbol{g}^*$. The recovery requirements as described by Abbe (2018) [7] can be defined as follows:

Exact recovery. An algorithm performs exact recovery if it takes as input the graph G and correctly recovers the underlying assignments of nodes to communities $\boldsymbol{g}^*$ with high probability (i.e., with probability tending to 1 as $n \to \infty$), considering all possible permutations of community labels:

$$\mathcal{P}[A(\hat{\boldsymbol{g}}, \boldsymbol{g}^*) = 1] \xrightarrow[n \to \infty]{} 1 \tag{3-4}$$

where $\hat{\boldsymbol{g}}$ is the vector of community assignments estimated by the algorithm.

Almost exact recovery. Whereas exact recovery requires all community assignments to be correctly recovered, almost exact recovery allows for a vanishing fraction of mislabelled vertices:

$$\mathcal{P}[A(\hat{\boldsymbol{g}}, \boldsymbol{g}^*) \to 1] \xrightarrow[n \to \infty]{} 1 \tag{3-5}$$

Weak recovery. This recovery requirement is in fact a special case of *partial recovery* [7], which allows for a constant fraction $\alpha \in (0, 1)$ of mislabelled vertices in the vector of community assignments:

$$\mathcal{P}\left[A(\hat{\boldsymbol{g}}, \boldsymbol{g}^*) \geq \alpha\right] \xrightarrow[n \to \infty]{} 1 \tag{3-6}$$

where α is called the agreement (or accuracy) of the algorithm. If the value of α is too small, partial recovery may not be an interesting requirement. In the SBM with K balanced communities (i.e. communities with equal sizes) we can achieve an accuracy of $1/K$ with an algorithm that simply ignores the graph G and draws community assignments $\hat{\boldsymbol{g}}$ at random. Thus, weak recovery requires that the algorithm performs significantly better than random guessing, which is realized if there exists $\epsilon > 0$, such that:

$$\mathcal{P}\left[A(\hat{\boldsymbol{g}}, \boldsymbol{g}^*) \geq \tfrac{1}{K} + \epsilon\right] \xrightarrow[n \to \infty]{} 1 \tag{3-7}$$

In the recovery requirements described above the affinity matrix $\mathbf{\Omega} = \mathbf{\Omega}(n)$ is assumed to be dependent on n, while the number of communities K is fixed and the relative sizes of the communities in $\boldsymbol{g}^*$ remains constant (i.e. the size of the communities grows linearly with n). Different asymptotic regimes are considered in the literature [7, 29] which describe how $\mathbf{\Omega}$ scales as a function of n. For example, following Del Pia *et al.* (2020) [29] three regimes can be distinguished:

- Very dense regime: $\mathbf{\Omega}(n) = \Theta(1)$
- Dense regime: $\mathbf{\Omega}(n) = \Theta(n^{-a})$ for $0 < a < 1$
- Sparse regime: $\mathbf{\Omega}(n) = \Theta(\log n/n)$

The solution to the MLE problem can be shown to recover the true community assignments with high probability, under some conditions on the parameter values of the connectivity matrix and on the asymptotic regime. Although community recovery can be solved, in some cases, via maximum likelihood estimation, the two problems (MLE vs. recovery) are fundamentally different. The recovery requirements are all asymptotic, and convergence guarantees focus on the probability of recovering the *true* underlying parameters of the model. In contrast, maximum likelihood estimation is a statistical method which may be applied without assuming the existence of true underlying community assignments $\boldsymbol{g}^*$.

In principle, maximum likelihood estimation may be applied to any network data, without the assumption that it was produced by a specific model (such as the SBM), for example to gain insights about the structure of the graph. In the non-asymptotic case, where we observe a (fixed) graph G, the optimal parameter values of maximum likelihood do not necessarily correspond to the true underlying parameters (ground truth parameters may not even exist in general). Moreover, the asymptotic regime does not occur in real-world applications and we can assume that asymptotic results hold only approximately, for large networks. The distinction between the MLE problem and community recovery is important, as this work focuses solely on the former problem, while most of the works in the literature address the latter.

3.3 Literature Review

There is a rich literature on Stochastic Block Models. They have been studied by various different communities, through the lens of statistics, computer science, machine learning, information theory, social sciences, statistical physics and mathematics. For general surveys on community detection and

SBMs we highlight the works by Fortunato *et al.* (2010) [2], Abbe (2017) [7] and Lee *el al.* (2019) [30].

A large body of literature on SBMs focuses on community recovery, specifically on the three recovery requirements described in the previous section, and on the problem of estimating the affinity matrix of an SBM by observing a realization of the graph (often called *parameter estimation* or *parameter learning*). Special cases of the SBM, such as the (balanced) Planted Partition Model are often considered. We briefly summarize a few results in this section.

Decelle *el al.* (2011) [31] address both community recovery and parameter learning. Their approach is rooted in a statistical physics perspective, and uses the cavity method of spin glasses to analyze the phase diagram. That is, they show that there is a range of parameter values where recovery is impossible, another region where it is possible, but exponentially hard, and finally a region where a belief propagation algorithm exactly recovers the communities and model parameters (in the asymptotic limit).

Abbe *et al.* (2015) [8] discuss a sharp threshold phenomenon (also called phase transition) for exact recovery in the balanced PPM, $\mathrm{PP}^{\mathrm{bal}}(p, q)$, with two communities (also denoted planted bisection model). Specifically, they show that recovering the underlying communities with high probability (as $n \to \infty$) is possible if $|\sqrt{a} - \sqrt{b}| > \sqrt{2}$ and impossible if $|\sqrt{a} - \sqrt{b}| < \sqrt{2}$, where $p = a\log(n)/n$ and $q = b\log(n)/n$ are constants with $a > b$ (i.e. strong assortativity is assumed). They introduce an efficient algorithm based on a semidefinite programming (SDP) relaxation of the MLE model and prove that it succeeds in exact recovery whenever it is information-theoretically possible. The algorithm is then proven by [32] to exactly recover the communities also at the threshold, as conjectured by [8].

Almost exact recovery is proven to be possible for the planted bisection model if and only if $n(p-q)^2/(p+q)$ diverges (as $n \to \infty$) [33], and the result is extended to general SBMs in [34].

For weak recovery, [35, 36] prove the existence of a sharp threshold phenomenon for the planted bisection model, as conjectured by [31]. Specifically, if $p = a/n$ and $q = b/n$, then weak recovery is possible if and only if $(a-b)^2 > 2(a+b)$.

Abbe and Sandon (2015) [34] investigate the phase transition phenomenon for the general (unbalanced) SBM with multiple communities. In particular they characterize a recovery threshold in terms of a proposed divergence function and introduce efficient algorithms for partial and exact recovery in different asymptotic regimes.

Various works propose algorithms for community recovery based on SDP relaxations of MLE [37, 38, 39], and prove results related recovering communities in SBMs with general K, and with an implicit assumption of strong assortativity. Amini *et al.* (2018) [24] propose an SDP relaxation which is tighter than previous ones, and that works for a wider class of SBMs, including for disassortative structures.

Del Pia *et al.* (2020) [29] consider the problem of exact recovery for the assortative planted bisection model and discuss the theoretical performance of linear programming (LP) relaxations of the minimum bisection problem for community recovery. They derive sufficient and necessary conditions for recovery using the LP relaxation for different asymptotic regimes.

Abbe (2017) [7] surveys the main recent developments on community detection for the PPM as well as for general SBMs, with a focus on community recovery and the characterization of threshold phenomena, for different recovery requirements and different asymptotic regimes. The survey also discusses the performance of some algorithms in recovering communities, for example based on SDP, belief propagation and spectral methods [40, 41, 42, 43]. For parameter learning, the survey reviews consistency analyses and optimality guarantees of different approaches, in the sense of recovering the true underlying parameters of the SBM in the asymptotic limit.

Algorithms based on expectation-maximization for MLE have been investigated, for example by [22] for the SBM with two communities. However, their method is practical only for small graphs. For large graphs they introduce a Bayesian estimation method based on Gibbs sampling. The EM algorithm is also used to maximize the pseudo-likelihood of the SBM parameters in Amini *et al.* (2013) [44]. The general idea of pseudo-likelihood is to approximate the likelihood by ignoring some of the dependency structure of the data and thus making the model more tractable.

In the Bayesian framework, other algorithms have been proposed for the MLE problem, for example Markov Chain Monte Carlo (MCMC) methods [45, 46, 47], variational methods [48, 49, 50, 51] and an algorithm based on belief propagation [31] (as mentioned above).

Some authors have explored exact solution methods for community detection based on modularity maximization [27, 52, 53], which is known to be $\mathcal{NP}$-hard [54]. Newman (2016) [19] shows that modularity maximization is equivalent to MLE of the PPM. The problem of modularity maximization is directly formulated as a mixed integer quadratic program (MIQP) by Xu *et al.* (2007) [52], and solved using a branch-and-bound method. They discuss the use symmetry-breaking constraints to improve the efficiency of the branch-

and-bound exploration. Aloise *et al.* (2010) [27] employ techniques based on column generation to improve on previous works [52, 53], reporting a reduction in computing time, and solving larger instances up to 512 vertices exactly (vs. 105 vertices in previous works). Modularity maximization (equivalently MLE of the PPM) is, however, a very specialized case of community detection, and no exact solution algorithms are known for MLE of the general SBM.

In a broader sense, this dissertation has connections with other works that investigate the use of techniques from mixed integer optimization applied to classical machine learning models. Mixed integer optimization has been recently used to learn optimal decision trees [55], Gaussian mixture models (GMM) [56, 57], ramp-loss support vector machines (SVM) [58] and for Bayesian network structure learning [59], just to name a few examples. Recent surveys [60, 61] discuss various commonly used machine learning models from a mathematical optimization perspective.

3.4 Contribution

The main contribution of this work is to make a first step towards developing exact solution methods for the MLE problem (described by Equation 3-2) of community detection with the DCSBM. Our solution methods are based on mathematical programming. To the best of our knowledge, this is the first work to address the problem with an exact approach. In our opinion, the study of exact algorithms are important for many reasons. Here we emphasize a few points:

- Exact algorithms provide benchmarks of exactly solved instances which can be used to evaluate the performance of heuristics. Studying the optimal solutions of a problem may provide insights on strategies to improve the current heuristic approaches.
- Exact algorithms may be stopped before finishing execution, if the run time exceeds a desired time limit. In this case the algorithm provides the best found solution, together with a bound on the solution optimality. However, the optimal solution may often be found at an early stage of the optimization process.
- There is active research in the field of mathematical programming and optimization solvers are regularly being improved, enabling the solution of increasingly bigger problems. Even though most MIP problems are $\mathcal{NP}$-hard, it is now possible to solve instances with complexity and dimensions that were impractical a few decades ago. This increase in

computational power of optimization solvers is largely due to algorithmic improvements which, coupled with hardware improvements over the last decades, has resulted in speedup factors of the order of 10^{11} (as estimated by Bixby, 2012 [62] and later by Bertsimas *et al.*, 2017 [55]).

Despite the advantages of exact solution methods, their use is currently still intractable for many $\mathcal{NP}$-hard problems and heuristic approaches are the only option for finding solutions in a reasonable computing time. Good heuristics may often find solutions which closely approximate the optimum. Improving heuristic approaches may also lead to the development of more efficient exact methods, as many exact algorithms contain steps or sub-problems which can be solved heuristically. A secondary contribution of this work is to compare the proposed exact methods with three simple heuristic approaches based on the EM algorithm through computational experiments.

4
Exact Solution Approaches

This chapter introduces mathematical programming formulations for the MLE problem given by Equation (3-2). We first introduce a descriptive formulation based on a mixed integer non-linear programming (MINLP) model. Different techniques are then employed to linearize the model, leading to a mixed integer linear programming (MILP) formulation. To further improve computational efficiency, we discuss the use of bound-tightening and symmetry-breaking techniques.

4.1
Descriptive Formulation

We introduce the binary variable z_{ir}, which takes value 1 if vertex $i \in V$ is assigned to community $r \in \mathcal{C}$ and 0 otherwise, where $\mathcal{C} = \{1, \ldots, K\}$ is the set of possible communities. The continuous variables ω_{rs}, for $r, s \in \mathcal{C}$, represent the elements of the connectivity matrix $\mathbf{\Omega}$. The MLE problem (3-2) can be modeled as the following mixed integer non-linear program (MINLP), where we minimize the negative log-likelihood:

$$\underset{\mathbf{Z},\mathbf{\Omega}}{\text{minimize}} \quad \frac{1}{2}\sum_{i,j}^{n}\sum_{r,s}^{K} f_{ij}(\omega_{rs})\, z_{ir} z_{js} \tag{4-1}$$

$$\text{subject to} \quad \sum_{r=1}^{K} z_{ir} = 1, \qquad \forall i \in V \tag{4-2}$$

$$z_{ir} \in \{0,1\}, \qquad \forall i \in V, r \in \mathcal{C} \tag{4-3}$$

$$\omega_{rs} \in \mathbb{R}^{+}, \qquad \forall r, s \in \mathcal{C} \tag{4-4}$$

where

$$f_{ij}(\omega_{rs}) = -A_{ij} \log \omega_{rs} + \tfrac{k_i k_j}{2m}\omega_{rs}. \tag{4-5}$$

The objective function represents the expression for the MLE problem and constraints (4-2) enforce that each vertex is assigned to exactly one community. This model can be solved to optimality by global optimization solvers, such as `Couenne` [63], for small-sized networks. However, solution time quickly increases with the size of the networks, to such an extent that this solution approach becomes impractical for networks with as few as 16 vertices. In the next sections we propose some techniques to linearize Model (4-1)-(4-4),

resulting in a mixed integer linear program (MILP), which is empirically shown to speed-up solution time of the MINLP.

4.2 Mixed Integer Linear Programming Formulation

The linearization of the MINLP formulation is performed in a few steps. We first consider the non-linear function $f_{ij}(\omega_{rs})$ and propose a linearization by piecewise outer-approximation. For any point $\omega \in \mathbb{R}^+$ in the domain, the function $f_{ij}(\omega)$ can be approximated by its tangent $a_{ij\tilde{\omega}}\,\omega + b_{ij\tilde{\omega}}$, calculated at $\tilde{\omega}$, where the coefficients $a_{ij\tilde{\omega}}$ and $b_{ij\tilde{\omega}}$ are given by:

$$a_{ij\tilde{\omega}} = -\frac{A_{ij}}{\tilde{\omega}} + \frac{k_i k_j}{2m}, \tag{4-6}$$

$$b_{ij\tilde{\omega}} = f_{ij}(\tilde{\omega}) - a_{ij\tilde{\omega}}\,\tilde{\omega} = A_{ij}(1 - \log\tilde{\omega}). \tag{4-7}$$

The function f_{ij} is convex everywhere in its domain, for $A_{ij} > 0$, since:

$$\frac{\partial^2 f_{ij}}{\partial \omega^2} = \frac{A_{ij}}{\omega^2} > 0, \qquad \forall \omega \in \mathbb{R}^+. \tag{4-8}$$

Hence, the value of f_{ij} is always greater than or equal to its tangent at any point:

$$f_{ij}(\omega) \geq a_{ij\tilde{\omega}}\,\omega + b_{ij\tilde{\omega}}, \qquad \forall \omega \in \mathbb{R}^+,\, \forall \tilde{\omega} \in \mathbb{R}^+. \tag{4-9}$$

By making use of this property, we introduce variables f_{ijrs} to represent the value of $f_{ij}(\omega_{rs})$, and reformulate Model (4-1)-(4-4) in an equivalent form:

$$\underset{\mathbf{Z},\mathbf{\Omega},\mathbf{F}}{\text{minimize}} \quad \frac{1}{2}\sum_{i,j}^{n}\sum_{r,s}^{K} f_{ijrs}\, z_{ir} z_{js} \tag{4-10}$$

$$\text{subject to} \quad \sum_{r=1}^{K} z_{ir} = 1, \qquad \forall i \in V \tag{4-11}$$

$$f_{ijrs} \geq a_{ij\tilde{\omega}}\,\omega_{rs} + b_{ij\tilde{\omega}}, \qquad \forall i,j \in V, \forall r,s \in \mathcal{C}, \forall \tilde{\omega} \in \mathbb{R}^+ \tag{4-12}$$

$$z_{ir} \in \{0,1\}, \qquad \forall i \in V, \forall r \in \mathcal{C} \tag{4-13}$$

$$\omega_{rs} \in \mathbb{R}^+, \qquad \forall r,s \in \mathcal{C} \tag{4-14}$$

$$f_{ijrs} \in \mathbb{R}, \qquad \forall i,j \in V, \forall r,s \in \mathcal{C}. \tag{4-15}$$

This model contains an infinite number of constraints of type (4-12), for every $\tilde{\omega} \in \mathbb{R}^+$, which are necessary for the equivalence to hold.

The objective function is still cubic in the variables and is linearized next. Let y_{ijrs} denote the product of the binary variables z_{ir} and z_{js} in the objective function, $y_{ijrs} := z_{ir} z_{js}$. The product of two binary variables can be expressed

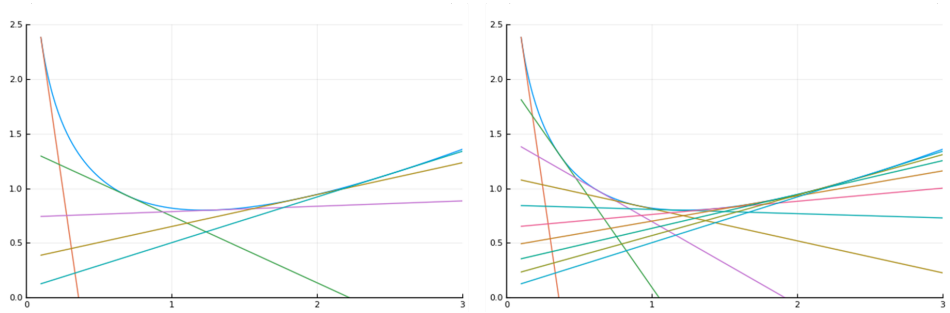


Figure 4.1: Piecewise linear outer-approximation of the function f_{ij} with 5 (left) and 10 (right) equally spaced breakpoints

as a set of linear constraints:

$$z_{ir} - y_{ijrs} \geq 0, \tag{4-16}$$

$$z_{js} - y_{ijrs} \geq 0, \tag{4-17}$$

$$1 - z_{ir} - z_{js} + y_{ijrs} \geq 0. \tag{4-18}$$

As a result, the objective function is then expressed as $f_{ijrs}y_{ijrs}$ which is a product of a continuous and a binary variable.

To linearize the expression $f_{ijrs}\, y_{ijrs}$ we introduce continuous variables $x_{ijrs} := f_{ijrs}\, y_{ijrs} = f_{ijrs}\, z_{ir} z_{js}$. The non-linear expression $f_{ijrs}\, y_{ijrs}$ can be linearized with the big-M technique, by introducing additional constraints:

$$x_{ijrs} \leq \overline{M}\, y_{ijrs}, \tag{4-19}$$

$$x_{ijrs} \geq \underline{M}\, y_{ijrs}, \tag{4-20}$$

$$x_{ijrs} \geq f_{ijrs} - \overline{M}(1 - y_{ijrs}). \tag{4-21}$$

Constraints (4-21) can be combined with constraints (4-12), yielding:

$$x_{ijrs} \geq a_{ij\tilde{\omega}}\, \omega_{rs} + b_{ij\tilde{\omega}} - \overline{M}(1 - y_{ijrs}), \qquad \forall \tilde{\omega} \in \mathbb{R}^{+}. \tag{4-22}$$

It is well known that formulations with big-M constants tend to suffer from a weak continuous relaxation [58, 64], if the values for the lower and upper bounds, $\underline{M}$ and $\overline{M}$, are not tight enough. Section 4.2.2 proposes some natural values for these bounds. The resulting model is a mixed integer linear program (MILP), which can be solved by the branch-and-bound method. The complete formulation of the MILP is available in Appendix B. The next sections address some challenges when solving the MILP in practice.

4.2.1 Dynamic Constraints Generation

As mentioned previously, the MILP model has an infinite number of constraints of the type (4-22). To solve it in practice, the method only considers a small finite set of these constraints, given by a set of break-points $\widetilde{\omega}_p$, indexed by $p \in \mathcal{B}$. New constraints are dynamically introduced in the model during the solution process. In practice this can be done by making use of the *lazy constraints callback*, which is available in standard mixed integer programming (MIP) solvers, such as `Gurobi` [65] and `CPLEX` [66]. Whenever an integer feasible solution is found during the branch-and-bound process, the callback checks if any of the constraints given by (4-22) are violated. If the solution violates any of these constraints, the solution is discarded and the violated constraints are included in the model. In effect the method iteratively refines the approximation of the function f_{ij} until a desired pre-defined precision is achieved.

4.2.2 Bounds Tightening

We start this section by deriving upper and lower bounds on ω_{rs}. Given a fixed assignment of vertices to communities, the optimal value of ω_{rs} can be found by solving a convex minimization problem by differentiation. We show that ω_{rs} is bounded above by $2m\rho$ (the value of ρ will be defined shortly).

$$\omega^*_{rs} = 2m \left(\frac{\sum_{i,j} A_{ij} z_{ir} z_{js}}{\sum_{i,j} k_i k_j z_{ir} z_{js}} \right) \leq 2m\rho \tag{4-23}$$

Inequality (4-23) is equivalent to:

$$\sum_{i,j} \left(\tfrac{1}{\rho} A_{ij} - k_i k_j \right) z_{ir} z_{js} \leq 0 \tag{4-24}$$

which is satisfied if:

$$\rho \geq \frac{A_{ij}}{k_i k_j}, \qquad \forall i, j \in V \tag{4-25}$$

Thus, by defining $\rho := \max_{i,j} \left\{ \frac{A_{ij}}{k_i k_j} \right\}$ we arrive at a valid upper bound:

$$\omega_{rs} \leq 2m \cdot \max_{i,j} \left\{ \frac{A_{ij}}{k_i k_j} \right\} \tag{4-26}$$

Let $\omega^L_{rs} \approx 0$ and ω^U_{rs} denote the lower and upper bounds, respectively, on ω_{rs}. We rely on these bounds to derive bounds $\underline{M_{ijrs}} \leq f_{ijrs} \leq \overline{M_{ijrs}}$.

Recall that $f_{ij}(\omega_{rs}) = (-A_{ij} \log \omega_{rs} + \frac{k_i k_j}{2m} \omega_{rs})$. If $A_{ij} = 0$, then the expression simplifies to:

$$f_{ij}(\omega_{rs}) = \tfrac{k_i k_j}{2m} \omega_{rs} \tag{4-27}$$

and therefore f_{ijrs} can be bounded by $0 \leq f_{ijrs} \leq \frac{k_i k_j}{2m}\omega_{rs}^U$.

Otherwise, $A_{ij} \neq 0$, then a lower bound can be obtained by calculating the global minimum of $f_{ij}(\omega_{rs})$ with respect to ω_{rs}. The minimum can be found by solving

$$\frac{\partial f_{ij}}{\partial \omega_{rs}} = -\frac{A_{ij}}{\omega_{rs}} + \frac{k_i k_j}{2m} = 0, \tag{4-28}$$

implying

$$\hat{\omega}_{rs} = \frac{2mA_{ij}}{k_i k_j}, \tag{4-29}$$

and therefore

$$\underline{M_{ijrs}} = -A_{ij} \log \tfrac{2mA_{ij}}{k_i k_j} + A_{ij} = A_{ij}\Big(1 - \log A_{ij} + \log \tfrac{k_i k_j}{2m}\Big). \tag{4-30}$$

Since $f_{ij}(\omega_{rs})$ is convex, the upper bound $\overline{M_{ijrs}}$ can be defined by calculating the function value at the extreme points of the domain $[\omega_{rs}^L, \omega_{rs}^U]$:

$$\overline{M_{ijrs}} = \max\{f_{ij}(\omega_{rs}^L), f_{ij}(\omega_{rs}^U)\}. \tag{4-31}$$

Overall, the upper and lower bounds are given by:

$$\overline{M_{ijrs}} := \begin{cases} \frac{k_i k_j}{2m}\omega_{rs}^U, & \text{if } A_{ij} = 0 \\ \max\{f_{ij}(\omega_{rs}^L), f_{ij}(\omega_{rs}^U)\}, & \text{if } A_{ij} \neq 0 \end{cases} \tag{4-32}$$

$$\underline{M_{ijrs}} := \begin{cases} 0, & \text{if } A_{ij} = 0 \\ A_{ij}(1 - \log A_{ij} + \log \frac{k_i k_j}{2m}), & \text{if } A_{ij} \neq 0 \end{cases} \tag{4-33}$$

In the experiments ω_{rs}^L is set to a small constant value, since the function f_{ij} is not defined for $\omega_{rs} = 0$.

4.2.3 Symmetry-breaking Constraints

In the formulations discussed above, any permutation of the group indices in the community assignment variables $\mathbf{Z}$ leads to an equivalent solution. Thus, for each solution there are always $K!$ equivalent ones, causing an inefficient exploration during the branch-and-bound process. To circumvent this issue, Frank Plastria (2002) [67] proposed a set a linear constraints that limit the set of feasible solutions by eliminating solutions which are equivalent. This is done by enforcing the model to accept only *lexicographically minimal solutions*, i.e. by forcing community r to always contain the lowest numbered object (vertex) which does not belong to any of the previous communities $1, \ldots, r-1$. As shown by [67], this can be achieved by including the following constraints to the model:

$$z_{11} = 1 \tag{4-34}$$

$$\sum_{i=2}^{j-1}\sum_{l=1}^{r-1} z_{il} - \sum_{l=1}^{r} z_{jl} \leq j-3, \qquad \forall r \in \{2,\ldots,K-1\}, \forall j \in \{r,\ldots,n\} \quad (4\text{-}35)$$

The last cluster K is not associated to any constraint, as it will automatically contain all remaining objects which do not belong to any of the previous clusters. These constraints effectively break the permutation symmetry of the problem and are shown to provide speed-ups to the solution method in the computational experiments.

5
Heuristic Solution Approaches

Heuristic methods usually solve a problem in reduced computing time when compared to exact methods, at the expense of having no guarantee of optimality of the solution. The original method proposed by Karrer and Newman [1] for community detection using the DCSBM is based on a local search heuristic on the space of community assignments. Inspired by their approach we introduce three natural variants of the expectation-maximization (EM) algorithm, and compare them to the exact methods of the previous chapter. In the next sections we review the main aspects of the general EM algorithm and describe the three variants proposed for this problem.

5.1
Expectation-Maximization Algorithm

The expectation-maximization (EM) algorithm was introduced by Dempster *et al.* in 1977 [68] as a general iterative scheme for finding the parameter estimates of maximum likelihood or maximum a posteriori probability (MAP) of statistical models with (unobserved) latent variables. Each iteration of the algorithm alternates between an expectation step (E-step) and a maximization step (M-step). Each step is guaranteed to increase the log-likelihood function, and therefore it always finds a (local) optimum. The EM algorithm has been applied to a variety of models in machine learning including Gaussian mixture models (GMM) and hidden Markov models (HMM) [69, 70], and in data clustering [71, 72].

The EM algorithm is a heuristic approach since there is no guarantee that the method converges to the global optimum of maximum likelihood. An instantiation of the EM algorithm for the MLE problem of the DCSBM is described in Algorithm 1. The next sections describe each step of the algorithm in detail. For the E-step three variants are introduced. In the first variant (`E-LS1`) the algorithm searches for community assignments $\mathbf{Z}$ that increase the value of the likelihood function via a simple local search heuristic. The second variant (`E-LS2`) similarly performs a local search on the assignments $\mathbf{Z}$, however the connectivity matrix $\boldsymbol{\Omega}$ is re-estimated at every evaluation of the likelihood function, and thus this variant is more tightly integrated with

the M-step. Finally, the third variant (`E-exact`) is based on a formulation of the E-step as an integer quadratic program, which can be solved exactly using MIP optimization solvers.

Algorithm 1: EM algorithm

```
(Randomly) initialize assignments of vertices to communities Z;
repeat
    (M-step) Estimate the connectivity matrix Ω by maximizing the
     likelihood, using the current assignments Z;
    (E-step) Search for assignments Z which maximize the
     likelihood for the current estimate Ω (from M-step);
until The likelihood function can no longer be improved by these
 two steps;
```

5.1.1 Maximization Step (M-step)

The maximization step consists in estimating the parameters $\mathbf{\Omega}$ which maximize the likelihood function, given a fixed assignment of vertices to communities $\mathbf{Z}$. The optimal value for ω_{rs} can be found by solving a concave maximization problem, by differentiating Equation (4-1). The solution can be written in a closed-form expression:

$$\omega^*_{rs} = 2m \left(\frac{\sum_{i,j} A_{ij} z_{ir} z_{js}}{\sum_{i,j} k_i k_j z_{ir} z_{js}} \right) = \frac{2m \cdot m_{rs}}{\kappa_r \kappa_s} \tag{5-1}$$

where $m_{rs} = \sum_{i,j}^{n} A_{ij} z_{ir} z_{js}$ is the number of edges between groups r and s, and $\kappa_r = \sum_{i}^{n} k_i z_{ir}$ is the sum of the degrees of the vertices in group r. This is a polynomial-time step, since calculating ω_{rs} for all $r, s \in \{1, \ldots, K\}$ has time complexity $O(K^2 n^2)$.

5.1.2 Expectation Step (E-step)

The expectation step consists in searching for community assignments $\mathbf{Z}$ that maximize the likelihood, given the current estimate for the affinity matrix $\mathbf{\Omega}$. This step corresponds to an $\mathcal{NP}$-hard combinatorial problem [44], which involves optimizing over all possible community assignments. Three variants for the E-step are discussed next which, combined with the M-step, result in three variants for the EM algorithm.

Local search on the community assignment variables (`E-LS1`) The first variant is based on a local search method where the value of $\mathbf{\Omega}$ remains fixed

and we search for community assignments $\mathbf{Z}$ that maximize the likelihood function. We do this by iterating over the vertices of the graph and reassigning a vertex to a different community whenever the relocation entails an improvement in the likelihood function. Namely, for each vertex i and for each community r, we consider relocating vertex i to community r. If the relocation results in an improvement of the likelihood function, we keep this assignment. The procedure is repeated until no more improving relocation can be identified.

Local search integrated with M-step (`E-LS2`) In this variant the value of $\mathbf{\Omega}$ is re-estimated (with the M-step) every time a relocation of a vertex is considered, and therefore the expectation and maximization steps become more tightly integrated. Algorithm 2 describes both `E-LS1` and `E-LS2` variants in a single pseudo-code. The for loop in the algorithm has complexity $O(K^3n^3)$, since each evaluation of the likelihood function requires $O(K^2n^2)$ elementary operations.

Algorithm 2: Expectation step (`E-LS1` and `E-LS2`)

```
1  Function ExpectationStep(Ω, Z):
2      L ← log P(A|Ω, Z);
3      repeat
4          for each vertex i ∈ V and community r ∈ C do
5              Consider Z' constructed from Z by relocating vertex i to
                community r;
6              (E-LS1) Ω' ← Ω;
7              (E-LS2) Ω' ← MaximizationStep(Z');
8              L' ← log P(A|Ω', Z');
9              if L' > L then
10                 Apply relocation and update solution:
                    Ω ← Ω'; Z ← Z'; L ← L';
11             end
12         end
13     until No improving relocation can be found;
14     return Z;
```

Exact community assignments (`E-exact`) The expectation step can be solved exactly, by formulating it as an integer quadratic program (IQP) as

follows:

$$\underset{\mathbf{Z},\boldsymbol{\Omega}}{\text{minimize}} \quad \frac{1}{2}\sum_{i,j}^{n}\sum_{r,s}^{K} f_{ij}(\omega_{rs})\, z_{ir} z_{js} \tag{5-2}$$

$$\text{subject to} \quad \sum_{r=1}^{q} z_{ir} = 1, \qquad \forall i \in V \tag{5-3}$$

$$z_{ir} \in \{0,1\}, \qquad \forall i \in V, r \in \mathcal{C} \tag{5-4}$$

In this model, we search for the community assignments $\mathbf{Z}$ which maximize the likelihood, for a fixed connectivity matrix $\boldsymbol{\Omega}$. Since ω_{rs} is fixed, the term $f_{ij}(\omega_{rs})$ in the objective function is constant. This is an $\mathcal{NP}$-hard problem, however we can solve it to optimality using standard MIP solvers, such as `Gurobi` [65] and `CPLEX` [66]. The standard solution approach employed by the solvers is based on the branch-and-cut method. The EM algorithm that results from this variant is an example of a *matheuristic* [73, 74], which combines the use of metaheuristics with mathematical programming techniques.

6
Computational Experiments and Analyses

This chapter discusses the computational experiments carried out to evaluate the proposed methods and to assess their performance. In the experiments we evaluate the two exact solution approaches, the MINLP (from Section 4.1) and the MILP (from Section 4.2), and compare them with the three heuristic approaches based on the EM algorithm, described in Chapter 5. The main goals of the computational experiments are twofold.

1. We compare the performance of the proposed exact methods in terms of solution time.

2. We evaluate how far the heuristic solutions given by the three variants of the EM algorithm are from the true optimum of maximum likelihood, by comparing them to the optimal solution and bounds found by the exact methods.

The computational experiments were performed on an Intel Xeon E5-2620 2.1 GHz processor machine with 128 GB of RAM memory and CentOS Linux 7 (Core) operating system. The high-level programming language used in the implementation was `Julia` [75], and the package `JuMP` [76] was used as the modeling language for the exact methods. The underlying optimization solvers adopted for the exact methods were `Couenne` [63] as the global optimization solver for the MINLP and `CPLEX` [66] for the MILP.

6.1
Instances

Two data sets were generated for the computational experiments, denoted S1 and S2. Both are composed of synthetic graphs produced by the DCSBM. Synthetic graphs are useful for benchmarking since they allow us to control the factors that might influence the difficulty of community detection, such as network size and different types of community structure. We restricted our experiments to small undirected networks, with a number of vertices n ranging from 8 to 16 and a number of edges m ranging from 4 to 115, for which the exact methods can find the global optimum in a reasonable time.

The first data set S1 considers the simplest case when $K = 2$ communities. We generate synthetic graphs with varying sizes: $n \in \{8, 10, 12, 14, 16\}$. Since $K = 2$, the affinity matrix $\mathbf{\Omega}$ of the generative model has three parameters: two diagonal elements ω_{11}, ω_{22} and one off-diagonal element $\omega_{12} = \omega_{21}$. We sample values for each of the three parameters from intervals centered at ω_{in} and ω_{out}, respectively for the diagonal and off-diagonal parameters, in the following way. For each $\omega_{\text{in}}, \omega_{\text{out}} \in \{0.1, 0.4, 0.6, 0.9\}$, such that $\omega_{\text{in}} \neq \omega_{\text{out}}$, we sample ω_{11}, ω_{22} from $\mathcal{U}(\omega_{\text{in}} - 0.1, \omega_{\text{in}} + 0.1)$ and we sample ω_{12} from $\mathcal{U}(\omega_{\text{out}} - 0.1, \omega_{\text{out}} + 0.1)$, where $\mathcal{U}(a, b)$ represents the uniform distribution in the interval $[a, b] \subseteq \mathbb{R}$. This results in a total of $4 \times 3 = 12$ combinations of values for the interval centers $(\omega_{\text{in}}, \omega_{\text{out}})$. Out of the 12 combinations there are six assortative and six disassortative structures, as shown in Table 6.1. To account for variability due to randomness in the generative model, 10 instances are generated for each combination, yielding a total of $5 \times 4 \times 3 \times 10 = 600$ network instances in the data set.

Number of vertices	ω_{in}	ω_{out}	Structure
$n \in \{8, 10, 12, 14, 16\}$	0.1	0.4	disassortative
		0.6	disassortative
		0.9	disassortative
	0.4	0.1	assortative
		0.6	disassortative
		0.9	disassortative
	0.6	0.1	assortative
		0.4	assortative
		0.9	disassortative
	0.9	0.1	assortative
		0.4	assortative
		0.6	assortative

Table 6.1: Combinations of values of the interval centers $(\omega_{\text{in}}, \omega_{\text{out}})$ that are used in the generation of the synthetic data set S1.

The second data set S2 is composed of synthetic graphs generated by the DCSBM with the number of clusters $K \in \{2, 3\}$ and three levels of assortative community strength: low, medium and high. For each level of community strength the diagonal and off-diagonal elements of the affinity matrix $\mathbf{\Omega}$ are drawn from a uniform distribution in the corresponding interval given by Table 6.2. All instances in data set S2 are generated by strongly assortative models. The number of vertices again varies with $n \in \{8, 10, 12, 14, 16\}$. A total of 20 instances was generated for each configuration, totalling $2 \times 3 \times 5 \times 20 = 600$ instances. Table 6.3 summarizes the defining features of data set S2.

	LOW	MEDIUM	HIGH
ω_{rr}	[0.4, 1.0]	[0.6, 1.0]	[0.8, 1.0]
$\omega_{rs}\,(r \neq s)$	[0.2, 0.4]	[0.1, 0.3]	[0.0, 0.2]

Table 6.2: The level of community strength (low, medium or high) defines the ranges of possible values for the diagonal and off-diagonal elements of the affinity matrix $\mathbf{\Omega}$.

Number of communities	Number of vertices	Community strength
$K \in \{2, 3\}$	$n \in \{8, 10, 12, 14, 16\}$	LOW
		MEDIUM
		HIGH

Table 6.3: Summary of the defining features of data set S2.

6.2 Performance of the exact methods

For each instance in data sets S1 and S2 we run the two exact methods (MINLP and MILP) with a time limit of 600 seconds. To assess the impact of the proposed symmetry-breaking constraints in the solution time, we run each method twice: once with symmetry-breaking constraints (SBC) and once without symmetry-breaking constraints (N-SBC).

The solution times for the exact methods are presented in Table 6.4 for data set S1 and in Table 6.5 for data set S2. The values reported in Table 6.4 correspond to average solution times (in seconds) over 10 instances, for each $n \in \{8, 10, 12, 14, 16\}$ and for each combination of $(\omega_{\text{in}}, \omega_{\text{out}})$. The results reported in Table 6.5 are average solution times (in seconds) over 20 instances, for each $K \in \{2, 3\}, n \in \{8, 10, 12, 14, 16\}$ and for each level of community strength. The values reported in parentheses indicate the number of instances for which the method could not find the optimal solution within the time limit.

As visible in these experiments, the MILP is notably faster than the MINLP, for both data sets. When $K = 2$, the MILP is able to arrive at the optimal solution of maximum likelihood for all instances up to $n = 14$, while the MINLP is already unable to find the optimum for a few instances with 14 vertices. With $K = 2$ and $n = 16$, both methods struggle to find the optimal solution within the time limit. When $K = 3$, we are able to arrive at optimal solutions up to 12 nodes, using the MILP with SBC (except for a few instances). When $K = 3$ and $n = 16$, none of the methods is able to find the optimal solution for any instance.

The use of symmetry-breaking constraints has a significant impact on solution time for both formulations. Figure 6.1 illustrates the impact of using symmetry-breaking constraints as a function of the number of vertices, for data

			MILP		**MINLP**	
n	ω_{in}	ω_{out}	N-SBC	SBC	N-SBC	SBC
8	0.1	0.4	0.8	**0.6**	4.2	3.3
		0.6	1.2	**0.8**	6.4	5.0
		0.9	1.2	**0.8**	6.9	5.3
	0.4	0.1	0.9	**0.7**	4.4	3.5
		0.6	2.0	**1.2**	9.0	7.2
		0.9	2.1	**1.3**	10.4	8.1
	0.6	0.1	1.1	**0.8**	5.6	4.6
		0.4	2.0	**1.2**	9.3	7.5
		0.9	2.8	**1.7**	12.5	10.1
	0.9	0.1	1.0	**0.8**	7.1	5.3
		0.4	2.4	**1.5**	11.1	8.8
		0.6	2.6	**1.6**	12.2	9.4
10	0.1	0.4	3.7	**2.6**	18.5	15.5
		0.6	3.6	**2.3**	18.4	12.9
		0.9	4.2	**2.6**	24.3	17.7
	0.4	0.1	3.1	**2.1**	15.0	12.4
		0.6	8.5	**5.4**	32.1	22.7
		0.9	9.0	**5.9**	35.6	25.1
	0.6	0.1	4.1	**2.8**	23.6	16.6
		0.4	7.1	**4.9**	29.2	23.2
		0.9	12.8	**7.9**	41.8	30.9
	0.9	0.1	4.6	**3.0**	25.2	19.1
		0.4	8.4	**5.5**	35.2	25.6
		0.6	12.1	**7.1**	41.0	29.9
12	0.1	0.4	11.3	**7.0**	56.4	37.8
		0.6	11.1	**7.7**	66.6	44.1
		0.9	13.6	**10.1**	83.3	56.6
	0.4	0.1	10.4	**7.0**	58.0	39.4
		0.6	37.8	**23.0**	130.4	77.5
		0.9	36.0	**21.6**	111.4	74.9
	0.6	0.1	14.5	**8.1**	79.6	44.8
		0.4	35.0	**19.4**	109.2	73.0
		0.9	49.4	**27.8**	150.3	91.2
	0.9	0.1	21.0	**12.1**	93.6	60.7
		0.4	33.5	**18.9**	120.8	79.4
		0.6	46.4	**26.9**	136.7	91.9
14	0.1	0.4	35.1	**22.1**	226.0	138.2
		0.6	46.8	**23.8**	310.8	162.4
		0.9	45.3	**29.2**	301.5	152.9
	0.4	0.1	58.2	**28.1**	277.0	162.4
		0.6	180.8	**98.0**	450.0	270.0
		0.9	217.5	**111.4**	469.2	271.9
	0.6	0.1	52.7	**29.0**	270.6	157.4
		0.4	152.0	**75.2**	(1) 465.9	266.2
		0.9	252.1	**130.3**	(5) 546.5	305.4
	0.9	0.1	62.8	**33.2**	339.8	186.4
		0.4	200.2	**115.6**	(2) 520.5	295.3
		0.6	236.0	**122.0**	(1) 542.5	328.1
16	0.1	0.4	262.8	**122.7**	(10) 652.7	(7) 591.9
		0.6	141.1	**88.5**	(9) 623.6	(6) 553.4
		0.9	165.7	**93.0**	(9) 633.2	(5) 565.2
	0.4	0.1	(1) 247.0	**140.8**	(8) 603.9	(5) 540.3
		0.6	(9) 579.9	(1) **471.0**	(10) 649.4	(10) 648.1
		0.9	(7) 528.6	(2) **416.9**	(10) 644.7	(10) 644.2
	0.6	0.1	(2) 282.0	(1) **170.1**	(8) 642.0	(4) 516.2
		0.4	(9) 599.7	(1) **417.5**	(10) 650.8	(9) 646.8
		0.9	(10) 600.0	(5) **561.4**	(10) 641.4	(10) 642.5
	0.9	0.1	(1) 177.3	**109.6**	(8) 625.8	(2) 484.4
		0.4	(9) 592.3	(2) **484.6**	(10) 642.0	(10) 643.8
		0.6	(10) 600.0	(6) **585.3**	(10) 638.6	(10) 639.7

Table 6.4: Solution times of the exact methods (MILP vs MINLP) for data set S1, with symmetry-breaking constraints (SBC) and without symmetry-breaking constraints (N-SBC).

K	n	Community strength	**MILP** N-SBC	**MILP** SBC	**MINLP** N-SBC	**MINLP** SBC
2	8	low	3.4	**2.0**	16.8	13.8
		medium	3.2	**1.9**	15.2	12.2
		high	2.4	**1.4**	12.5	11.1
	10	low	6.2	**4.0**	39.4	28.9
		medium	5.8	**3.9**	35.9	26.4
		high	3.9	**2.5**	33.0	27.6
	12	low	43.7	**17.1**	188.0	122.0
		medium	25.2	**11.4**	179.9	121.5
		high	16.2	**8.0**	138.5	93.2
	14	low	159.8	**89.9**	(15) 615.4	427.5
		medium	151.2	**82.1**	(8) 557.6	349.2
		high	78.9	**38.2**	(6) 489.7	330.1
	16	low	(16) 576.4	(10) **482.8**	(20) 645.9	(20) 647.7
		medium	(14) 565.6	(8) **455.3**	(20) 674.4	(20) 677.4
		high	(5) 326.3	**216.5**	(17) 639.0	(14) 591.3
3	8	low	30.5	**8.7**	195.5	70.4
		medium	31.1	**7.3**	150.5	58.9
		high	22.7	**5.9**	123.4	61.1
	10	low	190.0	**30.0**	(13) 562.7	238.7
		medium	109.9	**22.2**	(15) 590.2	251.0
		high	58.9	**12.0**	(8) 453.5	153.3
	12	low	(18) 584.1	**333.4**	(20) 658.4	(20) 657.0
		medium	(16) 532.9	(2) **316.7**	(20) 669.8	(20) 669.0
		high	(6) 398.6	**103.5**	(20) 644.0	(18) 624.2
	14	low	(20) 600.0	(20) 600.0	(20) 650.1	(20) 653.8
		medium	(20) 600.0	(20) 600.0	(20) 647.4	(20) 652.0
		high	(20) 600.0	(9) **502.8**	(20) 656.5	(20) 657.6
	16	low	(20) 600.0	(20) 600.0	(20) 639.2	(20) 640.3
		medium	(20) 600.0	(20) 600.0	(20) 615.8	(20) 615.6
		high	(20) 600.0	(20) 600.0	(20) 644.0	(20) 647.1

Table 6.5: Solution times of the exact methods (MILP vs MINLP) for data set S2, with symmetry-breaking constraints (SBC) and without symmetry-breaking constraints (N-SBC).

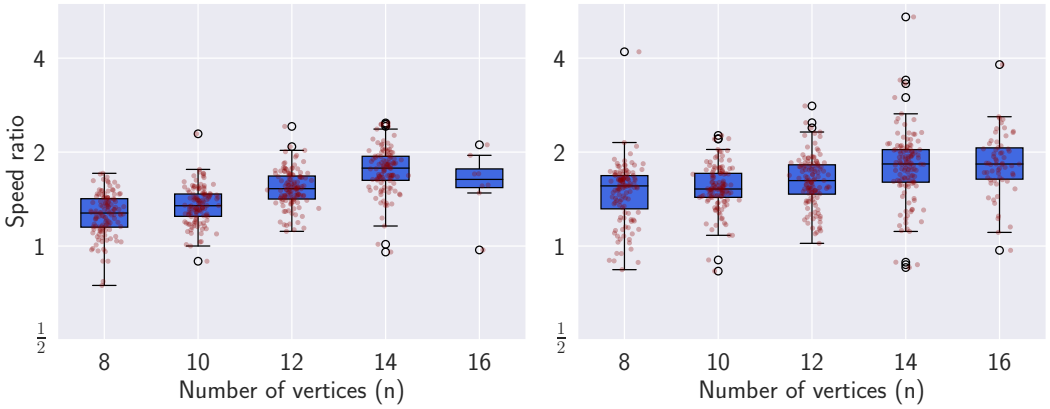


Figure 6.1: Impact of symmetry-breaking constraints on solution times for the MINLP (left) and the MILP (right) for data set S1. For each method the speed ratio is the solution time of the model *without* SBC, divided by the solution time of the model *with* SBC.

set S1. Instances for which either method could not find the optimal solution are not included in the plot. Even in this simple case with two communities, adding symmetry-breaking constraints consistently improves the solution time of both exact methods, for the great majority of instances. The improvement becomes slightly more evident as the number of vertices increases, with solution times up to almost 3x faster.

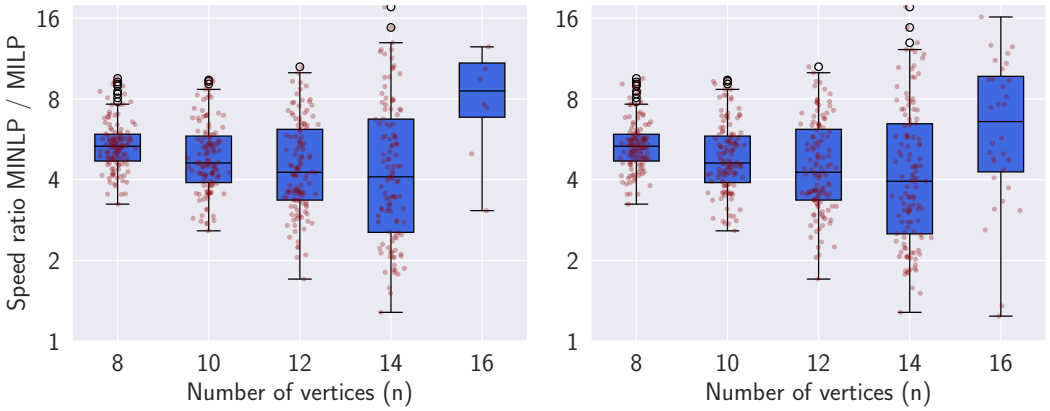


Figure 6.2: Solution time comparison of the MILP against the MINLP model, without (left) and with (right) SBC, for data set S1. Here the speed ratio is the solution time of the MINLP divided by the solution time of the MILP.

A comparison of the solution times of both exact methods is presented in Figure 6.2 for data set S1, as a function of the number of vertices. As n increases, the variance in the distribution of the speed ratio increases. The speed ratios are nonetheless consistently greater than 1, meaning that the method based on the MILP is consistently faster than the MINLP, regardless of whether symmetry-breaking constraints are included. The MILP improves

the solution time of the MINLP up to approximately 18x in the most extreme cases.

The MILP with symmetry-breaking constraints is clearly the formulation with the best performance. Figures 6.3 and 6.4 illustrate the solution times of the MILP with SBC for data sets S1 and S2 as a function of the features of each data set.

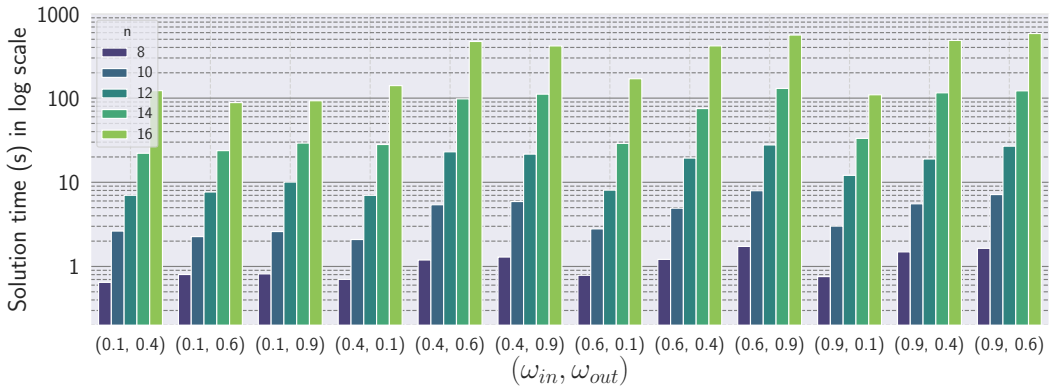


Figure 6.3: Solution times (in seconds, in log scale) of the MILP with SBC for data set S1 as a function of the number of vertices and the parameter values $(\omega_{\text{in}}, \omega_{\text{out}})$.

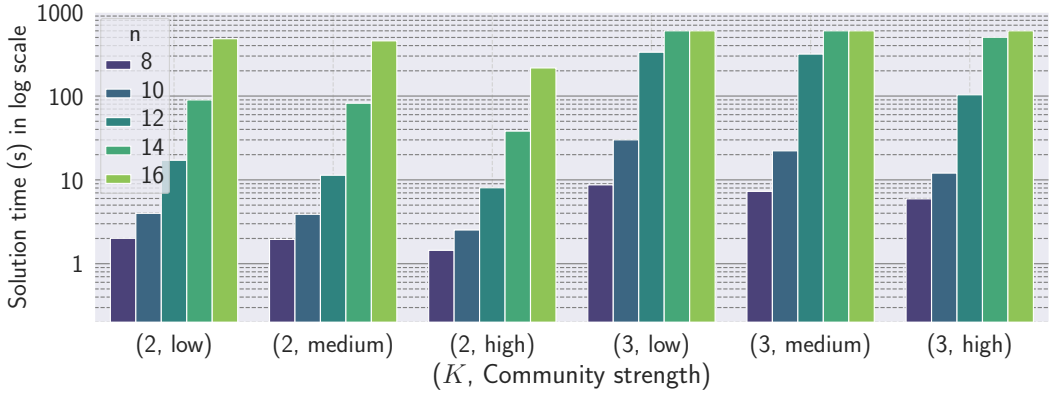


Figure 6.4: Solution times (in seconds, in log scale) of the MILP with SBC for data set S2 as a function of the number of vertices n, the number of communities K and the level of community strength.

6.3 Performance of the heuristic methods

To evaluate the quality of the solutions found by the heuristic approaches we compare their solution to the solution found by the exact methods. For each instance in data sets S1 and S2, we run the three EM variants for 50 trials (with different random starts). For each instance the relative percentage gap is calculated as:

$$\text{Gap}(\%) = \frac{\text{OBJ} - \text{UB}}{\text{UB}} \tag{6-1}$$

where OBJ is the objective value of the heuristic solution and UB (upper bound) is the objective value of the best integer feasible solution found by the MILP with SBC.

For data set S1, Table 6.6 presents the relative percentage gap and solution time (in seconds) of the best heuristic solution (out of a total of 50 trials) for each algorithm.

For almost all instances in data set S1 the heuristic approaches are capable of finding the optimal solution. Note that when $n = 16$ and $(\omega_{\text{in}}, \omega_{\text{out}}) = (0.9, 0.6)$ the reported gap is negative, which means that the heuristic objective value is better than the one found by the exact method (the exact methods are not able to find all optimal solutions within the time limit in this case).

Table 6.7 shows similar results for data set S2. When $K = 3$ and $n = 16$, all percentage gap values are negative indicating that the heuristic approaches outperform the exact methods in this setting.

To illustrate how often (out of 50 trials) the optimal solution is found, Figure 6.5 shows the (normalized) frequency with which the three EM variants arrive at the optimal solution, for data set S1.

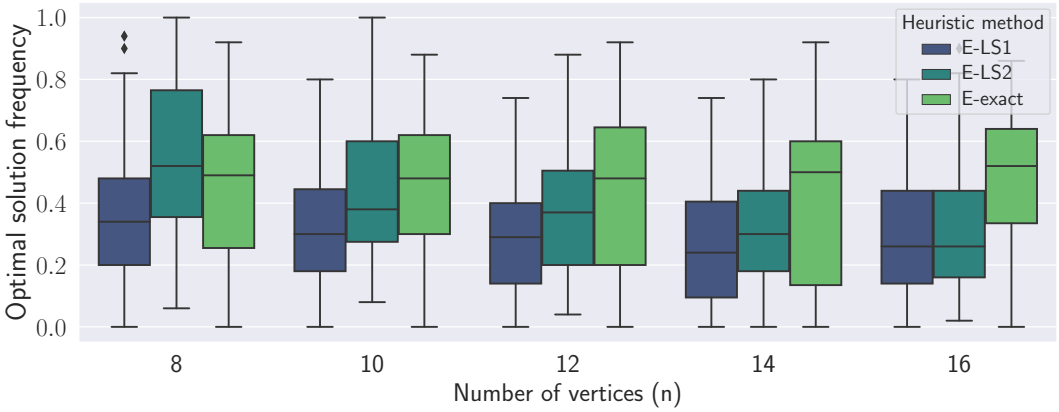


Figure 6.5: Normalized frequency of finding the optimal solution (in 50 trials) for each of the three EM variants.

This frequency could be interpreted as an empirical probability that the heuristic method finds the optimal solution. There is a large variance and the specific frequency value is highly dependent on the instance features. However the frequency is clearly greater than zero for almost all instances, which corroborates with the results presented in Table 6.6, showing that the heuristics can find the optimum of maximum likelihood with considerable probability.

We also analyze the average percentage gap (out of 50 trials): Figure 6.6 illustrates for data set S1 the average objective value for each heuristic method,

			E-LS1		E-LS2		E-exact	
n	ω_{in}	ω_{out}	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
8	0.1	0.4	0.0	1.9E-04	0.0	2.4E-04	0.0	3.3E-01
		0.6	0.0	1.8E-04	0.0	2.3E-04	0.0	4.2E-01
		0.9	0.0	1.9E-04	0.0	2.6E-04	0.0	3.2E-01
	0.4	0.1	0.0	1.4E-04	0.0	2.0E-04	0.0	4.2E-01
		0.6	0.0	1.6E-04	0.0	2.3E-04	0.0	4.7E-01
		0.9	0.0	1.7E-04	0.0	2.6E-04	0.0	5.8E-01
	0.6	0.1	0.0	1.2E-04	0.0	1.9E-04	0.0	4.7E-01
		0.4	2.7E-03	1.7E-04	0.0	2.2E-04	2.7E-03	6.0E-01
		0.9	0.0	1.3E-04	0.0	2.4E-04	0.0	5.0E-01
	0.9	0.1	0.0	1.3E-04	0.0	2.5E-04	0.0	4.1E-01
		0.4	0.0	1.4E-04	0.0	2.5E-04	2.5E-04	5.5E-01
		0.6	0.0	1.8E-04	0.0	2.2E-04	0.0	6.1E-01
10	0.1	0.4	0.0	2.8E-04	0.0	5.1E-04	0.0	5.4E-01
		0.6	0.0	3.0E-04	0.0	4.1E-04	0.0	6.6E-01
		0.9	0.0	2.7E-04	0.0	4.7E-04	0.0	5.4E-01
	0.4	0.1	0.0	2.9E-04	0.0	3.6E-04	0.0	7.3E-01
		0.6	0.0	3.0E-04	0.0	4.5E-04	0.0	7.9E-01
		0.9	9.8E-04	3.5E-04	0.0	4.8E-04	0.0	7.5E-01
	0.6	0.1	0.0	3.1E-04	0.0	4.0E-04	7.6E-03	6.9E-01
		0.4	0.0	3.0E-04	0.0	3.9E-04	0.0	8.7E-01
		0.9	0.0	3.4E-04	0.0	4.3E-04	0.0	9.7E-01
	0.9	0.1	0.0	3.2E-04	0.0	4.1E-04	0.0	7.7E-01
		0.4	1.5E-03	4.2E-04	0.0	5.4E-04	3.0E-04	9.0E-01
		0.6	5.5E-04	2.6E-04	0.0	4.2E-04	0.0	9.3E-01
12	0.1	0.4	0.0	5.1E-04	0.0	8.3E-04	0.0	9.5E-01
		0.6	0.0	4.8E-04	0.0	8.8E-04	0.0	8.3E-01
		0.9	0.0	5.4E-04	0.0	7.2E-04	0.0	8.7E-01
	0.4	0.1	1.0E-03	4.3E-04	0.0	6.2E-04	0.0	1.1E+00
		0.6	2.9E-04	6.0E-04	0.0	9.2E-04	1.4E-03	1.3E+00
		0.9	5.9E-03	6.2E-04	0.0	7.8E-04	1.2E-03	1.2E+00
	0.6	0.1	0.0	5.1E-04	0.0	8.3E-04	8.0E-04	1.1E+00
		0.4	0.0	4.4E-04	0.0	6.3E-04	0.0	1.3E+00
		0.9	2.8E-03	6.3E-04	0.0	1.0E-03	1.2E-03	1.1E+00
	0.9	0.1	0.0	5.8E-04	0.0	8.2E-04	0.0	1.1E+00
		0.4	1.5E-03	6.3E-04	0.0	8.1E-04	1.5E-03	1.2E+00
		0.6	5.9E-04	6.0E-04	0.0	8.3E-04	7.1E-04	1.3E+00
14	0.1	0.4	0.0	8.2E-04	0.0	1.3E-03	0.0	1.4E+00
		0.6	3.6E-03	8.7E-04	0.0	1.3E-03	0.0	1.2E+00
		0.9	0.0	9.9E-04	0.0	1.4E-03	0.0	1.8E+00
	0.4	0.1	0.0	7.4E-04	0.0	1.3E-03	0.0	1.9E+00
		0.6	8.6E-03	8.6E-04	0.0	1.2E-03	2.5E-03	1.6E+00
		0.9	4.7E-03	8.2E-04	0.0	1.3E-03	2.3E-03	2.3E+00
	0.6	0.1	0.0	7.8E-04	0.0	1.3E-03	0.0	4.7E+00
		0.4	1.2E-03	6.4E-04	0.0	1.2E-03	1.2E-03	6.9E+00
		0.9	3.5E-03	9.0E-04	0.0	1.6E-03	1.9E-03	4.5E+00
	0.9	0.1	0.0	7.5E-04	0.0	1.4E-03	0.0	1.5E+00
		0.4	4.7E-04	8.0E-04	1.3E-04	1.2E-03	3.2E-04	1.8E+00
		0.6	1.8E-03	7.7E-04	0.0	1.4E-03	2.7E-03	1.9E+00
16	0.1	0.4	0.0	1.5E-03	0.0	2.0E-03	0.0	1.9E+00
		0.6	0.0	1.4E-03	0.0	2.1E-03	0.0	1.8E+00
		0.9	0.0	1.1E-03	0.0	2.0E-03	0.0	1.7E+00
	0.4	0.1	1.3E-03	1.0E-03	0.0	1.8E-03	1.3E-03	6.7E+00
		0.6	1.0E-03	1.5E-03	0.0	1.7E-03	0.0	8.2E+00
		0.9	4.0E-03	1.5E-03	0.0	2.4E-03	4.8E-03	7.6E+00
	0.6	0.1	1.5E-04	1.3E-03	0.0	1.7E-03	0.0	6.1E+00
		0.4	0.0	1.2E-03	0.0	2.0E-03	2.7E-04	2.6E+00
		0.9	4.9E-04	1.3E-03	0.0	1.7E-03	9.7E-04	2.9E+00
	0.9	0.1	0.0	1.2E-03	0.0	1.8E-03	0.0	2.1E+00
		0.4	0.0	1.2E-03	0.0	2.0E-03	0.0	2.7E+00
		0.6	-8.0E-04	1.1E-03	-1.2E-03	2.1E-03	-4.7E-04	2.9E+00

Table 6.6: Relative percentage gap and solution times of the heuristic methods for data set S1.

n	K	Community strength	E-LS1 Gap (%)	E-LS1 Time (s)	E-LS2 Gap (%)	E-LS2 Time (s)	E-exact Gap (%)	E-exact Time (s)
2	8	low	0.0	1.7E-04	0.0	2.0E-04	0.0	1.6E+00
		medium	0.0	2.0E-04	0.0	2.2E-04	6.9E-05	1.3E+00
		high	0.0	1.8E-04	0.0	2.2E-04	0.0	1.4E+00
	10	low	1.2E-03	3.2E-04	0.0	4.0E-04	0.0	5.0E+00
		medium	2.0E-03	3.0E-04	0.0	3.6E-04	1.8E-03	4.6E+00
		high	0.0	3.8E-04	0.0	4.6E-04	0.0	4.2E+00
	12	low	7.4E-04	5.3E-04	0.0	7.1E-04	1.3E-03	3.7E+00
		medium	0.0	4.8E-04	0.0	7.1E-04	1.2E-03	3.4E+00
		high	0.0	5.0E-04	0.0	6.7E-04	0.0	3.4E+00
	14	low	0.0	8.3E-04	0.0	1.1E-03	8.0E-04	2.3E+01
		medium	0.0	8.8E-04	0.0	1.3E-03	0.0	1.8E+01
		high	0.0	1.1E-03	0.0	1.5E-03	3.1E-04	2.1E+01
	16	low	1.8E-03	1.4E-03	0.0	1.8E-03	2.8E-03	2.7E+00
		medium	9.4E-04	1.4E-03	3.9E-04	2.0E-03	7.5E-04	2.3E+00
		high	0.0	1.3E-03	0.0	1.8E-03	0.0	2.2E+00
3	8	low	0.0	5.5E-04	0.0	1.0E-03	0.0	1.2E+00
		medium	3.3E-03	5.0E-04	0.0	9.7E-04	1.9E-03	1.1E+00
		high	0.0	5.3E-04	0.0	9.1E-04	0.0	1.1E+00
	10	low	7.0E-03	9.5E-04	0.0	1.7E-03	0.0	4.9E+00
		medium	3.2E-03	8.3E-04	0.0	1.7E-03	1.3E-03	4.4E+00
		high	2.3E-03	8.0E-04	0.0	1.7E-03	4.5E-03	4.0E+00
	12	low	5.9E-03	2.0E-03	2.2E-03	3.2E-03	5.2E-03	7.0E+00
		medium	7.5E-03	2.3E-03	0.0	3.7E-03	9.0E-02	1.8E+01
		high	3.3E-03	2.0E-03	0.0	3.3E-03	1.7E-03	5.6E+00
	14	low	-1.9E-02	3.5E-03	-2.3E-02	5.6E-03	-2.2E-02	6.1E+00
		medium	-1.7E-02	3.0E-03	-2.0E-02	5.8E-03	-2.0E-02	5.5E+00
		high	-8.8E-03	2.8E-03	-8.8E-03	5.4E-03	-4.8E-03	5.1E+00
	16	low	-7.2E-02	5.4E-03	-7.6E-02	8.5E-03	-7.0E-02	1.2E+01
		medium	-7.1E-02	5.6E-03	-7.6E-02	8.1E-03	-7.3E-02	1.2E+01
		high	-4.0E-02	4.9E-03	-5.2E-02	8.4E-03	-4.7E-02	8.1E+00

Table 6.7: Relative percentage gap and solution times of the heuristic methods for data set S2.

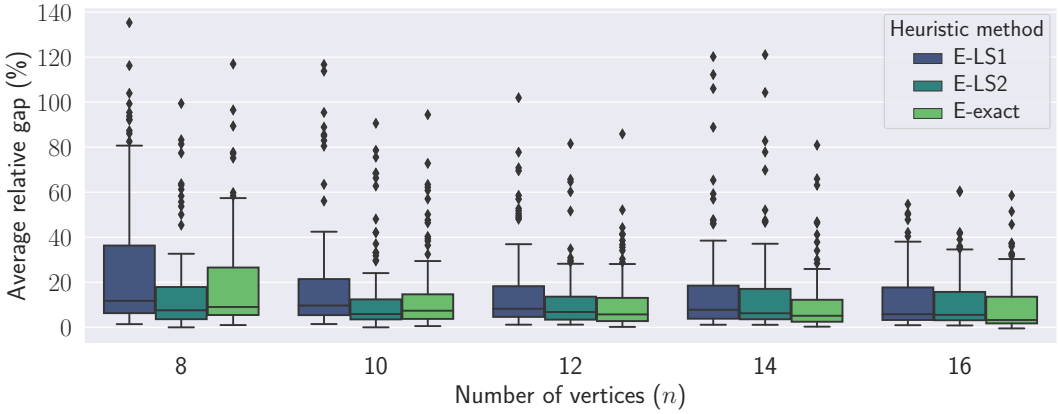


Figure 6.6: Average relative gap (out of 50 trials) of the heuristic solution found by each of the three EM variants to the best integer feasible solution found by the MILP with SBC.

as a function of the number of vertices. As visible from the figure, average gap values are typically below 20%. There is however considerable variance, and actual values may be as high as 120%.

Overall the proposed heuristics are shown to effectively arrive at optimal solutions (or high quality solutions) in reduced computing time. However we emphasize that the optimality of the heuristic solutions cannot be theoretically guaranteed. We can only show that a heuristic solution is indeed optimal once we have access to an exact method which can find the optimal solution and certify optimality.

6.4 Comparison to the ground truth

In this section we compare the model parameters found by the exact methods with the ground truth parameters used in the generation of each instance. Namely, we compare the resulting community assignments $\hat{\mathbf{Z}}$ and affinity matrix $\hat{\mathbf{\Omega}}$ of the optimal solution of maximum likelihood to the ground truth parameters $\mathbf{Z}^*$ and $\mathbf{\Omega}^*$ of the generative model. When the optimum is not known, the model parameters of the best integer feasible solution is considered instead. In this analysis we make use of the following notations:

- A is the community agreement $A(\hat{\mathbf{Z}}, \mathbf{Z}^*)$ between the community assignments of maximum likelihood and the ground truth communities.
- G is the relative percentage gap between the value of the likelihood function evaluated using the ground truth parameters, $\mathcal{L}^* = \log P(A|\mathbf{\Omega}^*, \mathbf{Z}^*)$, and the likelihood under the DCSBM of the estimated parameters $\hat{\mathcal{L}} = \log P(A|\hat{\mathbf{\Omega}}, \hat{\mathbf{Z}})$. Specifically:

$$G = \frac{\mathcal{L}^* - \hat{\mathcal{L}}}{\hat{\mathcal{L}}} \tag{6-2}$$

– R represents how frequently the assortativity structure of the estimated affinity matrix $\hat{\mathbf{\Omega}}$ (whether assortative or disassortative) matches the assortativity structure of the ground truth $\mathbf{\Omega}^*$. Here we consider that the affinity matrix is assortative if it is strongly assortative, otherwise it is considered disassortative.

The detailed results are presented in Tables 6.8 and 6.9 for data sets S1 and S2 respectively. Figure 6.7 shows that the agreement between the recovered communities and the ground truth communities is higher when n is larger and when the absolute difference $|\omega_{\text{in}} - \omega_{\text{out}}|$ is larger, for data set S1.

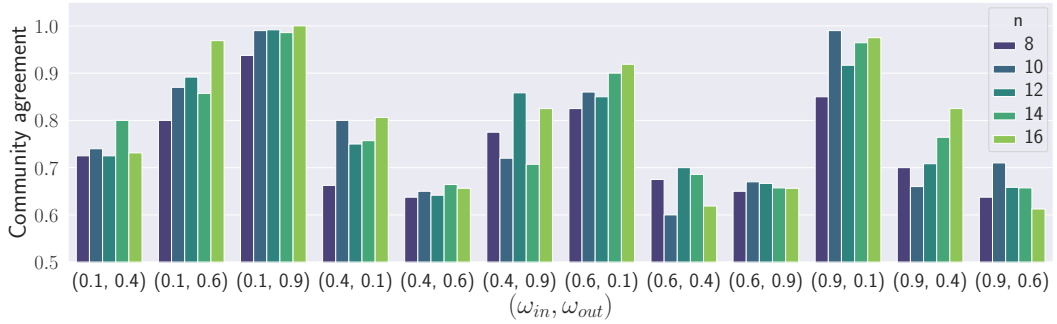


Figure 6.7: (Average) agreement between the community assignments of maximum likelihood and the true underlying community assignments, as a function of the number of vertices and the parameter values $(\omega_{\text{in}}, \omega_{\text{out}})$, for data set S1.

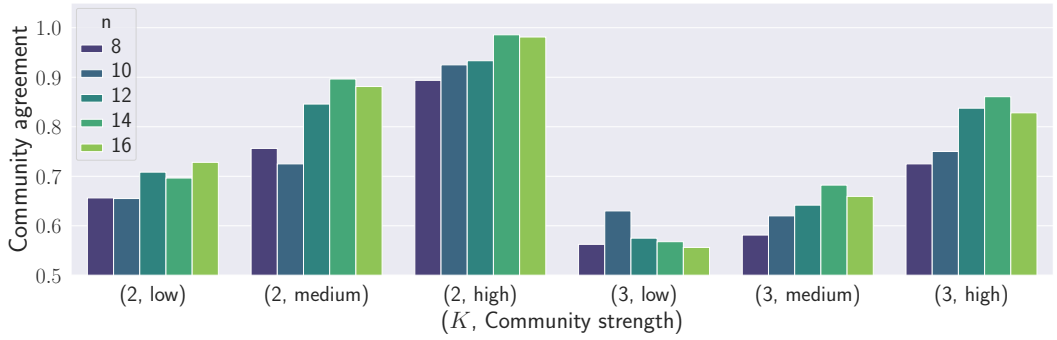


Figure 6.8: (Average) agreement between the community assignments of maximum likelihood and the true underlying community assignments, as a function of the number of vertices n, the number of communities K and the level of community strength, for data set S2.

Respectively, for data set S2, the agreement is shown to be higher when the community strength is higher (Figure 6.8). Recovery of the underlying assortativity structure is likewise easier when the absolute difference $|\omega_{\text{in}} - \omega_{\text{out}}|$

		$n = 8$			$n = 10$			$n = 12$			$n = 14$			$n = 16$		
ω_{in}	ω_{out}	A	G	R	A	G	R	A	G	R	A	G	R	A	G	R
0.1	0.4	0.72	2.36	9/10	0.74	1.43	8/10	0.72	1.26	8/10	0.80	1.74	10/10	0.73	1.06	8/10
	0.6	0.80	1.52	9/10	0.87	1.33	9/10	0.89	1.02	10/10	0.86	0.86	10/10	0.97	0.80	10/10
	0.9	0.94	0.80	10/10	0.99	0.61	10/10	0.99	0.39	10/10	0.99	0.42	10/10	1.00	0.39	10/10
0.4	0.1	0.66	2.18	7/10	0.80	2.02	9/10	0.75	1.70	7/10	0.76	1.20	9/10	0.81	1.01	9/10
	0.6	0.64	0.56	5/10	0.65	0.42	6/10	0.64	0.39	6/10	0.66	0.37	7/10	0.66	0.35	9/10
	0.9	0.78	0.40	8/10	0.72	0.26	9/10	0.86	0.17	10/10	0.71	0.18	7/10	0.82	0.17	9/10
0.6	0.1	0.82	1.17	9/10	0.86	0.92	8/10	0.85	0.84	9/10	0.90	0.89	10/10	0.92	0.70	10/10
	0.4	0.68	0.47	5/10	0.60	0.48	4/10	0.70	0.40	8/10	0.69	0.39	6/10	0.62	0.36	5/10
	0.9	0.65	0.23	9/10	0.67	0.20	6/10	0.67	0.15	9/10	0.66	0.17	8/10	0.66	0.13	6/10
0.9	0.1	0.85	0.78	8/10	0.99	0.37	10/10	0.92	0.31	10/10	0.96	0.35	10/10	0.97	0.41	10/10
	0.4	0.70	0.25	8/10	0.66	0.26	6/10	0.71	0.18	8/10	0.76	0.16	7/10	0.82	0.14	9/10
	0.6	0.64	0.24	7/10	0.71	0.20	5/10	0.66	0.15	5/10	0.66	0.15	5/10	0.61	0.12	5/10

Table 6.8: Comparison between the model parameters of maximum likelihood and the ground truth parameters, for data set S1.
A - Community agreement; G - Relative percentage gap; R - Recovered assortativity structure

K	Community strength	$n = 8$			$n = 10$			$n = 12$			$n = 14$			$n = 16$		
		A	G	R	A	G	R	A	G	R	A	G	R	A	G	R
2	low	0.66	0.57	8/20	0.66	0.46	10/20	0.71	0.47	15/20	0.70	0.40	17/20	0.73	0.38	15/20
	medium	0.76	0.52	17/20	0.72	0.47	13/20	0.85	0.38	16/20	0.90	0.30	19/20	0.88	0.29	19/20
	high	0.89	0.61	18/20	0.93	0.57	19/20	0.93	0.45	20/20	0.99	0.38	20/20	0.98	0.40	20/20
3	low	0.56	3.34	4/20	0.63	2.35	5/20	0.58	0.88	3/20	0.57	0.76	3/20	0.56	0.58	3/20
	medium	0.58	2.18	3/20	0.62	1.79	6/20	0.64	1.36	8/20	0.68	0.79	9/20	0.66	0.61	10/20
	high	0.72	10.24	12/20	0.75	2.51	12/20	0.84	3.10	14/20	0.86	1.69	15/20	0.83	0.89	16/20

Table 6.9: Comparison between the solutions of maximum likelihood and the ground truth parameters of the generative models, for data set S2.

is larger, which is naturally expected. Since data set S2 contains only assortative ground truth structures, in this case R simply measures how often the estimated affinity matrix $\hat{\boldsymbol{\Omega}}$ is assortative.

The relative gap G indicates that the solution of maximum likelihood may be far from the ground truth solution, in general. This is especially true for small networks, such as the ones considered in this work, since they are subject to higher variance. The experiments show that for these data sets the ground truth parameters are always sub-optimal for the model given by the DCSBM.

A performance comparison of the heuristic algorithms in recovering the ground truth communities is also presented, in Tables 6.10 and 6.11, respectively for data sets S1 and S2. The tables report the agreement between communities recovered by the heuristic (and exact) approaches and the true underlying communities used in the generation of each instance. In the columns under "Agreement of best found solution", the best solution in terms of objective value found by each heuristic (out of 50 trials) is compared to the solution found by the MILP with SBC. The performance of the heuristics is similar to that of the exact method, with no single approach being clearly better than the rest in recovering the ground truth communities. One reason for this lies in the fact that some instances have multiple globally optimal solutions, and the proposed algorithms may arrive at different optima, having the same objective value but different values of community agreement.

For some instances the resulting community agreement is considerably low, for both exact and heuristic solution methods. Of course, in some cases it may be impossible to exactly recover the ground truth communities, even in theory, as there is not enough information present in the graph.

We further investigate how such heuristic methods perform in community recovery in an average run. The table columns under "Average agreement" report the average community agreement out of 50 trials of each heuristic method. As visible in the results, when the average agreement is considered, the heuristics are outperformed by the exact approach in almost all instances, highlighting the importance of using exact (or high-performing heuristic) optimization algorithms.

			Agreement of best found solution				Average agreement			
n	ω_{in}	ω_{out}	E-LS1	E-LS2	E-exact	MILP	E-LS1	E-LS2	E-exact	MILP
8	0.1	0.4	0.72	0.72	**0.74**	0.72	0.69	**0.73**	0.71	0.72
		0.6	0.78	0.76	**0.8**	**0.8**	0.72	0.75	0.74	**0.8**
		0.9	**0.94**	0.92	0.92	**0.94**	0.76	0.85	0.82	**0.94**
	0.4	0.1	**0.7**	0.65	0.68	0.66	0.65	**0.67**	0.64	0.66
		0.6	0.65	**0.68**	**0.68**	0.64	0.66	0.66	**0.69**	0.64
		0.9	**0.79**	**0.79**	0.78	0.78	0.73	0.75	0.75	**0.78**
	0.6	0.1	0.81	0.80	0.79	**0.82**	0.68	0.70	0.68	**0.82**
		0.4	**0.7**	0.68	**0.7**	0.68	0.66	0.67	0.65	**0.68**
		0.9	**0.65**	**0.65**	**0.65**	**0.65**	0.63	0.64	0.62	**0.65**
	0.9	0.1	0.85	**0.88**	0.86	0.85	0.72	0.78	0.75	**0.85**
		0.4	**0.7**	**0.7**	0.69	**0.7**	0.66	0.67	0.68	**0.7**
		0.6	**0.64**	**0.64**	**0.64**	**0.64**	0.63	**0.64**	0.63	**0.64**
10	0.1	0.4	0.81	0.81	**0.83**	0.74	0.67	0.71	0.72	**0.74**
		0.6	0.84	0.80	0.85	**0.87**	0.74	0.78	0.77	**0.87**
		0.9	0.98	**0.99**	**0.99**	**0.99**	0.78	0.81	0.85	**0.99**
	0.4	0.1	0.75	0.76	0.78	**0.8**	0.68	0.69	0.71	**0.8**
		0.6	**0.65**	**0.65**	0.64	**0.65**	**0.66**	0.65	0.65	0.65
		0.9	0.68	**0.72**	**0.72**	**0.72**	0.68	0.68	**0.73**	0.72
	0.6	0.1	0.85	0.85	0.85	**0.86**	0.72	0.75	0.75	**0.86**
		0.4	0.59	0.59	**0.6**	**0.6**	0.62	**0.63**	0.62	0.60
		0.9	0.65	0.63	0.63	**0.67**	0.63	0.64	0.63	**0.67**
	0.9	0.1	**0.99**	**0.99**	**0.99**	**0.99**	0.79	0.79	0.80	**0.99**
		0.4	0.64	**0.66**	0.65	**0.66**	**0.66**	**0.66**	**0.66**	**0.66**
		0.6	**0.72**	0.71	0.71	0.71	0.65	0.65	0.65	**0.71**
12	0.1	0.4	0.78	0.78	**0.81**	0.72	0.67	0.69	0.70	**0.72**
		0.6	0.88	0.88	**0.89**	**0.89**	0.75	0.80	0.81	**0.89**
		0.9	**0.99**	**0.99**	0.98	**0.99**	0.76	0.81	0.82	**0.99**
	0.4	0.1	**0.76**	0.75	0.75	0.75	0.67	0.68	0.70	**0.75**
		0.6	0.62	0.61	0.61	**0.64**	0.63	0.62	**0.65**	0.64
		0.9	**0.87**	0.86	0.86	0.86	0.72	0.70	0.71	**0.86**
	0.6	0.1	0.86	**0.87**	0.84	0.85	0.71	0.71	0.74	**0.85**
		0.4	**0.7**	**0.7**	**0.7**	**0.7**	0.66	0.68	0.67	**0.7**
		0.9	**0.68**	0.67	**0.68**	0.67	0.64	0.63	0.66	**0.67**
	0.9	0.1	**0.92**	0.91	**0.92**	**0.92**	0.75	0.77	0.78	**0.92**
		0.4	**0.71**	**0.71**	**0.71**	**0.71**	0.67	0.66	**0.71**	**0.71**
		0.6	0.65	**0.66**	**0.66**	**0.66**	0.64	0.65	0.63	**0.66**
14	0.1	0.4	**0.82**	0.81	**0.82**	0.80	0.69	0.69	0.73	**0.8**
		0.6	**0.87**	0.86	0.86	0.86	0.70	0.73	0.76	**0.86**
		0.9	**0.99**	**0.99**	**0.99**	**0.99**	0.78	0.81	0.84	**0.99**
	0.4	0.1	**0.76**	**0.76**	**0.76**	**0.76**	0.67	0.69	0.67	**0.76**
		0.6	0.64	0.66	**0.67**	0.66	0.63	0.62	0.64	**0.66**
		0.9	**0.71**	**0.71**	0.70	**0.71**	0.65	0.65	0.67	**0.71**
	0.6	0.1	0.89	**0.9**	**0.9**	**0.9**	0.74	0.72	0.74	**0.9**
		0.4	0.67	**0.69**	0.67	**0.69**	0.63	0.62	0.65	**0.69**
		0.9	0.64	**0.66**	0.64	**0.66**	0.61	0.62	0.61	**0.66**
	0.9	0.1	**0.96**	**0.96**	**0.96**	**0.96**	0.75	0.77	0.79	**0.96**
		0.4	**0.79**	**0.79**	**0.79**	0.76	0.66	0.66	0.67	**0.76**
		0.6	**0.66**	**0.66**	**0.66**	**0.66**	0.63	0.63	0.62	**0.66**
16	0.1	0.4	0.73	**0.74**	**0.74**	0.73	0.65	0.66	0.71	**0.73**
		0.6	**0.98**	0.97	**0.98**	0.97	0.77	0.78	0.81	**0.97**
		0.9	**1.0**	**1.0**	**1.0**	**1.0**	0.80	0.82	0.84	**1.0**
	0.4	0.1	0.82	**0.83**	0.81	0.81	0.68	0.68	0.70	**0.81**
		0.6	**0.68**	0.66	0.66	0.66	0.62	0.62	0.63	**0.66**
		0.9	0.83	0.81	**0.85**	0.82	0.70	0.69	0.75	**0.82**
	0.6	0.1	**0.92**	**0.92**	**0.92**	**0.92**	0.75	0.74	0.77	**0.92**
		0.4	**0.62**	**0.62**	0.61	**0.62**	0.61	**0.62**	0.61	**0.62**
		0.9	0.67	0.66	**0.69**	0.66	0.65	0.64	0.65	**0.66**
	0.9	0.1	0.96	**0.98**	**0.98**	**0.98**	0.79	0.79	0.79	**0.98**
		0.4	**0.82**	**0.82**	**0.82**	**0.82**	0.69	0.70	0.77	**0.82**
		0.6	**0.64**	**0.64**	**0.64**	0.61	**0.61**	**0.61**	**0.61**	**0.61**

Table 6.10: Comparison between heuristic and exact solution algorithms in recovering the ground truth communities of data set S1.

K	n	Community strength	Agreement of best found solution				Average agreement			
			E-LS1	E-LS2	E-exact	MILP	E-LS1	E-LS2	E-exact	MILP
2	8	low	**0.66**	**0.66**	**0.66**	**0.66**	0.65	0.65	0.64	**0.66**
		medium	0.76	0.75	**0.79**	0.76	0.68	0.68	0.68	**0.76**
		high	**0.89**	0.88	**0.89**	**0.89**	0.75	0.77	0.75	**0.89**
	10	low	0.64	0.64	0.65	**0.66**	0.65	0.65	0.64	**0.66**
		medium	0.73	0.74	**0.75**	0.72	0.69	0.69	0.67	**0.72**
		high	**0.93**	0.92	0.92	0.92	0.76	0.79	0.77	**0.92**
	12	low	**0.71**	0.70	**0.71**	**0.71**	0.66	0.66	0.67	**0.71**
		medium	0.84	0.84	**0.85**	**0.85**	0.69	0.72	0.71	**0.85**
		high	0.93	**0.94**	0.93	0.93	0.79	0.81	0.80	**0.93**
	14	low	0.70	0.70	**0.71**	0.70	0.65	0.65	0.65	**0.7**
		medium	0.89	**0.9**	0.89	**0.9**	0.71	0.72	0.74	**0.9**
		high	**0.99**	0.98	**0.99**	**0.99**	0.76	0.78	0.78	**0.99**
	16	low	0.72	**0.73**	**0.73**	**0.73**	0.67	0.68	0.69	**0.73**
		medium	**0.88**	0.87	**0.88**	**0.88**	0.74	0.74	0.78	**0.88**
		high	**0.98**	**0.98**	**0.98**	**0.98**	0.79	0.82	0.82	**0.98**
3	8	low	**0.58**	**0.58**	**0.58**	0.56	0.57	**0.58**	0.56	0.56
		medium	0.60	0.60	**0.61**	0.58	0.58	**0.61**	0.58	0.58
		high	0.75	**0.76**	0.74	0.72	0.62	0.68	0.63	**0.72**
	10	low	0.62	0.62	**0.63**	**0.63**	0.57	0.59	0.57	**0.63**
		medium	**0.64**	0.62	0.63	0.62	0.59	0.60	0.59	**0.62**
		high	0.73	**0.76**	0.74	0.75	0.62	0.69	0.64	**0.75**
	12	low	0.57	0.57	0.57	**0.58**	0.56	0.56	0.55	**0.58**
		medium	**0.65**	0.64	0.59	0.64	0.60	0.61	0.55	**0.64**
		high	**0.85**	**0.85**	0.83	0.84	0.66	0.71	0.67	**0.84**
	14	low	**0.59**	0.58	0.56	0.57	0.55	0.55	**0.57**	**0.57**
		medium	**0.7**	**0.7**	**0.7**	0.68	0.60	0.63	0.61	**0.68**
		high	0.87	**0.88**	0.87	0.86	0.67	0.72	0.70	**0.86**
	16	low	**0.58**	0.57	**0.58**	0.56	0.54	0.55	0.55	**0.56**
		medium	**0.71**	0.70	0.70	0.66	0.60	0.61	0.62	**0.66**
		high	0.83	**0.84**	**0.84**	0.83	0.66	0.70	0.70	**0.83**

Table 6.11: Comparison between heuristic and exact solution algorithms in recovering the ground truth communities of data set S2.

7
Conclusions

In this work, we introduced mathematical programming formulations for the maximum likelihood estimation problem of community detection with the DCSBM. We first introduced a descriptive formulation based on a mixed integer non-linear program (MINLP) and then we used linearization techniques to arrive at a mixed integer linear program (MILP). By carefully analyzing the model, we were able to develop a more targeted MILP method which significantly reduced the computational time needed by the direct MINLP formulation. The proposed solution methods can find optimal solutions of maximum likelihood with a certificate of global optimality.

The ability to solve a machine learning model to optimality is valuable for the many reasons. In general, when proposing a new model for a specific task, it is useful to analyze the optimal solutions of the model as only optimal solutions really represent the model itself. If we only consider heuristic solutions which are far from optimality, we cannot draw reliable conclusions about the suitability of the model for the given task. Therefore, solving a machine learning model to optimality is an important step to validate the usefulness of the model and to assess the performance of (heuristic) solution methods.

In other words, exact methods allow us to decouple two sources of possible issues that may lead to undesired results: issues stemming from the adoption of an inadequate model (or inappropriate model family) and issues that arise from the difficulty of solving the search problem that underlies the assumed model. Once we are able to exactly solve a proposed model we can analyze it in a more principled way, eliminating possible issues that may arise from accepting sub-optimal solutions. In this regard, Gribel *et al.* (2019) [77] provide an interesting discussion on the use of high-performance optimization algorithms in machine learning, specifically for the minimum sum-of-squares clustering problem.

There is a common belief in machine learning that solving a model to optimality necessarily leads the model to overfit to the training data, therefore increasing the generalization error. Thus, practitioners and researchers are often not interested in solving to optimality the models that they adopt. In fact some methods in machine learning are even designed to find sub-optimal solutions in order to avoid overfitting and to reduce the generalization error. A

typical example is the use of early stopping techniques in neural networks and deep learning. On the other hand, a classical way of avoiding overfitting is to include regularization terms in the objective function or to reduce the model complexity in some specific way, effectively modifying the model itself instead of forcing the solution method to find sub-optimal solutions. While this notion that too much optimization hurts generalization may be true for some types of models, there is not enough evidence to suggest that it is universally true for all model families. As an example, Bertsimas *et al.* (2017) [55] have recently challenged this belief, and experimentally showed that optimal decision trees can have a lower generalization error than decision trees learned via heuristics (as long as proper regularization is included in the model).

Of course in some cases the benefits of exact solution methods may only be possible at a high computational cost. As visible in our computational experiments, the proposed exact methods are only practical for small networks with up to 16 vertices. For larger instances, the only option currently available is to rely on heuristics. For this reason, we have discussed three simple heuristic approaches based on the EM algorithm. The heuristics were shown to achieve optimal solutions for most instances, in reduced computing times compared to the exact methods. On some instances the heuristic solutions had better objective values than the ones found by the exact methods (in cases when they were unable to prove solution optimality within the time limit). Even though our computational experiments focused on small networks, we emphasize that the behavior and performance of heuristic approaches in larger networks may be very different from the results observed in this simpler setting.

This work raises some interesting possibilities for future research. In practice it would be interesting to solve instances with hundreds of thousands of vertices to optimality. Thus, there is a large room for improvement. In particular, during the computational experiments we noted that the MILP often finds the optimal solution early in the optimization process, but it takes a very long time to prove optimality. One possibility is to investigate different formulations to the problem and study the performance of the solution methods.

For future work, we aim to investigate the use of semidefinite programming relaxations in the branch-and-bound process. In the SBM literature many works have studied SDP relaxations for recovering communities in the Planted Partition Model and in general SBMs. We are interested in investigating whether such SDP relaxations can be extended to optimize for both the community assignments and the latent parameters which define the affinity matrix in the general SBM. By adopting such formulations we hope to have

tighter lower bounds, which could increase the efficiency of the branch-and-bound exploration.

Bibliography

[1] KARRER, B.; NEWMAN, M. E. J.. **Stochastic blockmodels and community structure in networks**. Physical Review E, 83(1):016107, 2011.

[2] FORTUNATO, S.; HRIC, D.. **Community detection in networks: a user guide**. Physics Reports, 659:1–44, 2016.

[3] GOLDENBERG, A.; ZHENG, A. X.; FIENBERG, S. E. ; AIROLDI, E. M.. **A survey of statistical network models**. Now Publishers Inc, 2010.

[4] CHEN, J.; YUAN, B.. **Detecting functional modules in the yeast protein–protein interaction network**. Bioinformatics, 22(18):2283–2290, 2006.

[5] CLINE, M. S.; SMOOT, M.; CERAMI, E.; KUCHINSKY, A.; LANDYS, N.; WORKMAN, C.; CHRISTMAS, R.; AVILA-CAMPILO, I.; CREECH, M. ; GROSS, B.. **Integration of biological networks and gene expression data using Cytoscape**. Nature Protocols, 2(10):2366, 2007.

[6] CABREROS, I.; ABBE, E. ; TSIRIGOS, A.. **Detecting community structures in Hi-C genomic data**. 2016 Annual Conference on Information Science and Systems (CISS), p. 584–589, 2016.

[7] ABBE, E.. **Community detection and stochastic block models: recent developments**. Journal of Machine Learning Research, 18(1):6446–6531, 2017.

[8] ABBE, E.; BANDEIRA, A. S. ; HALL, G.. **Exact recovery in the stochastic block model**. IEEE Transactions on Information Theory, 62(1):471–487, 2015.

[9] NEWMAN, M.. **Networks**. Oxford University Press, 2018.

[10] BARABÁSI, A. L.. **Network science**. Cambridge University Press, 2016.

[11] BOLLOBÁS, B.; BÉLA, B.. **Random graphs**. Número 73. Cambridge University Press, 2001.

[12] NEWMAN, M. E.. **Random graphs as models of networks**. Handbook of graphs and networks, 1:35–68, 2003.

[13] STROGATZ, S. H.. **Exploring complex networks**. Nature, 410(6825):268–276, 2001.

[14] ALBERT, R.; BARABÁSI, A. L.. **Statistical mechanics of complex networks**. Reviews of Modern Physics, 74(1):47, 2002.

[15] ERDŐS, P.; RÉNYI, A.. **On random graphs**. Publ. Math. Debrecen, 6:290–297, 1959.

[16] ERDŐS, P.; RÉNYI, A.. **On the evolution of random graphs**. Publ. Math. Inst. Hung. Acad. Sci, 5(1):17–60, 1960.

[17] JEUB, L. G.; BALACHANDRAN, P.; PORTER, M. A.; MUCHA, P. J. ; MAHONEY, M. W.. **Think locally, act locally: detection of small, medium-sized, and large communities in large networks**. Physical Review E, 91(1):012821, 2015.

[18] LESKOVEC, J.; LANG, K. J.; DASGUPTA, A. ; MAHONEY, M. W.. **Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters**. Internet Mathematics, 6(1):29–123, 2009.

[19] NEWMAN, M. E. J.. **Equivalence between modularity optimization and maximum likelihood methods for community detection**. Physical Review E, 94:052315, 2016.

[20] HOLLAND, P. W.; LASKEY, K. B. ; LEINHARDT, S.. **Stochastic blockmodels: first steps**. Social Networks, 5(2):109–137, 1983.

[21] FAUST, K.; WASSERMAN, S.. **Blockmodels: interpretation and evaluation**. Social Networks, 14(1-2):5–61, 1992.

[22] SNIJDERS, T. A.; NOWICKI, K.. **Estimation and prediction for stochastic blockmodels for graphs with latent block structure**. Journal of Classification, 14(1):75–100, 1997.

[23] GRIBEL, D.; VIDAL, T. ; GENDREAU, M.. **Assortative-constrained stochastic block models**. arXiv preprint arXiv:2004.11890, 2020.

[24] AMINI, A. A.; LEVINA, E.. **On semidefinite relaxations for the block model**. The Annals of Statistics, 46(1):149–179, 2018.

[25] LANCICHINETTI, A.; FORTUNATO, S.. **Consensus clustering in complex networks**. Scientific Reports, 2:336, 2012.

[26] VON LUXBURG, U.. **A tutorial on spectral clustering**. Statistics and Computing, 17(4):395–416, 2007.

[27] ALOISE, D.; CAFIERI, S.; CAPOROSSI, G.; HANSEN, P.; PERRON, S. ; LIBERTI, L.. **Column generation algorithms for exact modularity maximization in networks**. Physical Review E, 82:046112, 2010.

[28] MYUNG, I. J.. **Tutorial on maximum likelihood estimation**. Journal of Mathematical Psychology, 47(1):90–100, 2003.

[29] DEL PIA, A.; KHAJAVIRAD, A. ; KUNISKY, D.. **Linear programming and community detection**. arXiv preprint arXiv:2006.03213, 2020.

[30] LEE, C.; WILKINSON, D. J.. **A review of stochastic block models and extensions for graph clustering**. Applied Network Science, 4(1):122, 2019.

[31] DECELLE, A.; KRZAKALA, F.; MOORE, C. ; ZDEBOROVÁ, L.. **Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications**. Physical Review E, 84(6):066106, 2011.

[32] HAJEK, B.; WU, Y. ; XU, J.. **Achieving exact cluster recovery threshold via semidefinite programming**. IEEE Transactions on Information Theory, 62(5):2788–2797, 2016.

[33] MOSSEL, E.; NEEMAN, J. ; SLY, A.. **Consistency thresholds for binary symmetric block models**. arXiv preprint arXiv:1407.1591, 3(5), 2014.

[34] ABBE, E.; SANDON, C.. **Community detection in general stochastic block models: fundamental limits and efficient algorithms for recovery**. 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, p. 670–688, 2015.

[35] MOSSEL, E.; NEEMAN, J. ; SLY, A.. **A proof of the block model threshold conjecture**. Combinatorica, 38(3):665–708, 2018.

[36] MASSOULIÉ, L.. **Community detection thresholds and the weak ramanujan property**. Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, p. 694–703, 2014.

[37] CAI, T. T.; LI, X.. **Robust and computationally feasible community detection in the presence of arbitrary outlier nodes**. The Annals of Statistics, 43(3):1027–1059, 2015.

[38] CHEN, Y.; SANGHAVI, S. ; XU, H.. **Clustering sparse graphs**. Advances in Neural Information Processing Systems, p. 2204–2212, 2012.

[39] CHEN, Y.; XU, J.. **Statistical-computational tradeoffs in planted problems and submatrix localization with a growing number of clusters and submatrices**. Journal of Machine Learning Research, 17(1):882–938, 2016.

[40] LEI, J.; RINALDO, A.. **Consistency of spectral clustering in stochastic block models**. The Annals of Statistics, 43(1):215–237, 2015.

[41] QIN, T.; ROHE, K.. **Regularized spectral clustering under the degree-corrected stochastic blockmodel**. Advances in Neural Information Processing Systems, p. 3120–3128, 2013.

[42] ROHE, K.; CHATTERJEE, S. ; YU, B.. **Spectral clustering and the high-dimensional stochastic blockmodel**. The Annals of Statistics, 39(4):1878–1915, 2011.

[43] BORDENAVE, C.; LELARGE, M. ; MASSOULIÉ, L.. **Non-backtracking spectrum of random graphs: community detection and non-regular ramanujan graphs**. 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, p. 1347–1357, 2015.

[44] AMINI, A. A.; CHEN, A.; BICKEL, P. J. ; LEVINA, E.. **Pseudo-likelihood methods for community detection in large sparse networks**. The Annals of Statistics, 41(4):2097–2122, 2013.

[45] NOWICKI, K.; SNIJDERS, T. A. B.. **Estimation and prediction for stochastic blockstructures**. Journal of the American Statistical Association, 96(455):1077–1087, 2001.

[46] MCDAID, A. F.; MURPHY, T. B.; FRIEL, N. ; HURLEY, N. J.. **Improved bayesian inference for the stochastic block model with application to large networks**. Computational Statistics & Data Analysis, 60:12–31, 2013.

[47] PEIXOTO, T. P.. **Bayesian stochastic blockmodeling**. Advances in Network Clustering and Blockmodeling, p. 289–332, 2019.

[48] AIROLDI, E. M.; BLEI, D. M.; FIENBERG, S. E. ; XING, E. P.. **Mixed membership stochastic blockmodels**. Journal of Machine Learning Research, 9(Sep):1981–2014, 2008.

[49] BICKEL, P.; CHOI, D.; CHANG, X. ; ZHANG, H.. **Asymptotic normality of maximum likelihood and its variational approximation for stochastic blockmodels**. The Annals of Statistics, 41(4):1922–1943, 2013.

[50] CELISSE, A.; DAUDIN, J.-J. ; PIERRE, L.. **Consistency of maximum-likelihood and variational estimators in the stochastic block model**. Electronic Journal of Statistics, 6:1847–1899, 2012.

[51] MARIADASSOU, M.; ROBIN, S. ; VACHER, C.. **Uncovering latent structure in valued graphs: a variational approach**. The Annals of Applied Statistics, 4(2):715–742, 2010.

[52] XU, G.; TSOKA, S. ; PAPAGEORGIOU, L. G.. **Finding community structures in complex networks using mixed integer optimisation**. The European Physical Journal B, 60(2):231–239, 2007.

[53] GRÖTSCHEL, M.; WAKABAYASHI, Y.. **A cutting plane algorithm for a clustering problem**. Mathematical Programming, 45(1-3):59–96, 1989.

[54] BRANDES, U.; DELLING, D.; GAERTLER, M.; GORKE, R.; HOEFER, M.; NIKOLOSKI, Z. ; WAGNER, D.. **On modularity clustering**. IEEE Transactions on Knowledge and Data Engineering, 20(2):172–188, 2007.

[55] BERTSIMAS, D.; DUNN, J.. **Optimal classification trees**. Machine Learning, 106(7):1039–1082, 2017.

[56] BANDI, H.; BERTSIMAS, D. ; MAZUMDER, R.. **Learning a mixture of gaussians via mixed-integer optimization**. INFORMS Journal on Optimization, 1(3):221–240, 2019.

[57] FLAHERTY, P.; WIRATCHOTISATIAN, P.; LEE, J. A. ; TRAPP, A. C.. **Maximum a-posteriori estimation for the gaussian mixture model via mixed integer nonlinear programming**. arXiv preprint arXiv:1911.04285, 2019.

[58] BELOTTI, P.; BONAMI, P.; FISCHETTI, M.; LODI, A.; MONACI, M.; NOGALES-GÓMEZ, A. ; SALVAGNIN, D.. **On handling indicator constraints in mixed integer programming**. Computational Optimization and Applications, 65(3):545–566, 2016.

[59] CUSSENS, J.; JÄRVISALO, M.; KORHONEN, J. H. ; BARTLETT, M.. **Bayesian network structure learning with integer programming: Polytopes, facets and complexity**. Journal of Artificial Intelligence Research, 58:185–229, 2017.

[60] BENNETT, K. P.; PARRADO-HERNÁNDEZ, E.. **The interplay of optimization and machine learning research**. Journal of Machine Learning Research, 7(Jul):1265–1281, 2006.

[61] GAMBELLA, C.; GHADDAR, B. ; NAOUM-SAWAYA, J.. **Optimization models for machine learning: a survey**. arXiv preprint arXiv:1901.05331, 2019.

[62] BIXBY, R. E.. **A brief history of linear and mixed-integer programming computation**. Documenta Mathematica, p. 107–121, 2012.

[63] BELOTTI, P.. **Couenne: a user's manual**. Technical report, Lehigh University, 2009.

[64] BONAMI, P.; LODI, A.; TRAMONTANI, A. ; WIESE, S.. **On mathematical programming with indicator constraints**. Mathematical Programming, 151(1):191–223, 2015.

[65] GUROBI OPTIMIZATION. **Gurobi optimizer 8.0**. Gurobi: http://www.gurobi.com, 2018.

[66] CPLEX, IBM ILOG. **V12. 1: User's manual for CPLEX**. International Business Machines Corporation, 46(53):157, 2009.

[67] PLASTRIA, F.. **Formulating logical implications in combinatorial optimisation**. European Journal of Operational Research, 140(2):338–353, 2002.

[68] DEMPSTER, A. P.; LAIRD, N. M. ; RUBIN, D. B.. **Maximum likelihood from incomplete data via the EM algorithm**. Journal of the Royal Statistical Society: Series B (Methodological), 39(1):1–22, 1977.

[69] BILMES, J. A.. **A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models**. International Computer Science Institute, 4(510):126, 1998.

[70] XU, L.; JORDAN, M. I.. **On convergence properties of the EM algorithm for gaussian mixtures**. Neural Computation, 8(1):129–151, 1996.

[71] JAIN, A. K.; MURTY, M. N. ; FLYNN, P. J.. **Data clustering: a review**. ACM Computing Surveys (CSUR), 31(3):264–323, 1999.

[72] BOTTOU, L.; BENGIO, Y.. **Convergence properties of the k-means algorithms**. Advances in Neural Information Processing Systems, p. 585–592, 1995.

[73] BOSCHETTI, M. A.; MANIEZZO, V.; ROFFILLI, M. ; RÖHLER, A. B.. **Matheuristics: optimization, simulation and control**. Springer, Berlin, Heidelberg, 2009.

[74] ARCHETTI, C.; SPERANZA, M. G.. **A survey on matheuristics for routing problems**. EURO Journal on Computational Optimization, 2(4):223–246, 2014.

[75] BEZANSON, J.; EDELMAN, A.; KARPINSKI, S. ; SHAH, V. B.. **Julia: a fresh approach to numerical computing**. SIAM Review, 59(1):65–98, 2017.

[76] DUNNING, I.; HUCHETTE, J. ; LUBIN, M.. **JuMP: a modeling language for mathematical optimization**. SIAM Review, 59(2):295–320, 2017.

[77] GRIBEL, D.; VIDAL, T.. **HG-means: a scalable hybrid genetic algorithm for minimum sum-of-squares clustering**. Pattern Recognition, 88:569–583, 2019.

A
Log-Likelihood Function

The likelihood function can be expressed as:

$$P(A|\boldsymbol{g},\boldsymbol{\Omega},\boldsymbol{\theta}) = \prod_{i<j} \frac{(\theta_i\theta_j\omega_{g_ig_j})^{A_{ij}}}{A_{ij}!}\exp\left(-\theta_i\theta_j\omega_{g_ig_j}\right)\times \prod_i \frac{\left(\frac{1}{2}\theta_i^2\omega_{g_ig_i}\right)^{A_{ii}/2}}{\left(\frac{1}{2}A_{ii}\right)!}\exp\left(-\frac{1}{2}\theta_i^2\omega_{g_ig_i}\right) \tag{A-1}$$

Assuming that the expected number of edges between vertices i and j is given by $\theta_i\theta_j\omega_{g_ig_j} = \frac{k_ik_j}{2m}\omega_{g_ig_j}$, yields the following equation:

$$P(A|\boldsymbol{g},\boldsymbol{\Omega}) = \prod_{i<j} \frac{\left(\frac{k_ik_j}{2m}\omega_{g_ig_j}\right)^{A_{ij}}}{A_{ij}!}\exp\left(-\frac{k_ik_j}{2m}\omega_{g_ig_j}\right)\times \prod_i \frac{\left(\frac{1}{2}\frac{k_i^2}{2m}\omega_{g_ig_i}\right)^{A_{ii}/2}}{\left(\frac{1}{2}A_{ii}\right)!}\exp\left(-\frac{1}{2}\frac{k_i^2}{2m}\omega_{g_ig_i}\right) \tag{A-2}$$

Taking the logarithm of the above expression yields:

$$\mathcal{L} = \log P(A|\boldsymbol{g},\boldsymbol{\Omega}) = \sum_{i<j}^{n} A_{ij}\log(\tfrac{k_ik_j}{2m}) + A_{ij}\log\omega_{g_ig_j} - \log(A_{ij}!) - \tfrac{k_ik_j}{2m}\omega_{g_ig_j} + \sum_{i}^{n} \tfrac{1}{2}A_{ii}\log(\tfrac{1}{2}\tfrac{k_i^2}{2m}) + \tfrac{1}{2}A_{ii}\log\omega_{g_ig_i} - \log(\tfrac{1}{2}A_{ii}!) - \tfrac{1}{2}\tfrac{k_i^2}{2m}\omega_{g_ig_i} \tag{A-3}$$

Ignoring terms which are constant results in:

$$\mathcal{L} = \log P(A|\boldsymbol{g},\boldsymbol{\Omega}) = \sum_{i<j}^{n}\left(A_{ij}\log\omega_{g_ig_j} - \tfrac{k_ik_j}{2m}\omega_{g_ig_j}\right) + \sum_{i}^{n}\left(\tfrac{1}{2}A_{ii}\log\omega_{g_ig_i} - \tfrac{1}{2}\tfrac{k_i^2}{2m}\omega_{g_ig_i}\right) \tag{A-4}$$

$$= \frac{1}{2}\sum_{i,j}^{n}\left(A_{ij}\log\omega_{g_ig_j} - \tfrac{k_ik_j}{2m}\omega_{g_ig_j}\right) \tag{A-5}$$

which is the model adopted throughout the paper.

B
MILP formulation

The mixed integer linear program described in Section 4.2 can be explicitly stated as follows:

$$
\begin{aligned}
\min_{\mathbf{Z},\mathbf{\Omega},\mathbf{Y},\mathbf{X}} \quad & \frac{1}{2}\sum_{i,j}^{n}\sum_{r,s}^{K} x_{ijrs} \\
\text{s. t.} \quad & x_{ijrs} \geq a_{ij\widetilde{\omega}}\,\omega_{rs} + b_{ij\widetilde{\omega}} - \overline{M_{ijrs}}(1 - y_{ijrs}), && \forall ij \in V, \forall rs \in \mathcal{C}, \forall \widetilde{\omega} \in \mathbb{R}^{+} \\
& x_{ijrs} \leq \overline{M_{ijrs}}\, y_{ijrs} && \forall ij \in V, \forall rs \in \mathcal{C} \\
& x_{ijrs} \geq \underline{M_{ijrs}}\, y_{ijrs} && \forall ij \in V, \forall rs \in \mathcal{C} \\
& z_{ir} - y_{ijrs} \geq 0 && \forall ij \in V, \forall rs \in \mathcal{C} \\
& z_{js} - y_{ijrs} \geq 0 && \forall ij \in V, \forall rs \in \mathcal{C} \\
& 1 - z_{ir} - z_{js} + y_{ijrs} \geq 0 && \forall ij \in V, \forall rs \in \mathcal{C} \\
& \sum_{r=1}^{K} z_{ir} = 1 && \forall i \in V \\
& z_{11} = 1 \\
& \sum_{i=2}^{j-1}\sum_{l=1}^{r-1} z_{il} - \sum_{l=1}^{r} z_{jl} \leq j - 3, && \forall r \in \{2, \ldots, K-1\}, \forall j \in \{r, \ldots, n\} \\
& z_{ir} \in \{0, 1\} && \forall i \in V, \forall r \in \mathcal{C} \\
& y_{ijrs} \in \{0, 1\} && \forall ij \in V, \forall rs \in \mathcal{C} \\
& \omega_{rs} \in \mathbb{R}^{+} && \forall rs \in \mathcal{C} \\
& x_{ijrs} \in \mathbb{R}^{+} && \forall ij \in V, \forall rs \in \mathcal{C}
\end{aligned}
$$

$$
\begin{aligned}
\text{where} \quad & a_{ij\widetilde{\omega}} := -\frac{A_{ij}}{\widetilde{\omega}} + \frac{k_i k_j}{2m} && \forall ij \in V, \forall \widetilde{\omega} \in \mathbb{R}^{+}, \\
& b_{ij\widetilde{\omega}} := A_{ij}(1 - \log \widetilde{\omega}) && \forall ij \in V, \forall \widetilde{\omega} \in \mathbb{R}^{+}.
\end{aligned}
$$

In terms of the size of the problem, note that variable $\mathbf{Z}$ has nK elements, variable $\mathbf{Y}$ has n^2K^2 elements, variable $\mathbf{\Omega}$ has K^2 elements and variable $\mathbf{X}$ has n^2K^2 elements, resulting in a total of $2n^2K^2 + nK + K^2$ variables. In the special case when $K = 2$, the problem has $8n^2 + 2n + 4$ variables.