Pattern Recognition 152 (2024) 110487

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Community detection in the stochastic block model by mixed integer programming

Breno Serrano [a,c,*], Thibaut Vidal [b,c]

[a] School of Management, Technical University of Munich, Germany
[b] Department of Mathematical and Industrial Engineering, Polytechnique Montréal, Canada
[c] Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Brazil

ARTICLE INFO

Keywords:
Community detection
Stochastic block model
Mixed integer programming
Machine learning
Unsupervised learning
Local search

ABSTRACT

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 based on the DCSBM is to search for the model parameters that are the most likely to have produced the observed network data through 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 these exact methods with classical heuristic algorithms based on expectation–maximization (EM). The solutions given by exact methods give us a principled way of measuring the experimental performance of classical heuristics and comparing different variations thereof.

1. Introduction

In the community detection problem, we observe a graph $G = (V, E)$ and aim to find groups of vertices (or *communities*) which present a similar connection pattern [1,2]. Some important applications of community detection include the study of social networks [3–5] and predicting the functional family of proteins [6], among others. One of the most popular approaches for this task consists in fitting a generative model (such as the DCSBM) to the observed graph G, and to search for the parameters which maximize the likelihood of the model.

In the DCSBM, the number of edges connecting any two vertices i and j only depends on their group memberships g_i and g_j and on the set of parameters θ_i which control the expected degree of each vertex i. The DCSBM is characterized by a $K \times K$ affinity matrix $\boldsymbol{\Omega} = (\omega_{rs})$, where K is the number of communities in the graph. As in many previous works, we can assume that K is available from background knowledge on the application, or obtained through dedicated estimation techniques (see, e.g., [7]). The number of edges between any two nodes i and j is drawn from a Poisson distribution with mean $\theta_i\theta_j\omega_{g_ig_j}$. The probability that the observed network G, represented by the adjacency matrix A, was generated from the DCSBM can be expressed as:

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

which defines the likelihood function of the DCSBM. As in Newman [8], we consider in this work the case where $\theta_i\theta_j = \frac{k_ik_j}{2m}$, where $\frac{k_ik_j}{2m}$ corresponds to the expected number of edges in the configuration model. After applying the log on both sides and grouping together constant terms, the log-likelihood function becomes (cf. Newman [8]):

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

The MLE problem consists in finding the affinity matrix $\boldsymbol{\Omega}^{\text{ML}}$ and group membership assignments $\boldsymbol{g}^{\text{ML}}$ that maximize the log-likelihood function (2).

* Corresponding author at: School of Management, Technical University of Munich, Germany.
E-mail addresses: breno.serrano@tum.de (B. Serrano), thibaut.vidal@polymtl.ca (T. Vidal).

https://doi.org/10.1016/j.patcog.2024.110487
Received 26 January 2021; Received in revised form 4 April 2024; Accepted 5 April 2024
Available online 8 April 2024
0031-3203/© 2024 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

The most common methods for this problem are heuristics, which are not guaranteed to converge to the solution of maximum likelihood. Theoretical convergence guarantees typically focus on the probability of recovering the *true* underlying communities of a graph generated by the DCSBM in the asymptotic limit where the size of the network grows to infinity. Most studies (see, e.g., [9]) adopt a statistical and information-theoretic viewpoint and provide thresholds and conditions under which different types of algorithms recover the underlying community assignments with high probability for different asymptotic regimes. In this work, instead, we adopt a combinatorial optimization viewpoint, proposing mixed integer programming (MIP) formulations and exact solution methods that can provably find the optimal solution of the maximum likelihood model for an observed graph.

Firstly, we propose a simple descriptive formulation, which results in a mixed integer non-linear program (MINLP). This model can be solved to optimality with algorithms based on spatial branch-and-bound (sBB). Building upon this first formulation, we employ linearization techniques to produce a mixed integer linear programming (MILP) formulation. To improve the formulation, we rely on a dynamic generation of valid inequalities along with symmetry-breaking constraints. Moreover, we carefully analyze the problem to derive tight bounds on the model variables, permitting significant reductions of computational effort to find optimal solutions.

Our solution approach is motivated by recent advances in the field of mathematical programming. Due to extensive research, optimization solvers have continuously improved, permitting the solution of increasingly larger problems. Even though MILP is $\mathcal{NP}$-hard in general, it is now possible to solve instances of dimensions that were absolutely out of reach of solution methods a few decades ago. This increase in computational efficiency is a consequence of methodological improvements which, coupled with hardware improvements, have resulted in speed-up factors of the order of 10^{11} over two decades (based on the figures reported in [10]).

The study of exact solution algorithms is also an important methodological step for machine learning research. In particular, exact algorithms provide benchmark solutions that can be used to evaluate the performance of heuristics, measuring how far they are from the optima of the maximum likelihood model. To initiate such an analysis, we consider three basic variants of the 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.

Therefore, this work makes a significant step towards developing exact solution methods for the problem of MLE of the DCSBM. Specifically, our contribution is threefold. First, we propose a MILP formulation and an optimal solution algorithm for the problem. To the best of our knowledge, this is the first work to investigate exact methods for the DCSBM. Second, we perform computational experiments on synthetic graphs to assess the performance of the exact methods, showing that the algorithm based on the MILP formulation clearly outperforms that of the MINLP formulation. Third, we compare the exact methods with some classical heuristics based on the EM algorithm. These experiments show that MIP approaches are an important asset to evaluate the performance of heuristics and highlight the importance of optimal algorithms in machine learning research. All data sets and source codes needed to reproduce our results are available at: github.com/vidalt/Optimal-SBM.

The remainder of this paper is structured as follows. Section 2 draws an overview of related works. Section 3 introduces our mathematical programming formulations and solution techniques. Section 4 describes the heuristic approaches and Section 5 presents the computational experiments comparing the proposed solution methods. Finally, Section 6 concludes.

2. Related works

A large body of literature on SBMs focuses on community recovery from an information-theoretical and statistical viewpoint. Special cases of the SBM have regularly been considered, such as the (balanced) Planted Partition Model (PPM). We refer the reader to the survey by Abbe [9] for a detailed description of different recovery requirements and consistency analyses of various algorithms.

Some studies have explored exact solution methods for community detection based on modularity maximization [11,12], which is known to be $\mathcal{NP}$-hard [13]. Zhang and Moore [14] and Newman [8] show 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. [12] and solved using a branch-and-bound method. The authors discuss the use of symmetry-breaking constraints to improve the efficiency of the branch-and-bound exploration. Aloise et al. [11] employ techniques based on column generation to improve on previous works [12], reporting a reduction in computing time, and solving larger instances with up to 512 vertices to optimality (vs. 105 vertices in previous works). Recently, Aref et al. [15] proposed a branch-and-cut algorithm for modularity optimization that can more efficiently solve previously challenging instances. Modularity maximization (equivalently MLE of the PPM) is, however, a very specialized case of community detection, and no exact solution algorithm has been proposed for MLE of the general SBM to date.

Several studies proposed algorithms for community recovery based on SDP relaxations of the MLE model [16–18], leading to new results regarding the recovery of communities in SBMs with general K under an implicit assumption of strong assortativity. Amini and Levina [19] proposed an SDP relaxation that is tighter than previous ones and works for a broader class of SBMs, including for disassortative structures.

Del Pia et al. [20] considered the problem of exact community recovery for the assortative planted bisection model and discussed the theoretical performance of linear programming (LP) relaxations of the minimum bisection problem for community recovery. They derived sufficient and necessary conditions for recovery using the LP relaxation for different asymptotic regimes.

Algorithms based on expectation–maximization for MLE have been investigated, for example, by [21] 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. [22]. The general idea of pseudo-likelihood is to approximate the likelihood by ignoring some of the dependency structure of the data to make the model more tractable.

Metaheuristics have been applied to various related clustering problems. Among others, Gribel and Vidal [23] proposed a hybrid genetic algorithm for the minimum sum-of-squares clustering problem, and Hansen et al. [24] proposed a variable neighborhood search heuristic for normalized cut clustering.

More broadly, several related works investigated the application of mixed integer programming to classical machine learning models. Mixed integer optimization has been used to learn optimal decision trees and decision diagrams [25,26], Gaussian mixture models (GMM) [27,28], ramp-loss and hard-margin loss SVMs [29,30], among others, as well as to find optimal counterfactual explanations [31]. We refer to [32,33] for some surveys on mathematical programming applied to popular machine learning models.

3. Solving the DCSBM to optimality

This section introduces mathematical programming formulations for the MLE problem given by Eq. (2). We first present a descriptive formulation as a MINLP model. Then, we employ different techniques to linearize the model, leading to a MILP formulation. To further improve computational efficiency, we discuss the use of bounds-tightening and symmetry-breaking techniques.

3.1. Formulation as a mixed integer non-linear program

Let z_{ir} be a binary variable which takes value 1 if vertex $i \in V$ is assigned to community $r \in C$ and 0 otherwise, where $C = \{1, \dots, K\}$ represent the possible communities. The continuous variables ω_{rs}, for $r, s \in C$, represent the elements of the connectivity matrix $\boldsymbol{\Omega}$. The MLE problem (2) can be modeled as the following MINLP, where we minimize the negative log-likelihood (the constant term was omitted):

$$\underset{\mathbf{Z},\boldsymbol{\Omega}}{\text{minimize}} \quad \frac{1}{2}\sum_{i,j}^{n}\sum_{r,s}^{K} f_{ij}(\omega_{rs})\, z_{ir} z_{js} \tag{3}$$

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

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

$$\omega_{rs} \in \mathbb{R}^+ \qquad \forall r, s \in C \tag{6}$$

where

$$f_{ij}(\omega_{rs}) = -A_{ij} \log \omega_{rs} + \frac{k_i k_j}{2m}\omega_{rs}. \tag{7}$$

In this model, Constraints (4) ensure that each vertex is assigned to exactly one community. This model can be solved to optimality by global optimization solvers such as `Couenne` [34]. However, solution time quickly increases with the size of the networks. In the next sections, we propose some techniques to linearize Model (3)–(6) into a MILP along with additional refinements that permit a significant reduction of solution time in comparison to the MINLP.

3.2. Formulation as a mixed integer linear program

The linearization of the MINLP formulation is done in several steps. First of all, we linearize the function $f_{ij}(\omega_{rs})$ by piecewise outer-approximation. The function f_{ij} is convex everywhere in its domain, for $A_{ij} > 0$, since: $\frac{\partial^2 f_{ij}}{\partial \omega^2} = \frac{A_{ij}}{\omega^2} > 0$, $\forall \omega \in \mathbb{R}^+$. Thus, the value of f_{ij} is always greater than or equal to its tangent calculated at any point $\widetilde{\omega} \in \mathbb{R}^+$:

$$f_{ij}(\omega) \geq a_{ij\tilde{\omega}}\, \omega + b_{ij\tilde{\omega}} \qquad \forall \omega \in \mathbb{R}^+, \tag{8}$$

where the coefficients $a_{ij\tilde{\omega}}$ and $b_{ij\tilde{\omega}}$ defining the tangent line are given by:

$$a_{ij\tilde{\omega}} = -\frac{A_{ij}}{\widetilde{\omega}} + \frac{k_i k_j}{2m} \tag{9}$$

$$b_{ij\tilde{\omega}} = f_{ij}(\widetilde{\omega}) - a_{ij\tilde{\omega}}\, \widetilde{\omega} = A_{ij}(1 - \log \widetilde{\omega}). \tag{10}$$

By making use of this property, we introduce variables f_{ijrs} to represent the value of $f_{ij}(\omega_{rs})$, and reformulate Model (3)–(6) into:

$$\underset{\mathbf{Z},\boldsymbol{\Omega},\mathbf{F}}{\text{minimize}} \quad \frac{1}{2}\sum_{i,j}^{n}\sum_{r,s}^{K} f_{ijrs}\, z_{ir} z_{js} \tag{11}$$

$$\text{subject to} \quad \sum_{r=1}^{K} z_{ir} = 1 \qquad \forall i \in V \tag{12}$$

$$f_{ijrs} \geq a_{ij\tilde{\omega}}\, \omega_{rs} + b_{ij\tilde{\omega}} \qquad \forall i, j \in V,\ \forall r, s \in C,\ \forall \widetilde{\omega} \in \mathbb{R}^+ \tag{13}$$

$$z_{ir} \in \{0,1\} \qquad \forall i \in V,\ \forall r \in C \tag{14}$$

$$\omega_{rs} \in \mathbb{R}^+ \qquad \forall r, s \in C \tag{15}$$

$$f_{ijrs} \in \mathbb{R} \qquad \forall i, j \in V,\ \forall r, s \in C. \tag{16}$$

This model contains an infinite number of constraints of type (13), one for every $\widetilde{\omega} \in \mathbb{R}^+$.

Next, we linearize the objective function. 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 as a set of linear constraints:

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

$$z_{js} - y_{ijrs} \geq 0, \tag{18}$$

$$1 - z_{ir} - z_{js} + y_{ijrs} \geq 0. \tag{19}$$

As a result, the objective function can be 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:

$$x_{ijrs} \leq \overline{M}\, y_{ijrs}, \tag{20}$$

$$x_{ijrs} \geq \underline{M}\, y_{ijrs}, \tag{21}$$

$$x_{ijrs} \geq f_{ijrs} - \overline{M}(1 - y_{ijrs}). \tag{22}$$

Constraints (22) can be combined with Constraints (13), yielding:

$$x_{ijrs} \geq a_{ij\tilde{\omega}}\, \omega_{rs} + b_{ij\tilde{\omega}} - \overline{M}(1 - y_{ijrs}) \qquad \forall \widetilde{\omega} \in \mathbb{R}^+. \tag{23}$$

It is well known that formulations with big-M constants suffer from a weak continuous relaxation (and therefore from larger solution times) if the lower and upper bounds ($\underline{M}$ and $\overline{M}$) are large in absolute value [29,35]. Section 3.4 proposes some natural values for these bounds. The resulting model is a MILP, and it can be solved by conventional branch-and-cut algorithms.

3.3. Dynamic constraints generation

As mentioned previously, the MILP model has an infinite number of constraints of type (23). To solve it in practice, we initially only consider a small set of these constraints, for a set of break-points $\widetilde{\omega}_p$ indexed by $p \in \mathcal{B}$. Then, new constraints are dynamically introduced in the model during the solution process. Whenever an integer-feasible solution is found during the branch-and-bound, the algorithm checks if any constraint given by (23) is violated with a tolerance of ϵ. In this case, the solution is declared infeasible and the violated constraints are added to the model. In effect, the method iteratively refines the approximation of the function f_{ij} until the desired precision of ϵ is achieved.

3.4. Bounds tightening

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 $2m\rho$:

$$\omega^*_{rs} = 2m\left(\frac{\sum_{i,j} A_{ij} z_{ir} z_{js}}{\sum_{i,j} k_i k_j z_{ir} z_{js}}\right) \leq 2m\rho, \tag{24}$$

where

$$\rho := \max_{i,j}\left\{\frac{A_{ij}}{k_i k_j}\right\}. \tag{25}$$

To see that (24) holds, observe that it is equivalent to:

$$\sum_{i,j}\left(\tfrac{1}{\rho}A_{ij} - k_i k_j\right) z_{ir} z_{js} \leq 0 \tag{26}$$

which is satisfied since:

$$\rho \geq \frac{A_{ij}}{k_i k_j} \qquad \forall i, j \in V \tag{27}$$

by definition of ρ in Eq. (25).

Let ω^L_{rs} 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:

$$f_{ij}(\omega_{rs}) = \frac{k_i k_j}{2m}\omega_{rs} \tag{28}$$

and therefore f_{ijrs} can be bounded by $0 \le f_{ijrs} \le \frac{k_i k_j}{2m}\omega^U_{rs}$.

Otherwise, if $A_{ij} \neq 0$, 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

$$\frac{\partial f_{ij}}{\partial \omega_{rs}} = -\frac{A_{ij}}{\omega_{rs}} + \frac{k_i k_j}{2m} = 0 \tag{29}$$

implying

$$\hat{\omega}_{rs} = \frac{2mA_{ij}}{k_i k_j} \tag{30}$$

and therefore

$$\underline{M_{ijrs}} = -A_{ij}\log\frac{2mA_{ij}}{k_i k_j} + A_{ij} = A_{ij}\left(1 - \log A_{ij} + \log\frac{k_i k_j}{2m}\right). \tag{31}$$

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 $[\omega^L_{rs}, \omega^U_{rs}]$:

$$\overline{M_{ijrs}} = \max\{f_{ij}(\omega^L_{rs}), f_{ij}(\omega^U_{rs})\}. \tag{32}$$

Overall, the upper and lower bounds are given by:

$$\overline{M_{ijrs}} = \begin{cases} \frac{k_i k_j}{2m}\omega^U_{rs}, & \text{if } A_{ij} = 0 \\ \max\{f_{ij}(\omega^L_{rs}), f_{ij}(\omega^U_{rs})\}, & \text{if } A_{ij} \neq 0 \end{cases} \tag{33}$$

$$\underline{M_{ijrs}} := \begin{cases} 0, & \text{if } A_{ij} = 0 \\ A_{ij}(1 - \log A_{ij} + \log\frac{k_i k_j}{2m}), & \text{if } A_{ij} \neq 0\,. \end{cases} \tag{34}$$

For numerical stability, we also set a lower bound $\omega^L_{rs} = 10^{-12}$ since function f_{ij} is not defined at $\omega_{rs} = 0$.

3.5. 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, each solution is, in practice, represented $K!$ times in the model, leading to an inefficient solution process. To circumvent this issue, Plastria [36] proposed a set of linear constraints that limits 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 $1, \dots, r-1$. As shown in [36], this can be achieved by including the following constraints:

$$z_{11} = 1 \tag{35}$$

$$\sum_{i=2}^{j-1}\sum_{l=1}^{r-1} z_{il} - \sum_{l=1}^{r} z_{jl} \le j - 3 \qquad \forall r \in \{2, \dots, K-1\}, \forall j \in \{r, \dots, n\} \tag{36}$$

The last cluster K is not associated with any constraint, as it will automatically contain all remaining objects which do not belong to any of the previous clusters. These constraints effectively reduce the symmetry of the problem and speed-up the solution method.

4. Heuristics for the DCSBM

The mathematical programming approaches discussed in Section 3 permit to find optimal solutions of the MLE model with a certificate of global optimality. In practice, however, they are limited to fairly small problems since their computational effort quickly rises as a function of the number of nodes in the graph. In contrast, heuristic methods usually solve a problem in reduced computational time but do not provide solution-quality guarantees. We review three natural variants of the EM algorithm based on the method proposed by Karrer and Newman [37] for community detection using the DCSBM. These approaches are based on a local search heuristic on the space of community assignments. Since we can now find optimal solutions, we conduct a disciplined experimental analysis of these heuristics to measure how far they are from the known optima of the MLE model.

4.1. Expectation–maximization algorithm

The EM algorithm was introduced by Dempster et al. [38] 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. The EM algorithm has been applied to a variety of models in machine learning, including Gaussian mixture models (GMM) and hidden Markov models (HMM) [39], as well as data clustering [40,41]. The algorithm iteratively performs an expectation step (E-step) and a maximization step (M-step). In the context of community detection, the E-step searches for community assignments $\mathbf{Z}$ that maximize the likelihood function given a connectivity matrix $\boldsymbol{\Omega}$, whereas the M-step estimates $\boldsymbol{\Omega}$ using the current assignments $\mathbf{Z}$. Each step is guaranteed to increase the log-likelihood function, and therefore the method converges towards a local optimum. We describe the M-step in Section 4.2 and discuss three natural algorithmic variations for the E-step in Section 4.3.

4.2. Maximization step (M-step)

The maximization step consists in estimating the parameters $\boldsymbol{\Omega}$ which maximize the likelihood function given a fixed assignment of vertices to communities $\mathbf{Z}$. The optimal value for ω_{rs} can be calculated in closed form as:

$$\omega^*_{rs} = 2m\left(\frac{\sum_{i,j} A_{ij} z_{ir} z_{js}}{\sum_{i,j} k_i k_j z_{ir} z_{js}}\right) = \frac{2m \cdot m_{rs}}{\kappa_r \kappa_s} \tag{37}$$

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.

4.3. Expectation step (E-step)

The expectation step consists in searching for community assignments $\mathbf{Z}$ that maximize the likelihood given the current affinity matrix $\boldsymbol{\Omega}$. This step corresponds to an $\mathcal{NP}$-hard combinatorial optimization problem [22]. There are different possible ways to perform the E-step. We highlight three main approaches which, combined with the M-step, result in three variations of the EM algorithm reported in Algorithms 1, 2, and 3. In all cases, random community assignments are used as initial state.

```
Initialize community assignments Z;
repeat
    Ω ← MaximizationStep(Z);
    ℒ ← log P(A|Ω, Z);
    repeat
        for each vertex i ∈ V and group r ∈ C do
            Consider Z′ constructed from Z by relocating
              vertex i to group r;
            ℒ′ ← log P(A|Ω, Z′);
            if ℒ′ > ℒ then
                Apply relocation and update solution:
                Z ← Z′; ℒ ← ℒ′;
    until No improving relocation can be found;
until The likelihood function can no longer be improved;
```

Algorithm 1: EM-LS1 algorithm

```
1  Initialize community assignments Z;
2  Ω ← MaximizationStep(Z);
3  ℒ ← log P(A|Ω, Z);
4  repeat
5      repeat
6          for each vertex i ∈ V and group r ∈ C do
7              Consider Z′ constructed from Z by relocating
                vertex i to group r;
8              Ω′ ← MaximizationStep(Z′);
9              ℒ′ ← log P(A|Ω′, Z′);
10             if ℒ′ > ℒ then
11                 Apply relocation and update solution:
12                 Z ← Z′; ℒ ← ℒ′; Ω ← Ω′;
13     until No improving relocation can be found;
14 until The likelihood function can no longer be improved;
```

Algorithm 2: EM−LS2 algorithm

```
1  Initialize community assignments Z;
2  repeat
3      Ω ← MaximizationStep(Z);
4      Z ← E-exact (Ω) (Find Z by solving Model (38)–(40))
5  until The likelihood function can no longer be improved;
```

Algorithm 3: EM−exact algorithm

EM−LS1 algorithm: Local search on the community assignment variables. The first variant, based on a local search approach, is described in Algorithm 1. Line 3 of the algorithm performs the M-step, while lines 5–12 correspond to the first variant of the E-step. For a fixed value of $\boldsymbol{\Omega}$, the method searches for community assignments $\mathbf{Z}$ by iterating over each vertex i of the graph and relocating it to a different community r whenever it leads to an improvement in the likelihood function. This procedure is repeated until no more improving relocation exists. It is important to note that $\boldsymbol{\Omega}$ stays fixed during the improvement phase based on relocation (E-step) and is only optimized in the M-step.

EM−LS2 algorithm: Local search integrated with M-step. This variant is described in Algorithm 2. Here the value of $\boldsymbol{\Omega}$ is re-estimated (with the M-step) each time a vertex relocation is evaluated. Therefore, the maximization step is "embedded" into the expectation step (line 8). Calculating the value of the likelihood function can be done in $O(K^2n^2)$ elementary operations from scratch. Yet, Karrer and Newman [37] described how to find the best relocation move more efficiently by instead evaluating the change in the likelihood, exploiting the property that, when a vertex changes groups, only some terms of the likelihood function need to be updated. Thus, finding the community relocation that produces the maximum increase in the likelihood function can be done in time $O(K(K+\bar{k}))$ on average, where $\bar{k}$ is the average degree of a vertex in G.

EM−exact algorithm: Exact community assignments. Finally, as shown in Algorithm 3, the complete E-step can be formulated as an integer quadratic program (IQP) of Eqs. (38)–(40) and solved using an exact solution method.

$$\underset{\mathbf{Z}}{\text{minimize}} \quad \frac{1}{2}\sum_{i,j}^{n}\sum_{r,s}^{K} f_{ij}(\omega_{rs})\, z_{ir} z_{js} \tag{38}$$

$$\text{subject to} \quad \sum_{r=1}^{q} z_{ir} = 1 \qquad \forall i \in V \tag{39}$$

$$z_{ir} \in \{0,1\} \qquad \forall i \in V, r \in C \tag{40}$$

This variant of EM effectively applies, in turn, an optimal expectation step and an optimal maximization step. It is, therefore, the approach that is closest to the canonical EM concept. Model (38)–(40) seeks community assignments $\mathbf{Z}$ that maximize the likelihood, for a fixed $\boldsymbol{\Omega}$. In contrast to Model (3)–(6), the terms $f_{ij}(\omega_{rs})$ in the objective function are now constant. Despite this simplification, the E-step remains an $\mathcal{NP}$-hard problem [22]. It can be solved to optimality using standard MIP solvers based on branch-and-cut, such as Gurobi and CPLEX, for small and medium instances. It is less scalable, but noteworthy as a benchmark to evaluate the impact of optimal expectation steps in EM heuristics. This variant of the EM algorithm effectively becomes a *matheuristic* [42], a term used to refer to methods that combine metaheuristics with mathematical programming components.

5. Computational experiments

The goals of our computational experiments are twofold.

1. We compare the performance of the proposed exact methods in terms of computational time and scalability.
2. Using our knowledge of optimal solutions and bounds, we measure to which extent the heuristics can find the true optimum of the maximum likelihood estimation problem.

The experiments were performed on an Intel Xeon E5-2620 2.1 GHz processor machine with 128 GB of RAM and CentOS Linux 7 (Core) operating system. The high-level programming language used in the implementation was Julia [43], and the package JuMP [44] was used as the modeling language for the exact methods. The underlying optimization solvers adopted for the exact methods were Couenne [34] as the global optimization solver for the MINLP and CPLEX for the MILP. For a fair comparison between the methods, we set the precision for dynamic constraints generation to a value of $\epsilon = 10^{-6}$ such that it is consistent with the numerical-precision parameters (e.g., relative gap and convergence tolerances) of the solvers. For reproducibility, we provide our source code and all experimental data at github.com/vidalt/Optimal-SBM.

5.1. Instances

Synthetic graphs allow us to control the factors that might influence the difficulty of community detection, such as network size and community structure (e.g., degree of separability and assortativity). Therefore, we generated two groups of data sets, denoted S1 and S2, composed of synthetic graphs produced by the DCSBM. These graphs contain a number of vertices n ranging from 8 to 16 and a number of edges m ranging from 4 to 115. This problem scale allows to find optimal solutions and, in the largest cases, still challenges the solution capabilities of the exact methods.

For group S1, we set $K = 2$ and generated graphs with $n \in \{8, 10, 12, 14, 16\}$. Since $K = 2$, the affinity matrix $\boldsymbol{\Omega}$ of the model has three parameters: two diagonal elements ω_{11}, ω_{22} and one off-diagonal $\omega_{12} = \omega_{21}$. For each $(\omega_{\text{in}}, \omega_{\text{out}}) \in \{0.1, 0.4, 0.6, 0.9\}^2$, such that $\omega_{\text{in}} \neq \omega_{\text{out}}$, we sampled ω_{11}, ω_{22} from $\mathcal{U}(\omega_{\text{in}} - 0.1, \omega_{\text{in}} + 0.1)$ and we sampled ω_{12} from $\mathcal{U}(\omega_{\text{out}} - 0.1, \omega_{\text{out}} + 0.1)$, where $\mathcal{U}(a, b)$ is the uniform distribution in the interval $[a, b]$. This gives $4 \times 3 = 12$ combinations of values for $(\omega_{\text{in}}, \omega_{\text{out}})$. Six combinations are assortative (with $\omega_{\text{in}} > \omega_{\text{out}}$) and six combinations are disassortative (with $\omega_{\text{in}} < \omega_{\text{out}}$). For statistical significance, we generated 10 instances for each combination, yielding a total of $5 \times 4 \times 3 \times 10 = 600$ instances.

Data group S2 is composed of strongly assortative graphs with $K \in \{2, 3\}$, $n \in \{8, 10, 12, 14, 16\}$ and three levels of community strength: low, medium and high. For each level of community strength, we

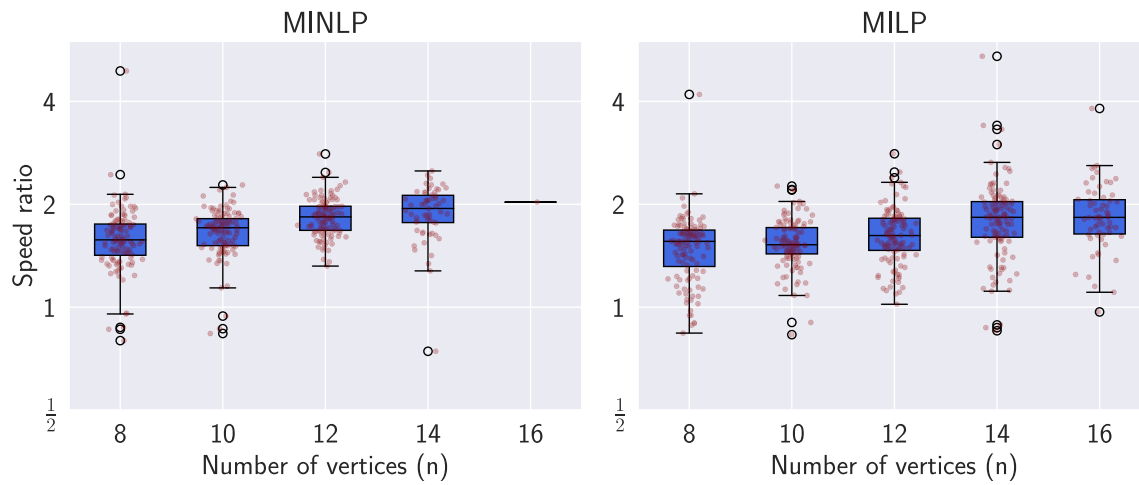

Fig. 1. Speed ratios between the solution time of the methods *without* and *with* SBC (MINLP on the left and MILP on the right).

Table 1
Range of possible values for the diagonal and off-diagonal elements of the affinity matrix $\boldsymbol{\Omega}$.

	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]

sampled the diagonal and off-diagonal elements of $\boldsymbol{\Omega}$ from a uniform distribution in the corresponding interval given by Table 1. For each configuration, we generated a total of 10 instances, leading to $2 \times 3 \times 5 \times 10 = 300$ instances.

5.2. Performance of the exact methods

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

General Comparison. Tables 2 and 3 present the following results for the exact methods: number of instances solved to optimality ("Opt"), percentage gap ("Gap"), solution time in seconds ("Time"), and number of nodes explored in the branch-and-bound tree ("Nodes"). The values reported in both tables are averaged over the 10 instances of each type (except for "Opt"). The exact methods' percentage gap is calculated based on the log-likelihood function (including the constant terms) as $\text{Gap} = (\text{UB} - \text{LB})/\text{UB}$, where LB and UB are the lower and upper objective bounds. A summary line is included in the bottom of each table showing the total number of instances solved to optimality (for column "Opt") and average results for all other columns.

As seen in these results, the MILP is clearly faster than the MINLP for both groups of data sets. When $K = 2$, the MILP can optimally solve all instances with up to $n = 14$ vertices, whereas the MINLP is already unable to find the optimum for some instances with 14 vertices. This visible reduction in solution time is a consequence of a more efficient branch-and-bound exploration. Indeed, for the instances that are solved to optimality, the MILP visits fewer search nodes than the MINLP. For larger instances (e.g., when $K = 3$ in Table 3), both methods have difficulties to consistently find optimal solutions within the time limit. However, for instances that could not be solved to optimality, percentage gaps are generally much smaller for the MILP. These results clearly illustrate that the DCSBM is indeed challenging to solve to optimality.

Impact of the symmetry-breaking constraints. Fig. 1 illustrates the impact of using SBC with the MINLP (on the left) and with the MILP (on the right). It shows the speed ratio between the solution time of the method without and with SBC, depending on the number of vertices in data group S1. The results are summarized as boxplots, with whiskers that extend to 1.5 times the interquartile range. Points outside this range are marked as outliers and noted with a "∘".

As highlighted on Fig. 1, the use of SBC has a beneficial impact on the MINLP and MILP solution methods. Even in the simple case with $K = 2$, adding SBC clearly improves the solution time of both exact methods for the great majority of instances. The improvement becomes more marked as n increases, with solution times up to 2x faster.

Comparison of the formulations. Finally, Fig. 2 compares the solution times of the exact methods on data sets of group S1. As n increases, the variance in the distribution of the speed ratio increases. The speed ratios are nonetheless always greater than 1, meaning that the MILP approach is faster than the MINLP, regardless of the use of the SBC. In some cases, the MILP is as high as 32 times faster than the MINLP.

5.3. Performance of the heuristic methods

To evaluate the performance of the heuristic solution approaches, we compare their solutions to the solutions found by the exact methods. For each instance in S1 and S2, we run the three EM variants for 50 trials (with different random starts). As an additional reference and baseline, we also provide the results of a standard method for spectral clustering (see, e.g., [45–47]). For each instance the relative percentage gap is calculated as:

$$\text{Gap (\%)} = 100 \times \frac{\text{OBJ} - \text{BKS}}{\text{BKS}} \tag{41}$$

where OBJ is the objective value of the heuristic solution, and BKS (best-known solution) is the objective value of the optimal or best integer-feasible solution found by the MILP with SBC.

Tables 4 and 5 present average gap values and solution times out of the 50 trials for data groups S1 and S2, respectively. As visible from these results, the solution times of `EM-LS1` and `EM-LS2` are generally close, with `EM-LS1` being slightly faster on average. The solution time of `EM-exact` is orders of magnitude higher than that of the other heuristics since it involves the exact solution of a MIP during each expectation step, whereas `spectral` appears to be the fastest method overall.

In terms of solution quality, `EM-LS2` achieves a lower gap on average, compared to `EM-LS1`. The comparison with `EM-exact` leads to more contrasted observations: for data group S1, `EM-exact` achieved the lowest average gap, whereas for S2 it was outperformed by `EM-LS2`. For some settings, the average gap obtained by the heuristics is small or even negative (e.g., for S2, with $K = 3$, $n = 16$ and

Table 2
General performance of the exact methods (MINLP vs. MILP) for data group S1, with SBC and without them (NoSBC).

n	ω_{in}	ω_{out}	MINLP								MILP							
			NoSBC				SBC				NoSBC				SBC			
			Opt	Gap	Time	Nodes	Opt	Gap	Time	Nodes	Opt	Gap	Time	Nodes	Opt	Gap	Time	Nodes
8	0.1	0.4	10	0.00	6.3	306.3	10	0.00	3.7	167.6	10	0.00	0.8	92.5	10	0.00	0.6	59.6
		0.6	10	0.00	10.1	587.4	10	0.00	6.6	274.7	10	0.00	1.2	166.6	10	0.00	0.8	86.9
		0.9	10	0.00	11.0	569.3	10	0.00	7.0	216.7	10	0.00	1.2	120.0	10	0.00	0.8	60.3
	0.4	0.1	10	0.00	6.5	388.5	10	0.00	4.2	203.8	10	0.00	0.9	140.7	10	0.00	0.7	69.4
		0.6	10	0.00	12.9	730.0	10	0.00	7.8	349.5	10	0.00	2.0	298.5	10	0.00	1.2	146.4
		0.9	10	0.00	14.9	661.3	10	0.00	9.7	317.6	10	0.00	2.1	271.0	10	0.00	1.3	138.4
	0.6	0.1	10	0.00	8.2	459.6	10	0.00	6.0	266.7	10	0.00	1.1	157.4	10	0.00	0.8	104.5
		0.4	10	0.00	14.4	704.2	10	0.00	9.2	306.8	10	0.00	2.0	302.5	10	0.00	1.2	148.2
		0.9	10	0.00	19.5	796.1	10	0.00	11.9	326.3	10	0.00	2.8	340.7	10	0.00	1.7	171.6
	0.9	0.1	10	0.00	10.2	463.8	10	0.00	6.9	238.5	10	0.00	1.0	126.6	10	0.00	0.8	73.2
		0.4	10	0.00	16.1	786.1	10	0.00	10.4	333.5	10	0.00	2.4	359.6	10	0.00	1.5	179.0
		0.6	10	0.00	16.5	797.9	10	0.00	10.8	381.1	10	0.00	2.6	359.8	10	0.00	1.6	210.5
10	0.1	0.4	10	0.00	27.4	2 056.3	10	0.00	15.4	915.9	10	0.00	3.7	608.7	10	0.00	2.6	311.7
		0.6	10	0.00	28.6	1 676.5	10	0.00	17.7	839.2	10	0.00	3.6	440.6	10	0.00	2.3	223.7
		0.9	10	0.00	39.4	2 143.0	10	0.00	22.6	999.1	10	0.00	4.2	397.1	10	0.00	2.6	192.2
	0.4	0.1	10	0.00	22.7	1 585.0	10	0.00	15.9	988.8	10	0.00	3.1	514.9	10	0.00	2.1	301.7
		0.6	10	0.00	49.5	3 625.7	10	0.00	28.8	1 528.7	10	0.00	8.5	1 347.8	10	0.00	5.4	672.1
		0.9	9	0.00	106.9	17 598.8	10	0.00	31.9	1 433.4	10	0.00	9.0	1 198.6	10	0.00	5.9	584.7
	0.6	0.1	10	0.00	35.8	2 427.8	10	0.00	21.8	1 140.8	10	0.00	4.1	720.2	10	0.00	2.8	370.5
		0.4	9	0.00	99.8	18 520.7	10	0.00	27.2	1 243.0	10	0.00	7.1	976.3	10	0.00	4.9	568.6
		0.9	10	0.00	57.9	3 393.3	10	0.00	35.8	1 563.7	10	0.00	12.8	1 561.5	10	0.00	7.9	783.8
	0.9	0.1	10	0.00	41.4	2 126.2	10	0.00	23.7	942.5	10	0.00	4.6	545.2	10	0.00	3.0	300.1
		0.4	10	0.00	57.2	3 238.4	10	0.00	32.3	1 427.1	10	0.00	8.4	1 164.3	10	0.00	5.5	562.8
		0.6	10	0.00	59.1	3 178.2	10	0.00	33.5	1 528.7	10	0.00	12.1	1 610.8	10	0.00	7.1	707.6
12	0.1	0.4	10	0.00	86.0	6 522.3	10	0.00	48.8	3 280.1	10	0.00	11.3	1 955.8	10	0.00	7.0	946.8
		0.6	10	0.00	105.6	7 322.4	10	0.00	59.9	3 242.9	10	0.00	11.1	1 523.8	10	0.00	7.7	769.8
		0.9	10	0.00	130.5	8 374.4	10	0.00	70.3	3 716.4	10	0.00	13.6	1 551.8	10	0.00	10.1	1 042.3
	0.4	0.1	10	0.00	98.2	8 169.0	10	0.00	51.0	3 613.0	10	0.00	10.4	2 138.1	10	0.00	7.0	1 222.5
		0.6	10	0.00	194.6	14 369.8	10	0.00	102.1	6 625.3	10	0.00	37.8	4 945.8	10	0.00	23.0	2 786.3
		0.9	10	0.00	176.7	11 083.3	10	0.00	99.9	5 439.4	10	0.00	36.0	4 215.0	10	0.00	21.6	2 205.0
	0.6	0.1	10	0.00	120.5	9 293.9	10	0.00	65.4	4 137.3	10	0.00	14.5	2 603.0	10	0.00	8.1	1 166.3
		0.4	10	0.00	169.5	13 084.0	10	0.00	96.6	6 478.6	10	0.00	35.0	5 318.6	10	0.00	19.4	2 735.3
		0.9	10	0.00	206.1	14 039.0	10	0.00	109.4	6 667.8	10	0.00	49.4	5 938.7	10	0.00	27.8	3 003.1
	0.9	0.1	10	0.00	148.8	9 853.2	10	0.00	78.9	4 347.5	10	0.00	21.0	2 808.6	10	0.00	12.1	1 266.2
		0.4	10	0.00	177.3	11 626.7	10	0.00	96.7	5 881.0	10	0.00	33.5	4 288.4	10	0.00	18.9	2 170.4
		0.6	9	0.00	258.6	24 597.3	10	0.00	111.4	6 751.7	10	0.00	46.4	5 334.6	10	0.00	26.9	2 711.4
14	0.1	0.4	9	0.00	362.9	30 388.0	10	0.00	188.2	14 498.9	10	0.00	35.1	5 687.8	10	0.00	22.1	3 263.6
		0.6	10	0.00	445.0	33 400.6	10	0.00	230.4	16 568.3	10	0.00	46.8	7 168.1	10	0.00	23.8	3 236.5
		0.9	9	0.40	420.9	27 246.6	10	0.00	215.2	12 588.9	10	0.00	45.3	4 864.8	10	0.00	29.2	2 746.2
	0.4	0.1	8	0.49	478.5	37 337.6	10	0.00	233.9	17 723.4	10	0.00	58.2	10 053.6	10	0.00	28.1	4 597.2
		0.6	1	4.85	574.6	37 600.6	10	0.00	350.0	24 600.9	10	0.00	180.8	19 339.3	10	0.00	98.0	10 378.5
		0.9	1	6.86	594.4	35 976.6	10	0.00	395.0	24 594.2	10	0.00	217.5	18 671.0	10	0.00	111.4	9 342.6
	0.6	0.1	8	0.63	425.6	30 829.4	10	0.00	235.4	16 307.2	10	0.00	52.7	7 601.7	10	0.00	29.0	3 664.9
		0.4	2	5.95	598.5	39 127.7	10	0.00	353.0	25 993.2	10	0.00	152.0	16 275.0	10	0.00	75.2	7 830.8
		0.9	1	9.92	596.3	33 486.5	10	0.00	442.1	28 049.6	10	0.00	252.1	21 144.3	10	0.00	130.3	9 842.1
	0.9	0.1	8	0.51	530.5	35 931.5	10	0.00	254.7	15 866.8	10	0.00	62.8	6 718.1	10	0.00	33.2	3 399.7
		0.4	1	7.86	598.8	36 665.6	10	0.00	398.0	26 503.8	10	0.00	200.2	18 398.8	10	0.00	115.6	11 125.2
		0.6	1	11.35	597.1	32 256.1	10	0.00	426.7	25 344.6	10	0.00	236.0	17 489.5	10	0.00	122.0	9 803.4
16	0.1	0.4	0	15.88	601.0	33 644.3	1	9.23	592.4	38 779.1	10	0.00	262.8	35 403.1	10	0.00	122.7	17 711.9
		0.6	0	19.67	600.9	31 410.4	2	8.27	573.8	32 368.4	10	0.00	141.1	16 431.6	10	0.00	88.5	9 795.2
		0.9	0	23.84	600.8	26 570.6	2	11.29	575.4	27 781.8	10	0.00	165.7	12 670.4	10	0.00	93.0	6 962.4
	0.4	0.1	1	13.86	575.5	34 141.5	1	7.31	557.7	36 824.9	9	0.58	247.0	31 551.6	10	0.00	140.8	17 516.1
		0.6	0	29.13	600.8	26 345.5	0	18.02	600.8	29 144.8	1	6.30	579.9	39 058.6	9	0.32	471.0	33 429.2
		0.9	0	31.66	600.6	22 139.2	0	21.77	600.6	24 664.4	3	4.97	528.6	31 429.7	8	0.87	416.9	25 972.9
	0.6	0.1	0	15.10	600.8	31 180.2	4	7.95	562.1	33 916.7	8	0.81	282.0	30 419.5	9	0.28	170.1	18 015.5
		0.4	0	23.83	600.8	25 580.5	0	16.84	600.9	30 136.4	1	5.16	599.7	47 506.8	9	0.28	417.5	35 457.5
		0.9	0	31.91	600.6	19 998.5	0	24.05	600.8	24 056.4	0	7.70	600.0	32 472.5	5	2.56	561.4	33 329.6
	0.9	0.1	0	19.38	600.7	27 602.0	3	8.96	558.4	28 266.1	9	0.47	177.3	16 306.2	10	0.00	109.6	9 363.8
		0.4	0	27.95	600.6	21 412.2	0	21.98	600.7	24 294.1	1	6.59	592.3	35 953.7	8	0.78	484.6	32 075.5
		0.6	0	30.05	600.5	21 792.6	0	21.64	600.6	25 592.2	0	9.43	600.0	30 829.3	4	3.29	585.3	32 882.9
Aggregate			417	5.52	267.5	15 653.7	493	2.96	202.3	11 376.3	542	0.70	112.3	9 498.2	582	0.14	78.9	6 346.6

low community strength), meaning in the latter case that the heuristic objective value was better than the one found by the exact method (only possible when the exact method was unable to find the optimal solution within the time limit). As visible from Table 4, `spectral` outperformed the other heuristics in some cases, e.g., for assortative instances, but led to large error gaps on disassortative instances. On average, `spectral` achieved the largest gaps among the considered methods for both data groups. This is likely because, unlike the other heuristics, the `spectral` algorithm does not directly optimize the MLE objective (Eq. (2)).

On several runs, we observed that the heuristics effectively found the optimal solutions (or high-quality solutions). This insight directly derives from our ability to find optimal solutions with the proposed exact algorithms, as the heuristics by themselves cannot give such a

Table 3
General performance of the exact methods (MILP vs. MINLP) for data group S2, with SBC and without them (NoSBC).

K	n	Comm. strength	MINLP								MILP							
			NoSBC				SBC				NoSBC				SBC			
			Opt	Gap	Time	Nodes	Opt	Gap	Time	Nodes	Opt	Gap	Time	Nodes	Opt	Gap	Time	Nodes
2	8	Low	10	0.00	12.9	551.1	10	0.00	8.3	259.9	10	0.00	3.4	250.1	10	0.00	1.9	122.7
		Medium	10	0.00	12.8	726.3	10	0.00	8.6	309.3	10	0.00	3.3	268.8	10	0.00	1.9	131.8
		High	10	0.00	10.1	432.4	10	0.00	6.4	204.2	10	0.00	2.2	142.7	10	0.00	1.4	72.1
	10	Low	10	0.00	42.5	2 543.1	10	0.00	26.1	1 147.1	10	0.00	5.9	835.6	10	0.00	3.9	428.7
		Medium	10	0.00	41.5	2 717.2	10	0.00	25.1	1 205.4	10	0.00	6.0	878.3	10	0.00	4.1	454.4
		High	10	0.00	39.4	2 260.7	10	0.00	24.7	985.2	10	0.00	4.3	529.6	10	0.00	2.7	270.7
	12	Low	10	0.00	163.0	11 381.0	10	0.00	91.9	5 779.5	10	0.00	49.9	5 554.3	10	0.00	20.1	2 414.6
		Medium	10	0.00	149.8	10 292.3	10	0.00	81.5	4 846.0	10	0.00	28.8	3 401.1	10	0.00	11.9	1 395.9
		High	10	0.00	144.9	7 602.1	10	0.00	73.1	3 672.0	10	0.00	15.2	1 924.3	10	0.00	6.9	817.4
	14	Low	4	3.92	575.2	38 428.5	10	0.00	346.0	24 528.2	10	0.00	153.3	12 810.8	10	0.00	82.8	6 691.3
		Medium	4	4.66	511.1	32 873.6	10	0.00	301.8	21 439.0	10	0.00	162.3	12 307.9	10	0.00	88.1	7 492.4
		High	10	0.00	457.3	30 102.3	10	0.00	246.4	15 366.9	10	0.00	84.6	6 258.7	10	0.00	38.3	3 014.3
	16	Low	0	24.59	600.8	26 048.4	0	18.47	600.9	30 157.5	2	6.37	579.9	27 377.8	6	1.63	474.1	25 546.0
		Medium	0	23.94	600.8	25 228.8	0	16.80	600.8	28 751.8	3	5.57	544.9	17 520.7	5	2.24	437.8	15 999.2
		High	0	21.02	600.7	27 075.7	1	9.41	598.9	33 362.2	7	2.24	376.9	12 027.5	10	0.00	252.3	9 534.4
3	8	Low	6	9.86	486.0	8 260.7	10	0.00	65.8	1 887.0	10	0.00	31.1	2 505.8	10	0.00	9.3	520.3
		Medium	8	2.89	436.3	10 404.9	10	0.00	51.0	1 668.7	10	0.00	39.3	2 866.0	10	0.00	8.0	410.2
		High	6	10.93	404.7	6 771.4	10	0.00	52.3	1 788.2	10	0.00	22.8	1 827.0	10	0.00	6.0	372.6
	10	Low	0	33.24	600.9	23 751.9	9	0.00	361.3	20 935.7	10	0.00	225.5	22 935.7	10	0.00	37.1	4 417.9
		Medium	3	18.32	532.7	25 077.7	9	0.00	284.0	18 346.7	10	0.00	106.6	11 739.5	10	0.00	23.1	3 193.6
		High	3	15.26	490.3	28 013.6	10	0.00	207.3	12 059.5	10	0.00	67.3	6 818.6	10	0.00	12.7	1 653.1
	12	Low	0	60.62	601.0	16 741.0	0	39.07	600.7	18 220.6	0	10.17	600.0	32 058.7	10	0.00	348.6	27 153.8
		Medium	0	58.57	601.0	15 707.1	0	33.35	600.6	17 730.2	2	8.06	538.6	27 221.1	9	0.36	299.5	20 088.8
		High	0	59.16	601.1	17 991.1	0	31.70	600.6	20 065.9	6	2.15	474.5	28 412.3	10	0.00	115.5	9 244.4
	14	Low	0	74.51	600.5	7 937.1	0	69.01	600.2	8 061.0	0	19.77	600.0	13 066.7	0	13.02	600.0	16 901.7
		Medium	0	64.45	600.5	7 016.3	0	68.67	600.2	7 958.9	0	18.89	600.0	12 710.3	0	12.62	600.0	15 150.3
		High	0	57.07	600.7	9 456.2	0	64.72	600.3	9 602.9	0	11.25	600.0	17 031.9	7	2.36	492.6	23 093.9
	16	Low	0	100.00	599.8	1 770.4	0	92.69	599.7	3 017.5	0	29.09	600.0	4 430.8	0	24.13	600.0	6 488.1
		Medium	0	100.00	600.0	2 407.4	0	92.67	599.6	2 509.6	0	27.60	600.0	4 882.4	0	22.48	600.0	6 196.3
		High	0	99.26	600.0	2 587.7	0	93.27	599.9	4 183.8	0	25.58	600.0	7 191.3	0	19.77	600.0	7 161.1
Aggregate			134	28.08	410.6	13 405.3	179	20.99	315.5	10 668.3	200	5.6	257.6	9 926.2	237	3.29	192.7	7 214.4

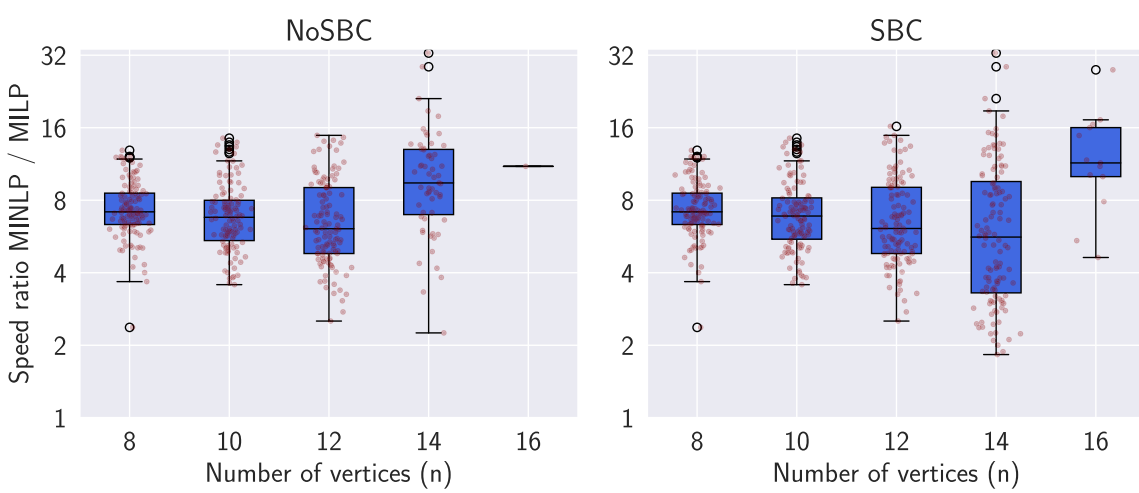

Fig. 2. Speed ratios between the solution time of the MINLP and MILP approaches (without SBC on the left, and with SBC on the right).

performance certificate. It also remains an open question whether this behavior holds for larger instance, but such an analysis would require significant methodological advances to solve larger cases to proven optimality.

5.4. Comparison to the ground truth

We finally compare the model parameters found by the exact methods with the ground truth parameters used in the generation of each instance. In this analysis, we calculate the agreement $A(\hat{\mathbf{Z}}, \mathbf{Z}^*)$ between the community assignments $\hat{\mathbf{Z}}$ of the optimal solution of maximum likelihood and the ground truth communities $\mathbf{Z}^*$ of the model. The agreement function $A(\cdot,\cdot)$ measures the maximum number of common elements between two vectors of community assignments, considering all possible permutations of the community labels. When the optimum is not known, the estimated communities of the BKS are considered instead.

Fig. 3 shows that the agreement between the recovered communities and the ground truth communities of data set S1 is higher when n is larger and when the absolute difference $|\omega_{\text{in}} - \omega_{\text{out}}|$ is larger. This is expected since there is more information in the graph. Similarly for

Table 4
(Average) Relative percentage gap and solution times of the heuristic methods for S1.

n	ω_{in}	ω_{out}	E-LS1		E-LS2		E-exact		spectral	
			Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
8	0.1	0.4	3.13	1.6×10^{-4}	1.65	2.5×10^{-4}	1.76	3.9×10^{-1}	8.20	5.9×10^{-5}
		0.6	4.29	1.3×10^{-4}	2.08	2.0×10^{-4}	3.09	4.8×10^{-1}	8.45	5.9×10^{-5}
		0.9	7.18	1.4×10^{-4}	3.44	2.4×10^{-4}	4.33	4.3×10^{-1}	17.07	6.0×10^{-5}
	0.4	0.1	3.94	1.2×10^{-4}	2.51	1.9×10^{-4}	3.29	4.0×10^{-1}	5.83	5.7×10^{-5}
		0.6	1.97	1.5×10^{-4}	1.27	2.1×10^{-4}	1.65	5.4×10^{-1}	3.40	5.6×10^{-5}
		0.9	3.35	1.6×10^{-4}	1.75	2.5×10^{-4}	2.89	5.4×10^{-1}	6.53	6.1×10^{-5}
	0.6	0.1	3.56	1.2×10^{-4}	2.26	1.9×10^{-4}	2.70	4.9×10^{-1}	3.77	6.0×10^{-5}
		0.4	2.10	1.4×10^{-4}	1.39	2.1×10^{-4}	1.83	6.1×10^{-1}	3.59	5.7×10^{-5}
		0.9	2.87	1.4×10^{-4}	1.90	2.4×10^{-4}	2.12	6.3×10^{-1}	5.74	5.7×10^{-5}
	0.9	0.1	6.13	1.3×10^{-4}	3.68	2.2×10^{-4}	4.46	4.6×10^{-1}	4.41	6.5×10^{-5}
		0.4	1.96	1.4×10^{-4}	1.42	2.2×10^{-4}	1.56	6.0×10^{-1}	2.76	5.7×10^{-5}
		0.6	2.00	1.4×10^{-4}	1.39	2.2×10^{-4}	1.67	6.2×10^{-1}	2.71	5.7×10^{-5}
10	0.1	0.4	2.26	4.6×10^{-4}	1.41	3.7×10^{-4}	1.45	7.3×10^{-1}	4.96	1.7×10^{-4}
		0.6	3.57	2.5×10^{-4}	1.76	3.9×10^{-4}	2.44	7.1×10^{-1}	8.42	6.9×10^{-5}
		0.9	6.06	2.4×10^{-4}	4.13	3.9×10^{-4}	3.98	6.7×10^{-1}	13.07	7.2×10^{-5}
	0.4	0.1	2.62	2.3×10^{-4}	2.15	3.4×10^{-4}	2.11	6.7×10^{-1}	3.23	6.4×10^{-5}
		0.6	1.37	2.7×10^{-4}	1.26	3.8×10^{-4}	0.81	8.5×10^{-1}	3.42	7.3×10^{-5}
		0.9	2.22	3.0×10^{-4}	1.36	4.2×10^{-4}	1.61	8.9×10^{-1}	5.58	6.9×10^{-5}
	0.6	0.1	3.28	2.6×10^{-4}	2.41	3.8×10^{-4}	2.29	7.1×10^{-1}	1.47	6.5×10^{-5}
		0.4	1.93	2.9×10^{-4}	1.50	4.0×10^{-4}	1.32	9.1×10^{-1}	3.33	6.8×10^{-5}
		0.9	1.78	2.8×10^{-4}	1.43	4.0×10^{-4}	1.23	9.2×10^{-1}	2.63	6.8×10^{-5}
	0.9	0.1	4.17	2.6×10^{-4}	3.68	3.9×10^{-4}	3.72	8.1×10^{-1}	0.52	6.8×10^{-5}
		0.4	2.32	3.3×10^{-4}	1.67	4.5×10^{-4}	1.69	8.6×10^{-1}	4.20	6.7×10^{-5}
		0.6	2.24	2.9×10^{-4}	1.83	3.9×10^{-4}	1.57	9.5×10^{-1}	3.58	6.7×10^{-5}
12	0.1	0.4	2.39	5.4×10^{-4}	1.79	6.9×10^{-4}	1.30	9.5×10^{-1}	6.71	7.8×10^{-5}
		0.6	4.82	4.5×10^{-4}	2.99	7.7×10^{-4}	3.06	9.6×10^{-1}	10.81	7.8×10^{-5}
		0.9	5.54	4.6×10^{-4}	3.81	7.3×10^{-4}	3.63	1.0	13.64	7.9×10^{-5}
	0.4	0.1	2.30	4.0×10^{-4}	2.16	6.6×10^{-4}	1.69	1.1	2.23	8.4×10^{-5}
		0.6	1.47	5.2×10^{-4}	1.14	7.2×10^{-4}	0.80	1.4	2.77	8.6×10^{-5}
		0.9	1.95	5.0×10^{-4}	1.95	7.0×10^{-4}	1.83	1.2	4.97	9.4×10^{-5}
	0.6	0.1	2.40	4.5×10^{-4}	2.08	7.4×10^{-4}	1.94	1.1	2.88	8.0×10^{-5}
		0.4	1.58	4.2×10^{-4}	1.30	6.0×10^{-4}	1.22	1.2	2.10	7.9×10^{-5}
		0.9	1.75	6.0×10^{-4}	1.57	7.7×10^{-4}	1.25	1.3	3.87	8.7×10^{-5}
	0.9	0.1	3.07	5.2×10^{-4}	2.68	7.3×10^{-4}	2.45	1.1	2.97	7.8×10^{-5}
		0.4	1.90	5.6×10^{-4}	1.62	8.2×10^{-4}	1.28	1.2	2.70	7.9×10^{-5}
		0.6	1.35	5.5×10^{-4}	1.19	7.3×10^{-4}	0.88	1.3	2.77	1.2×10^{-4}
14	0.1	0.4	2.59	7.2×10^{-4}	2.33	1.1×10^{-3}	1.49	1.4	6.89	1.3×10^{-4}
		0.6	3.50	7.9×10^{-4}	2.76	1.2×10^{-3}	1.95	1.4	7.26	1.0×10^{-4}
		0.9	5.61	8.4×10^{-4}	4.38	1.1×10^{-3}	3.85	1.6	14.04	1.0×10^{-4}
	0.4	0.1	2.06	7.1×10^{-4}	1.81	10.0×10^{-4}	1.25	1.6	2.07	9.2×10^{-5}
		0.6	1.21	7.4×10^{-4}	1.15	9.7×10^{-4}	0.69	1.7	2.90	1.0×10^{-4}
		0.9	1.64	7.6×10^{-4}	1.43	1.1×10^{-3}	0.98	2.0	3.34	1.0×10^{-4}
	0.6	0.1	3.76	6.5×10^{-4}	3.86	1.1×10^{-3}	2.73	5.0	2.37	8.9×10^{-5}
		0.4	1.23	7.8×10^{-4}	1.06	1.1×10^{-3}	0.75	6.2	2.53	9.4×10^{-5}
		0.9	1.39	8.6×10^{-4}	1.23	1.2×10^{-3}	0.94	4.1	3.10	9.2×10^{-5}
	0.9	0.1	4.23	7.9×10^{-4}	3.73	1.2×10^{-3}	3.12	1.6	1.71	9.4×10^{-5}
		0.4	1.48	8.2×10^{-4}	1.28	1.1×10^{-3}	0.98	1.8	2.78	9.3×10^{-5}
		0.6	1.61	8.4×10^{-4}	1.56	1.2×10^{-3}	1.24	1.9	3.27	9.2×10^{-5}
16	0.1	0.4	1.77	1.3×10^{-3}	1.54	1.7×10^{-3}	0.98	2.1	3.92	1.0×10^{-4}
		0.6	3.50	1.1×10^{-3}	3.09	1.7×10^{-3}	2.42	2.1	8.74	1.0×10^{-4}
		0.9	5.18	1.1×10^{-3}	4.16	1.8×10^{-3}	4.09	2.1	13.12	1.1×10^{-4}
	0.4	0.1	1.70	9.9×10^{-4}	1.59	1.5×10^{-3}	1.15	7.6	2.71	1.5×10^{-4}
		0.6	1.15	1.1×10^{-3}	1.07	1.7×10^{-3}	0.45	8.4	2.32	1.3×10^{-4}
		0.9	1.88	1.4×10^{-3}	1.83	1.9×10^{-3}	1.28	8.2	4.66	1.5×10^{-4}
	0.6	0.1	2.52	1.1×10^{-3}	2.59	1.7×10^{-3}	2.04	5.7	2.64	1.5×10^{-4}
		0.4	1.03	1.2×10^{-3}	1.03	1.6×10^{-3}	0.52	2.7	1.93	1.0×10^{-4}
		0.9	1.16	1.2×10^{-3}	1.14	1.6×10^{-3}	0.71	2.8	2.85	1.1×10^{-4}
	0.9	0.1	3.78	1.1×10^{-3}	4.03	1.7×10^{-3}	3.80	2.2	2.05	1.0×10^{-4}
		0.4	1.20	1.2×10^{-3}	1.15	1.7×10^{-3}	0.85	2.6	2.08	1.2×10^{-4}
		0.6	0.94	1.2×10^{-3}	0.99	1.7×10^{-3}	0.59	3.0	2.24	1.1×10^{-4}
Average			2.71	5.7×10^{-4}	2.07	8.3×10^{-4}	1.92	1.8	4.77	8.8×10^{-5}

S2, we observe that instances with higher community strength have a higher community agreement (Fig. 4).

The solution of maximum likelihood may be far from the ground truth, in general. This is especially true for small networks, such as those considered in this work, since there is often not enough information to correctly recover the underlying communities.

We also compare the heuristics in their ability to recover the ground truth communities in Tables 6 and 7, respectively for S1 and S2. These tables report the average agreement (out of 50 trials) between the recovered communities and the ground truth communities used in the generation of each instance. We compare the performance of the heuristics with the MILP with SBC.

For some instances, the resulting community agreement is low for both exact and heuristic methods, since there is not enough information present in the graph and it may be theoretically impossible to recover the ground truth. Still, in the other cases, the exact approach

Table 5
(Average) Relative percentage gap and solution times of the heuristic methods for S2.

K	n	Community strength	E-LS1 Gap (%)	E-LS1 Time (s)	E-LS2 Gap (%)	E-LS2 Time (s)	E-exact Gap (%)	E-exact Time (s)	spectral Gap (%)	spectral Time (s)
2	8	Low	2.54	1.3×10^{-4}	1.69	1.8×10^{-4}	2.00	1.6	5.82	5.9×10^{-5}
		Medium	3.14	1.5×10^{-4}	2.49	1.9×10^{-4}	2.32	1.3	2.95	6.0×10^{-5}
		High	5.81	1.5×10^{-4}	4.33	1.8×10^{-4}	5.82	1.5	5.45	5.9×10^{-5}
	10	Low	2.45	2.6×10^{-4}	1.62	3.7×10^{-4}	1.81	5.0	4.95	7.3×10^{-5}
		Medium	2.40	2.5×10^{-4}	1.94	3.5×10^{-4}	1.85	4.8	1.86	6.7×10^{-5}
		High	4.85	5.2×10^{-4}	3.63	4.1×10^{-4}	3.93	4.8	1.48	1.8×10^{-4}
	12	Low	1.47	4.5×10^{-4}	1.29	6.1×10^{-4}	0.91	3.7	2.26	7.9×10^{-5}
		Medium	3.09	4.5×10^{-4}	2.94	6.4×10^{-4}	2.33	3.4	3.19	7.9×10^{-5}
		High	4.12	4.4×10^{-4}	3.56	6.6×10^{-4}	3.80	3.3	2.40	7.7×10^{-5}
	14	Low	2.18	7.9×10^{-4}	1.85	1.1×10^{-3}	2.65	2.2×10^{1}	1.99	9.2×10^{-5}
		Medium	2.21	8.1×10^{-4}	1.88	9.8×10^{-4}	2.00	1.3×10^{1}	2.76	1.1×10^{-4}
		High	5.16	1.1×10^{-3}	4.33	1.2×10^{-3}	4.64	2.2×10^{1}	1.09	1.1×10^{-4}
	16	Low	1.61	1.1×10^{-3}	1.58	1.5×10^{-3}	1.11	2.7	2.91	1.1×10^{-4}
		Medium	2.27	1.2×10^{-3}	2.03	1.7×10^{-3}	1.60	2.4	2.79	1.1×10^{-4}
		High	4.51	1.1×10^{-3}	3.51	1.7×10^{-3}	3.45	2.7	0.79	1.0×10^{-4}
3	8	Low	5.35	4.8×10^{-4}	1.15	9.7×10^{-4}	4.55	1.3	7.61	6.4×10^{-5}
		Medium	5.90	4.4×10^{-4}	1.64	9.2×10^{-4}	5.52	1.2	6.79	6.8×10^{-5}
		High	5.95	4.4×10^{-4}	1.90	8.2×10^{-4}	5.75	1.1	4.64	7.0×10^{-5}
	10	Low	3.81	8.0×10^{-4}	1.94	1.7×10^{-3}	3.50	4.8	4.49	7.5×10^{-5}
		Medium	4.90	7.6×10^{-4}	2.01	1.8×10^{-3}	4.28	4.2	6.88	7.5×10^{-5}
		High	4.87	7.1×10^{-4}	2.35	1.6×10^{-3}	3.90	4.2	5.66	7.4×10^{-5}
	12	Low	3.25	1.7×10^{-3}	2.10	2.9×10^{-3}	2.68	7.6	5.12	9.6×10^{-5}
		Medium	3.98	1.8×10^{-3}	2.21	3.5×10^{-3}	9.01	1.8×10^{1}	3.40	1.2×10^{-4}
		High	4.55	1.8×10^{-3}	2.43	3.3×10^{-3}	4.15	6.1	2.73	8.7×10^{-5}
	14	Low	2.59	3.2×10^{-3}	1.69	4.3×10^{-3}	2.03	6.2	4.93	1.0×10^{-4}
		Medium	2.67	2.9×10^{-3}	1.46	4.8×10^{-3}	2.38	6.0	3.37	1.1×10^{-4}
		High	4.17	2.9×10^{-3}	1.96	4.8×10^{-3}	3.40	4.5	3.10	1.0×10^{-4}
	16	Low	0.84	5.2×10^{-3}	−0.05	7.2×10^{-3}	0.43	1.2×10^{1}	2.44	1.2×10^{-4}
		Medium	1.36	5.5×10^{-3}	0.49	7.6×10^{-3}	1.19	1.1×10^{1}	1.41	1.2×10^{-4}
		High	3.32	5.1×10^{-3}	1.85	7.8×10^{-3}	2.34	9.7	1.86	1.2×10^{-4}
Average			3.51	1.4×10^{-3}	2.12	2.2×10^{-3}	3.17	6.4	3.57	9.2×10^{-5}

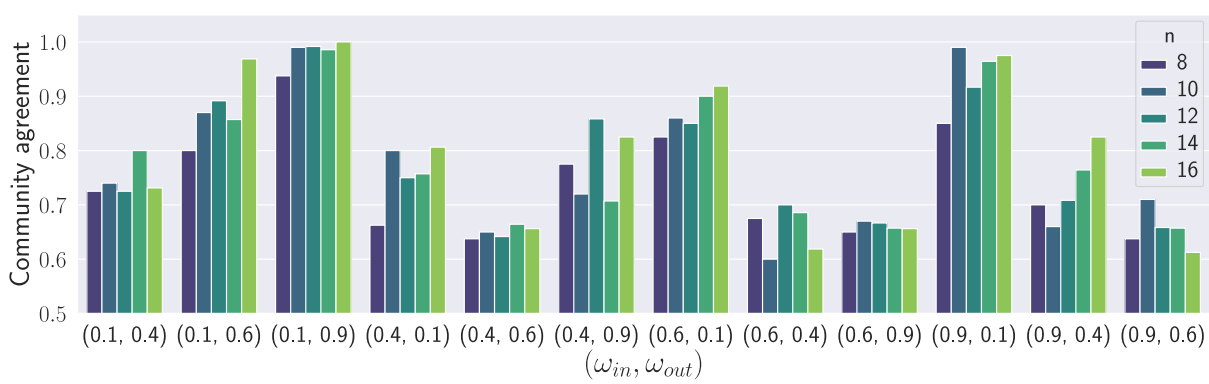

Fig. 3. (Average) agreement between the community assignments of maximum likelihood and the ground truth, as a function of n and $(\omega_{\text{in}}, \omega_{\text{out}})$, for data sets in group S1.

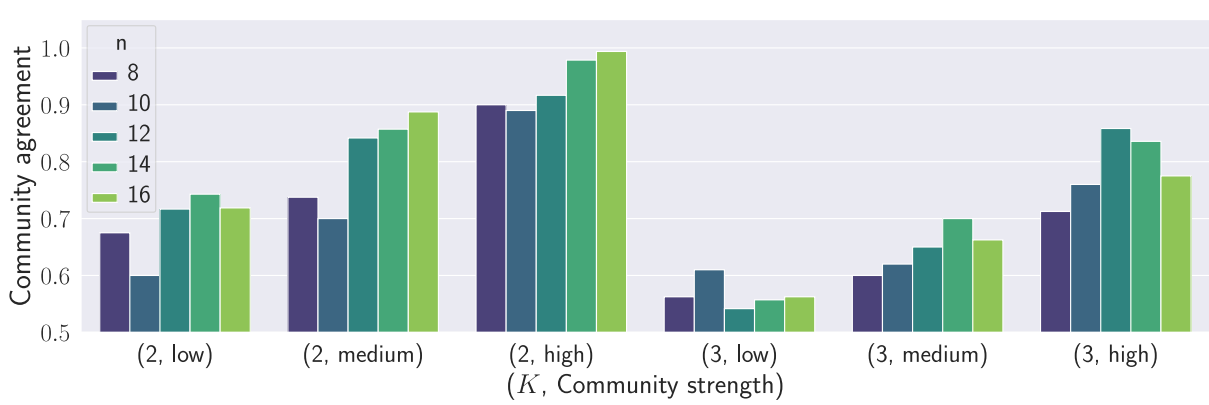

Fig. 4. (Average) agreement between the community assignments of maximum likelihood and the ground truth, as a function of n, K and the level of community strength, for data sets in group S2.

Table 6
Comparison between heuristic and exact solution algorithms in recovering the ground truth communities of data sets in group S1.

n	ω_{in}	ω_{out}	Average agreement				
			E-LS1	E-LS2	E-exact	spectral	MILP
8	0.1	0.4	0.69	**0.73**	0.71	0.60	0.72
		0.6	0.72	0.75	0.74	0.57	**0.8**
		0.9	0.76	0.85	0.82	0.56	**0.94**
	0.4	0.1	0.65	**0.67**	0.64	0.66	0.66
		0.6	0.66	0.66	**0.69**	0.57	0.64
		0.9	0.73	0.75	0.75	0.61	**0.78**
	0.6	0.1	0.68	0.70	0.68	0.70	**0.82**
		0.4	0.66	0.67	0.65	**0.68**	**0.68**
		0.9	0.63	0.64	0.62	0.61	**0.65**
	0.9	0.1	0.72	0.78	0.75	0.81	**0.85**
		0.4	0.66	0.67	0.68	**0.71**	0.70
		0.6	0.63	0.64	0.63	**0.67**	0.64
10	0.1	0.4	0.67	0.71	0.72	0.58	**0.74**
		0.6	0.74	0.78	0.77	0.60	**0.87**
		0.9	0.78	0.81	0.85	0.56	**0.99**
	0.4	0.1	0.68	0.69	0.71	0.71	**0.8**
		0.6	**0.66**	0.65	0.65	0.61	0.65
		0.9	0.68	0.68	**0.73**	0.59	0.72
	0.6	0.1	0.72	0.75	0.75	**0.86**	**0.86**
		0.4	0.62	**0.63**	0.62	0.60	0.60
		0.9	0.63	0.64	0.63	0.61	**0.67**
	0.9	0.1	0.79	0.79	0.80	0.93	**0.99**
		0.4	0.66	0.66	0.66	**0.7**	0.66
		0.6	0.65	0.65	0.65	0.69	**0.71**
12	0.1	0.4	0.67	0.69	0.70	0.58	**0.72**
		0.6	0.75	0.80	0.81	0.55	**0.89**
		0.9	0.76	0.81	0.82	0.57	**0.99**
	0.4	0.1	0.67	0.68	0.70	0.71	**0.75**
		0.6	0.63	0.62	**0.65**	0.57	0.64
		0.9	0.72	0.70	0.71	0.57	**0.86**
	0.6	0.1	0.71	0.71	0.74	0.75	**0.85**
		0.4	0.66	0.68	0.67	**0.7**	**0.7**
		0.9	0.64	0.63	0.66	0.56	**0.67**
	0.9	0.1	0.75	0.77	0.78	0.82	**0.92**
		0.4	0.67	0.66	**0.71**	0.66	**0.71**
		0.6	0.64	0.65	0.63	**0.69**	0.66
14	0.1	0.4	0.69	0.69	0.73	0.56	**0.8**
		0.6	0.70	0.73	0.76	0.55	**0.86**
		0.9	0.78	0.81	0.84	0.55	**0.99**
	0.4	0.1	0.67	0.69	0.67	0.68	**0.76**
		0.6	0.63	0.62	0.64	0.59	**0.66**
		0.9	0.65	0.65	0.67	0.56	**0.71**
	0.6	0.1	0.74	0.72	0.74	0.85	**0.9**
		0.4	0.63	0.62	0.65	0.60	**0.69**
		0.9	0.61	0.62	0.61	0.58	**0.66**
	0.9	0.1	0.75	0.77	0.79	0.87	**0.96**
		0.4	0.66	0.66	0.67	0.66	**0.76**
		0.6	0.63	0.63	0.62	0.64	**0.66**
16	0.1	0.4	0.65	0.66	0.71	0.56	**0.73**
		0.6	0.77	0.78	0.81	0.53	**0.97**
		0.9	0.80	0.82	0.84	0.53	**1.0**
	0.4	0.1	0.68	0.68	0.70	0.67	**0.81**
		0.6	0.62	0.62	0.63	0.54	**0.66**
		0.9	0.70	0.69	0.75	0.58	**0.82**
	0.6	0.1	0.75	0.74	0.77	0.80	**0.92**
		0.4	0.61	0.62	0.61	**0.64**	0.62
		0.9	0.65	0.64	0.65	0.57	**0.66**
	0.9	0.1	0.79	0.79	0.79	0.87	**0.98**
		0.4	0.69	0.70	0.77	0.72	**0.82**
		0.6	0.61	0.61	0.61	**0.63**	0.61
Average			0.69	0.70	0.71	0.65	**0.77**

Table 7
Comparison between heuristic and exact solution algorithms in recovering the ground truth communities of data sets in group S2.

K	n	Community	Average agreement				
		strength	E-LS1	E-LS2	E-exact	spectral	MILP
2	8	Low	0.64	0.65	0.63	0.67	**0.68**
		Medium	0.67	0.67	0.67	**0.8**	0.74
		High	0.75	0.77	0.74	0.78	**0.9**
	10	Low	0.63	0.62	0.60	**0.65**	0.60
		Medium	0.68	0.68	0.64	**0.73**	0.70
		High	0.75	0.77	0.78	**0.89**	**0.89**
	12	Low	0.65	0.65	0.67	0.63	**0.72**
		Medium	0.70	0.72	0.71	0.73	**0.84**
		High	0.80	0.82	0.82	0.85	**0.92**
	14	Low	0.68	0.69	0.67	0.71	**0.74**
		Medium	0.70	0.71	0.70	0.76	**0.86**
		High	0.76	0.79	0.79	0.94	**0.98**
	16	Low	0.66	0.67	0.69	0.65	**0.72**
		Medium	0.76	0.76	0.80	0.77	**0.89**
		High	0.77	0.82	0.82	0.93	**0.99**
3	8	Low	0.58	0.59	0.57	**0.6**	0.56
		Medium	0.57	0.60	0.56	**0.64**	0.60
		High	0.63	0.69	0.65	**0.76**	0.71
	10	Low	0.56	0.56	0.57	**0.64**	0.61
		Medium	0.60	0.60	0.61	**0.62**	**0.62**
		High	0.61	0.66	0.61	0.69	**0.76**
	12	Low	0.56	0.56	0.54	**0.59**	0.54
		Medium	0.62	0.62	0.57	**0.65**	**0.65**
		High	0.66	0.72	0.67	0.79	**0.86**
	14	Low	0.54	0.54	0.54	0.54	**0.56**
		Medium	0.60	0.63	0.61	0.66	**0.7**
		High	0.68	0.73	0.70	0.78	**0.84**
	16	Low	0.53	0.54	0.54	**0.59**	0.56
		Medium	0.61	0.62	0.63	**0.69**	0.66
		High	0.66	0.69	0.69	0.75	**0.78**
Average			0.65	0.67	0.66	0.72	**0.74**

outperforms the heuristics and the spectral method for most instances, highlighting the importance of good solutions for this task.

6. Conclusions

This study allowed us to fill a significant methodological gap: the lack of exact solution methods for community detection in the general SBM. Exact algorithms are indeed essential for a disciplined analysis of machine learning models and training algorithms, as they permit a precise evaluation of heuristic performance. The goal of a heuristic is to achieve an optimality gap that is systematically close to 0% for the model at hand. As heuristics do not provide guarantees regarding solution quality, we cannot evaluate their true optimality gap unless we have access to an efficient algorithm that produces optimal solutions (or at least good bounds on solution value).

To that end, we have introduced new mathematical programming formulations for the MLE model of the DCSBM. We introduced a descriptive formulation based on a MINLP and employed linearization techniques to transform it into a MILP. We also proposed bound tightening and symmetry-breaking strategies, which lead to critical improvements to the model. The proposed solution methods can find optimal solutions of maximum likelihood with a certificate of global optimality. Furthermore, we have reviewed three natural variants of the EM algorithm for this problem, and conducted extensive numerical experiments to analyze their performance.

This work raises several interesting avenues for future research. In particular, there is still space to improve the scalability of the exact methods. In our computational experiments, we noted that the MILP often identifies the optimal solution early in the optimization process but that it takes a much longer time to find good lower bounds and prove optimality. To improve this behavior, research could be pursued on new problem formulations and valid inequalities permitting to achieve tighter lower bounds and enhance the efficiency of the branch-and-bound exploration. Another alternative is to explore mathematical decomposition techniques such as column generation, which have the potential to lead to structurally-different formulations and solution approaches. Finally, we generally encourage the pursuit of a disciplined analysis of algorithms for other learning tasks, and likewise develop mathematical programming approaches for other models of importance.

CRediT authorship contribution statement

Breno Serrano: Formal analysis, Investigation, Methodology, Validation, Visualization, Writing – original draft, Data curation, Software. **Thibaut Vidal:** Conceptualization, Funding acquisition, Methodology, Project administration, Resources, Supervision, Validation, Writing – review & editing, Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research has been partially funded by CAPES, CNPq [grant number 308528/2018-2] and FAPERJ [grant number E-26/202.790/2019] in Brazil, and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) [grant number 277991500/GRK2201] in Germany. This support is gratefully acknowledged.

References

[1] S. Fortunato, D. Hric, Community detection in networks: a user guide, Phys. Rep. 659 (2016) 1–44.
[2] S. Fortunato, M.E.J. Newman, 20 Years of network community detection, Nat. Phys. 18 (8) (2022) 848–850.
[3] A. Goldenberg, A.X. Zheng, S.E. Fienberg, E.M. Airoldi, A Survey of Statistical Network Models, Now Publishers Inc, Hanover, MA, 2010.
[4] X. Qi, W. Tang, Y. Wu, G. Guo, E. Fuller, C.Q. Zhang, Optimal local community detection in social networks based on density drop of subgraphs, Pattern Recognit. Lett. 36 (2014) 46–53.
[5] P. Zhao, C.Q. Zhang, A new clustering method and its application in social networks, Pattern Recognit. Lett. 32 (15) (2011) 2109–2118.
[6] S. Mallek, I. Boukhris, Z. Elouedi, Community detection for graph-based similarity: application to protein binding pockets classification, Pattern Recognit. Lett. 62 (2015) 49–54.
[7] A. Cerqueira, F. Leonardi, Estimation of the number of communities in the stochastic block model, IEEE Trans. Inform. Theory 66 (10) (2020) 6403–6412.
[8] M.E.J. Newman, Equivalence between modularity optimization and maximum likelihood methods for community detection, Phys. Rev. E 94 (5) (2016) 052315.
[9] E. Abbe, Community detection and stochastic block models: recent developments, J. Mach. Learn. Res. 18 (1) (2017) 6446–6531.
[10] R.E. Bixby, A brief history of linear and mixed-integer programming computation, Doc. Math. (2012) 107–121.
[11] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, S. Perron, L. Liberti, Column generation algorithms for exact modularity maximization in networks, Phys. Rev. E 82 (4) (2010) 046112.
[12] G. Xu, S. Tsoka, L.G. Papageorgiou, Finding community structures in complex networks using mixed integer optimisation, Eur. Phys. J. B 60 (2) (2007) 231–239.
[13] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, D. Wagner, On modularity clustering, IEEE Trans. Knowl. Data Eng. 20 (2) (2007) 172–188.
[14] P. Zhang, C. Moore, Scalable detection of statistically significant communities and hierarchies, using message passing for modularity, Proc. Natl. Acad. Sci. 111 (51) (2014) 18144–18149.
[15] S. Aref, H. Chheda, M. Mostajabdaveh, The bayan algorithm: detecting communities in networks through exact and approximate optimization of modularity, 2022, arXiv preprint arXiv:2209.04562.
[16] T.T. Cai, X. Li, Robust and computationally feasible community detection in the presence of arbitrary outlier nodes, Ann. Statist. 43 (3) (2015) 1027–1059.
[17] Y. Chen, S. Sanghavi, H. Xu, Clustering sparse graphs, Adv. Neural Inf. Process. Syst. 25 (2012) 2204–2212.
[18] Y. Chen, J. Xu, Statistical-computational tradeoffs in planted problems and submatrix localization with a growing number of clusters and submatrices, J. Mach. Learn. Res. 17 (1) (2016) 882–938.
[19] A.A. Amini, E. Levina, On semidefinite relaxations for the block model, Ann. Statist. 46 (1) (2018) 149–179.
[20] A. Del Pia, A. Khajavirad, D. Kunisky, Linear programming and community detection, Math. Oper. Res. 48 (2) (2023) 885–913.
[21] T.A. Snijders, K. Nowicki, Estimation and prediction for stochastic blockmodels for graphs with latent block structure, J. Classification 14 (1) (1997) 75–100.
[22] A.A. Amini, A. Chen, P.J. Bickel, E. Levina, Pseudo-likelihood methods for community detection in large sparse networks, Ann. Statist. 41 (4) (2013) 2097–2122.
[23] D. Gribel, T. Vidal, HG-means: a scalable hybrid genetic algorithm for minimum sum-of-squares clustering, Pattern Recognit. 88 (2019) 569–583.
[24] P. Hansen, M. Ruiz, D. Aloise, A VNS heuristic for escaping local extrema entrapment in normalized cut clustering, Pattern Recognit. 45 (12) (2012) 4337–4345.
[25] D. Bertsimas, J. Dunn, Optimal classification trees, Mach. Learn. 106 (7) (2017) 1039–1082.
[26] A.M. Florio, P. Martins, M. Schiffer, T. Serra, T. Vidal, Optimal decision diagrams for classification, in: B. Williams, Y. Chen, J. Neville (Eds.), Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI'23, AAAI Press, Washington, D.C., 2023, pp. 7577–7585.
[27] H. Bandi, D. Bertsimas, R. Mazumder, Learning a mixture of gaussians via mixed-integer optimization, INFORMS J. Optim. 1 (3) (2019) 221–240.
[28] R.A. Sampaio, J.D. Garcia, M. Poggi, T. Vidal, Regularization and optimization in model-based clustering, Pattern Recognit. 150 (2024) 110310.
[29] P. Belotti, P. Bonami, M. Fischetti, A. Lodi, M. Monaci, A. Nogales-Gómez, D. Salvagnin, On handling indicator constraints in mixed integer programming, Comput. Optim. Appl. 65 (3) (2016) 545–566.
[30] Í. Santana, B. Serrano, M. Schiffer, T. Vidal, Support vector machines with the hard-margin loss: optimal training via combinatorial Benders' cuts, 2022, arXiv preprint arXiv:2207.07690.
[31] A. Parmentier, T. Vidal, Optimal counterfactual explanations in tree ensembles, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, ICML'21, Vol. 139, PMLR, Virtual, 2021, pp. 8422–8431.
[32] K.P. Bennett, E. Parrado-Hernández, The interplay of optimization and machine learning research, J. Mach. Learn. Res. 7 (2006) 1265–1281.
[33] C. Gambella, B. Ghaddar, J. Naoum-Sawaya, Optimization problems for machine learning: a survey, European J. Oper. Res. 290 (3) (2021) 807–828.
[34] P. Belotti, Couenne: A User's Manual, Tech. rep., Lehigh University, 2009.
[35] P. Bonami, A. Lodi, A. Tramontani, S. Wiese, On mathematical programming with indicator constraints, Math. Program. 151 (1) (2015) 191–223.
[36] F. Plastria, Formulating logical implications in combinatorial optimisation, European J. Oper. Res. 140 (2) (2002) 338–353.
[37] B. Karrer, M.E.J. Newman, Stochastic blockmodels and community structure in networks, Phys. Rev. E 83 (1) (2011) 016107.
[38] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B Stat. Methodol. 39 (1) (1977) 1–22.
[39] L. Xu, M.I. Jordan, On convergence properties of the EM algorithm for Gaussian mixtures, Neural Comput. 8 (1) (1996) 129–151.
[40] L. Bottou, Y. Bengio, Convergence properties of the k-means algorithms, Adv. Neural Inf. Process. Syst. (1995) 585–592.
[41] A.K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[42] M.A. Boschetti, V. Maniezzo, M. Roffilli, A.B. Röhler, Matheuristics: Optimization, Simulation and Control, Springer, Berlin, Heidelberg, 2009.
[43] J. Bezanson, A. Edelman, S. Karpinski, V.B. Shah, Julia: a fresh approach to numerical computing, SIAM Rev. 59 (1) (2017) 65–98.
[44] I. Dunning, J. Huchette, M. Lubin, JuMP: a modeling language for mathematical optimization, SIAM Rev. 59 (2) (2017) 295–320.
[45] X. Dong, P. Frossard, P. Vandergheynst, N. Nefedov, Clustering with multilayer graphs: A spectral perspective, IEEE Trans. Signal Process. 60 (11) (2012) 5820–5831.
[46] U. Von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416.
[47] Y. Zhang, E. Levina, J. Zhu, Detecting overlapping communities in networks using spectral methods, SIAM J. Math. Data Sci. 2 (2) (2020) 265–283.

Breno Serrano is a Ph.D. student at the Technical University of Munich, Germany. He obtained his M.Sc. degree from the Pontifical Catholic University of Rio de Janeiro, Brazil. His main research interests include combinatorial optimization and machine learning.

Thibaut Vidal holds the SCALE-AI Chair in Data-Driven Supply Chains and is a professor at the Department of Mathematics and Industrial Engineering (MAGI) of Polytechnique Montréal, Canada. He is also a member of CIRRELT and an adjunct professor at the Pontifical Catholic University of Rio de Janeiro, Brazil. His research interests include combinatorial optimization, heuristic search and interpretable machine learning, with applications to logistics and supply chain management, production management, resource allocation and information processing.