From your Block to our Block: How to Find Shared Structure between Stochastic Block Models over Multiple Graphs

Iiro Kumpulainen[1], Sebastian Dalleiger[2], Jilles Vreeken[3], Nikolaj Tatti[1]
[1]HIIT, University of Helsinki
[2]KTH Royal Institute of Technology
[3]CISPA Helmholtz Center for Information Security
{iiro.kumpulainen, nikolaj.tatti}@helsinki.fi, sdall@kth.se, jv@cispa.de

Abstract

Stochastic Block Models (SBMs) are a popular approach to modeling single real-world graphs. The key idea of SBMs is to partition the vertices of the graph into blocks with similar edge densities within, as well as between different blocks. However, what if we are given not one but multiple graphs that are unaligned and of different sizes? How can we find out if these graphs share blocks with similar connectivity structures? In this paper, we propose the shared stochastic block modeling (SSBM) problem, in which we model n graphs using SBMs that share parameters of s blocks. We show that fitting an SSBM is **NP**-hard, and consider two approaches to fit good models in practice. In the first, we directly maximize the likelihood of the shared model using a Markov chain Monte Carlo algorithm. In the second, we first fit an SBM for each graph and then select which blocks to share. We propose an integer linear program to find the optimal shared blocks and to scale to large numbers of blocks, we propose a fast greedy algorithm. Through extensive empirical evaluation on synthetic and real-world data, we show that our methods work well in practice.

Code — https://version.helsinki.fi/dacs/SharedSBM

Introduction

In many network settings, we are interested in what is similar and what is different between two or more graphs. Most existing methods, however, focus on quantifying similarity rather than qualifying what are the structures that are specific to or shared between the graphs. Examples of when we need more than just a measure of *how* similar graphs are, but we are specifically interested in characterizing *what* makes the graphs similar, include comparing brain scans of different patients; protein interaction networks of different species; social networks of different kinds; or comparing trade networks measured at different points of time. In some of these examples, the graphs will have aligned vertices, but often the graphs are unaligned or of different sizes. In this paper, we propose a method for finding shared connectivity structures between graphs that work in each of these settings.

We build our method upon the notion of the Stochastic Block Model (SBM), which is a popular approach to modeling single graphs (Peixoto 2019). The key idea of the SBM is to partition the vertices of a graph into blocks, modeling the edge likelihoods of nodes within a block as well as those of nodes in different blocks using Bernoulli variables. By optimizing likelihood, we can so obtain a partition such that the edge densities within each block are similar, and those between every pair of blocks are similar too.

Towards our goal of revealing shared structure, we extend the notion of the SBM to n graphs, modeling each with its own SBM, but, requiring these n models to share s blocks. That is, we model the edges of a shared block using the same distribution across all n graphs. This formulation allows the domain expert to flexibly explore what is similar (the s shared blocks) and what is different (the other blocks) between the n graphs. We dub this the Shared Stochastic Block Model or SSBM for short.

We show that fitting the optimal SSBM, as well as the sub-problem where we know the block memberships per graph, are both **NP**-hard, even for $n = 2$ graphs. Moreover, these problems are inapproximable unless $\mathbf{P} = \mathbf{NP}$. To fit good models in practice, we propose two approaches. In the first, we jointly search for the optimal blocks that should be shared. We show how we can use a Markov chain Monte Carlo (MCMC) approach with simulated annealing to search for the optimal assignment.

Our second approach for fitting a good model consists of two stages. In the first stage, we search only for the assignment of vertices into blocks. Once the vertex assignment is fixed, we then select which blocks should be shared. We show that this assignment can be done optimally with an integer linear program (ILP), but also a faster greedy approach where we select that shared block that reduces the log-likelihood the least. We consider and experiment with several strategies for the initial assignment of vertices: fitting a model to each graph individually, or using the assignment given from the first approach as the starting point.

Through an extensive set of experiments, we show that our methods work well in practice. On synthetic data we show that 1) we obtain the best results if we use MCMC with an agglomeration heuristic to find initial SBMs per graph, and then an ILP to find the s-SSBM, that 2) we can use BIC to recover the ground truth number of shared blocks, and that 3) the runtime of our approaches scales linearly in the number of edges. Through a case study, we show our method reveals easily interpretable knowledge of the similarities between brain scans of ADHD patients.

Preliminaries

Let $G = (V, E)$ be a graph with $|V| = N$ nodes and $|E| = M$ edges. In the standard stochastic block model, we have B blocks and a partition vector $\mathbf{b}$ whose entries $b_i \in \{1, 2, \ldots, B\}$ assign each node v_i in the graph to one of the blocks. In addition, we have a $B \times B$ matrix of probabilities $\mathbf{\Theta}$ where an entry θ_{ab} denotes the probability of an edge between a node in block a and another node in block b.

Let $\mathbf{A}$ denote the adjacency matrix of graph G with entries $A_{ij} \in \{0, 1\}$ indicating edges between nodes v_i and v_j in the graph. The likelihood of the graph according to the SBM is given by the Bernoulli distribution

$$P(G \mid \mathbf{\Theta}, \mathbf{b}) = \prod_{i,j,i \neq j} \theta_{b_i,b_j}^{A_{ij}} (1 - \theta_{b_i,b_j})^{1-A_{ij}} \ ,$$

where we assume that the graph is directed. In the case of an undirected graph, the product is over pairs of indices i, j with $i < j$ instead. To simplify the notation, we define C_{ij} as the number of edges between blocks i and j. Similarly, we define F_{ij} as the number of missing edges between i and j. The log-likelihood of the model is then

$$\log P(G \mid \mathbf{\Theta}, \mathbf{b}) = \sum_{i,j \in [B]} C_{ij} \log \theta_{ij} + F_{ij} \log (1 - \theta_{ij}) \ .$$

Problem Definition

We argue that stochastic block models are also valuable in settings with multiple graphs and can be used to compare connectivity patterns. Shared models may better explain common structures in the data than individual SBMs and require fewer parameters. To that end, we introduce the concept of shared blocks in SBMs. We then define associated optimization problems and show that they are **NP**-hard.

Assume that we have n graphs $G_1, \ldots, G_n$ with an SBM for each graph G_k with the partition vector $\mathbf{b}_k$ and the probability matrix $\mathbf{\Theta}_k$ with entries θ_{ij}^k. We then say that these SBMs have s shared blocks if we can select s blocks from each SBM such that the probabilities of edges between corresponding block pairs are identical in each SBM.

More formally, SBMs share s blocks if there are n injective mappings $\sigma_1, \ldots, \sigma_n$, where $\sigma_k : [s] \to [B_k]$ such that $\theta_{\sigma_k(i)\sigma_k(j)}^k$ does not depend on k. In other words, the different $\mathbf{\Theta}_k$ matrices must have identical $s \times s$ submatrices corresponding to the shared blocks. Note that regardless of having shared blocks between the SBMs, we can compute the log-likelihood for each graph as before, and the total log-likelihood for n graphs is

$$\begin{aligned}&\log P(G_1, \ldots, G_n \mid \{\mathbf{\Theta}_k, \mathbf{b}_k\}_k) \\ &\quad = \sum_{k=1}^{n} \log P(G_k \mid \mathbf{\Theta}_k, \mathbf{b}_k) \ . \end{aligned} \tag{1}$$

This gives rise to the optimization problem of inferring a maximum-likelihood stochastic block model with shared blocks. We formally define the problem as follows.

Problem 1 (SSBM)**.** *Given n graphs $G_1, \ldots, G_n$, each graph G_k with fixed number of blocks B_k, as well as a non-negative integer s, find partition vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ and probability matrices $\mathbf{\Theta}_k \in \mathbb{R}^{B_k \times B_k}$ with s shared blocks between them such that the log-likelihood $\log P(G_1, \ldots, G_n \mid \{\mathbf{\Theta}_k, \mathbf{b}_k\}_{k=1}^n)$ is maximized.*

One way to approach this problem is to find block assignments for each graph independently and then find which parameters to share between the SBMs such that the log-likelihood remains as high as possible. This leads us to the subproblem of inferring the shared SBM parameters while the assignment of vertices into blocks is fixed.

Problem 2 (SSBM-FIXED)**.** *Given n graphs $G_1, \ldots, G_n$ and n partition vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$, as well as a non-negative integer s, find probability matrices $\mathbf{\Theta}_k$ with s shared blocks between them such that the log-likelihood $\log P(G_1, \ldots, G_n \mid \{\mathbf{\Theta}_k, \mathbf{b}_k\}_{k=1}^n)$ is maximized.*

Note that the maximum-likelihood probability matrices $\mathbf{\Theta}_k$ for each graph G_k can be easily computed once the block assignments and shared blocks are known. More formally, let C_{ij}^k and F_{ij}^k denote the numbers of edges and missing edges for a pair of blocks $i, j \in [B_k]$. When i and j are not both shared between graphs, the maximum log-likelihood is achieved by

$$\theta_{ij}^k = \frac{C_{ij}^k}{C_{ij}^k + F_{ij}^k} \ . \tag{2}$$

Moreover, the optimal parameter between the ith and jth shared block in kth graph is

$$\theta_{\sigma_k(i)\sigma_k(j)}^k = \frac{\sum_{\ell=1}^n C_{\sigma_\ell(i)\sigma_\ell(j)}^\ell}{\sum_{\ell=1}^n C_{\sigma_\ell(i)\sigma_\ell(j)}^\ell + F_{\sigma_\ell(i)\sigma_\ell(j)}^\ell} \ . \tag{3}$$

Thus, to solve SSBM or SSBM-FIXED, it is sufficient to find the assignments of vertices into blocks as well as which blocks to share, and the probability matrices can then be computed according to Equations 2-3.

Next we show that both SSBM and SSBM-FIXED are **NP**-hard and inapproximable by a reduction from the well-known problem of finding cliques of size k.

Theorem 1. SSBM *and* SSBM-FIXED *are **NP**-hard, even for $n = 2$. Moreover, unless **P** = **NP**, there is no polynomial-time algorithm that always produces a solution for* SSBM *or* SSBM-FIXED *with a log-likelihood LLH such that $LLH \geq \alpha OPT$, where OPT is the log-likelihood of the optimal solution and $\alpha \geq 1$ is any constant.*

The second result states that SSBM and SSBM-FIXED are inapproximable. Note that $\alpha \geq 1$ makes sense since the log-likelihood of a Bernoulli SBM is always non-positive.

Proof. Let us focus first on SSBM. We will use a reduction from the **NP**-hard k-CLIQUE problem, which asks whether a given graph has a clique of size k. Given a graph G and integer k as an instance of k-CLIQUE, we construct an instance of SSBM with two graphs by setting $G_1 = G$ and G_2 as a complete graph with the same number of vertices. In addition, set $s = k$, and set $B_1 = B_2 = |V(G)|$.

We claim that G has a k-clique if and only if the optimal log-likelihood equals 0. Assume that there is a k-clique. Set the partition vectors $\mathbf{b}_1, \mathbf{b}_2$ such that each vertex belongs to a different block. Set $\mathbf{\Theta}_i$ to be the adjacency matrix of G_i. The log-likelihood in this case is 0 and the clique provides k shared blocks.

Assume that the optimal log-likelihood is 0. This is only possible if each parameter in $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$ is 0 or 1. Let W be the vertices in G_1 in blocks shared with G_2. Since k blocks are shared, we have $|W| \geq k$. Since G_2 is fully connected, Eq. 3 implies that any parameter between two shared blocks cannot be 0, so it must be 1. Consequently, any two shared blocks are fully connected in G_1, that is, W is a clique. The case for SSBM-FIXED is essentially the same, except now we fix the vectors $\mathbf{b}_1, \mathbf{b}_2$ such that each vertex is in its own block already in the construction.

In summary, solving SSBM or SSBM-FIXED also solves k-CLIQUE, which means that both problems are **NP**-hard. Furthermore, the optimal objective value is 0, which means that an approximation algorithm with any multiplicative approximation guarantee would have to find the exact optimal solution thus solving k-CLIQUE. □

Algorithms

Our next step is to consider techniques to fit our model. We will first consider a method that solves the main problem SSBM with a Markov chain Monte Carlo (MCMC) approach. Next, we consider a two-step approach in which we first find and fix the block assignments and then find the shared blocks. The latter step can be then optimized with an ILP or with a greedy algorithm. We evaluate different group assignment strategies in our experiments.

MCMC Algorithm for SSBM

Although exact inference of a maximum-likelihood SBM is computationally infeasible, there are efficient algorithms to find an SBM that fits the data well in practice, such as the MCMC approach (Peixoto 2019). We modify this approach to suit our model. For each graph G_k, we start with a randomly initialized partition vector $\mathbf{b}_k$ and assign the first s blocks in each graph to be shared such that blocks with corresponding indices are mapped to each other, that is, $\sigma_k(i) = i$ for $i \in [s]$. At each step of the algorithm, we propose a new block assignment $\mathbf{b}'_k$ where one vertex is moved to a new block.

Rather than proposing new blocks uniformly randomly, we use a heuristic described by Peixoto (2019) where moves to neighboring blocks become more likely. This move proposal is then accepted with probability

$$\min\left(1, \frac{P(\mathbf{b}'_k \mid G_1, \ldots, G_n)^\beta P(\mathbf{b}_k \mid \mathbf{b}'_k)}{P(\mathbf{b}_k \mid G_1, \ldots, G_n)^\beta P(\mathbf{b}'_k \mid \mathbf{b}_k)}\right),$$

where β is an inverse temperature parameter used for simulated annealing. Computing the acceptance probability boils down to computing the log-likelihood given in Eq. 1 with the maximum likelihood parameters given in Eqs. 2–3.

Instead of computing the full log-likelihood for each move proposal, we can compute the change in log-likelihood more efficiently by keeping track of the counts of edges between blocks along with the counts of vertices in each block. Then each move proposal and update can be performed in time $\mathcal{O}(\deg(v) + B_k)$, where $\deg(v)$ represents the degree of the vertex v being moved and B_k is the number of blocks in that graph. Hence, one full iteration over all the nodes in all graphs takes time $\mathcal{O}(\sum_{k=1}^n |E_k| + |V_k| B_k)$, where $|E_k|$ and $|V_k|$ are the numbers of edges and vertices in graph G_k.

The number of graphs n is a small constant in our experiments, so we use a simpler implementation without storing and continuously updating the counts of total edges or vertices for shared blocks. This leads to a running time of $\mathcal{O}(n \sum_{k=1}^n |E_k| + |V_k| B_k)$ per iteration over all the nodes.

Integer Linear Program for SSBM-FIXED

Next, we consider solving SSBM-FIXED. The number of blocks in an SBM is typically much smaller than the number of vertices. In our case, when the numbers of blocks B_k are small, we can solve SSBM-FIXED exactly using an integer linear program (ILP).

Solving SSBM-FIXED amounts to finding s injective functions σ_k that maximize the log-likelihood. To express this problem as an ILP, we need to introduce additional notation. Assume that we are given σ_k and fix $i \in [s]$. Then define a vector r of length n with $r_k = \sigma_k(i)$, that is, r_k indicates the ith shared block in graph G_k. Instead of searching for σ_k, we will search for such vectors r. For this purpose, let us define the space of such vectors as $\mathcal{T} = [B_1] \times [B_2] \times \cdots \times [B_k]$.

Next, we express the log-likelihood as a linear function over certain variables and then impose conditions on those variables so that the program matches solving SSBM-FIXED. To that end assume that i or j are not shared blocks in G_k. Then the associated log-likelihood between these blocks is

$$U_{ij}^k = C_{ij}^k \log \theta_{ij}^k + F_{ij}^k \log(1 - \theta_{ij}^k) \ ,$$

where θ_{ij}^k is given in Eq. 2. On the other hand, let $r, t \in \mathcal{T}$ be two vectors indicating shared blocks. Then the associated log-likelihood between these shared blocks is

$$Q_{rt} = \sum_{k=1}^n C_{r_k t_k}^k \log \theta_{rt} + F_{r_k t_k}^k \log(1 - \theta_{rt}) \ ,$$

where

$$\theta_{rt} = \frac{\sum_{k=1}^n C_{r_k t_k}^k}{\sum_{k=1}^n C_{r_k t_k}^k + F_{r_k t_k}^k} \ .$$

To state our ILP, we need the following two sets of binary variables. The first set z_{kij}, where $k \in [n]$ and $i, j \in [B_k]$, indicates whether blocks i and j are shared in G_k. The second set w_{rt}, where $r, t \in \mathcal{T}$, indicates if r and t correspond to shared blocks. The log-likelihood is then equal to

$$\sum_{r,t \in \mathcal{T}} w_{rt} Q_{rt} + \sum_{k=1}^n \sum_{i,j=1}^{B_k} (1 - z_{kij}) U_{ij}^k \ . \qquad (4)$$

Next, we need to impose constraints on z_{kij} and w_{rt}. To that end, we introduce binary variables $x_{ki\ell}$, indicating that for graph G_k, the block $i \in [B_k]$ is the shared ℓth block. In other words, $x_{ki\ell}$ will correspond to having $\sigma_k(\ell) = i$. To guarantee this correspondence we require that each graph has exactly one ℓth shared block by imposing

$$\sum_{i=1}^{B_k} x_{ki\ell} = 1, \qquad k \in [n],\ \ell \in [s]\ . \tag{5}$$

Now, we introduce an auxiliary binary variable y_{ki}, indicating that the block $i \in [B_k]$ for graph G_k is shared. We achieve this by requiring that

$$y_{ki} = \sum_{\ell=1}^{s} x_{ki\ell}, \qquad k \in [n],\ i \in [B_k]\ . \tag{6}$$

Note that since y_{ki} are binary, the corresponding functions σ_k are forced to be injective. By definition, $z_{kij} = 1$ if and only if $y_{ki} = 1$ and $y_{kj} = 1$. We achieve this by requiring that

$$\begin{aligned} z_{kij} \le y_{ki},\ z_{kij} \le y_{kj}, \quad & k \in [n],\ i,j \in [B_k]\ , \\ z_{kij} \ge y_{ki} + y_{kj} - 1, \quad & k \in [n],\ i,j \in [B_k]\ . \end{aligned} \tag{7}$$

To constrain w_{rt} we introduce binary variables $c_{r\ell}$ that indicate that from each graph G_k, the block r_k is mapped to the shared ℓth block. We achieve this by requiring

$$\begin{aligned} c_{r\ell} \le x_{kr_k\ell}, \qquad & r \in \mathcal{T},\ k = [n],\ \ell \in [s]\ , \\ c_{r\ell} \ge 1 + \sum_{k=1}^{n} (x_{kr_k\ell} - 1), \qquad & r \in \mathcal{T}\ ,\ \ell \in [s]\ . \end{aligned} \tag{8}$$

In addition, we introduce binary variables d_r indicating that $\sigma_k(r_k) = \ell$ for some $\ell \in [s]$. This is done by requiring

$$d_r = \sum_{\ell=1}^{s} c_{r\ell}, \qquad r \in \mathcal{T}\ . \tag{9}$$

Finally, by definition, $w_{rt} = 1$ if and only if $d_r = 1$ and $d_t = 1$. We achieve this by requiring that

$$w_{rt} \le d_r,\ w_{rt} \le d_t,\ w_{rt} \ge d_r + d_t - 1 \quad r,t \in \mathcal{T}\ . \tag{10}$$

Note that Eq. 4 is linear since U_{ij}^k and Q_{rt} are constants. Moreover, the constraints in Eqs. 5–10 are linear. Hence, maximizing Eq. 4 subject to Eqs. 5–10 can be done with an ILP. Finally, the constraints guarantee that we can construct σ_k from $c_{r\ell}$ by setting $\sigma_k(\ell) = r_k$ and that the cost in Eq. 4 matches to the log-likelihood.

The number of variables and equations are in $\mathcal{O}\left(|\mathcal{T}|^2\right) = \mathcal{O}\left(\prod B_k^2\right)$, which grows exponentially with the number of graphs. Moreover, solving an ILP can be done in $\log(2h)^{\mathcal{O}(h)}$ time, where h is the number of variables (Reis and Rothvoss 2023). However, the ILP does not depend on the number of nodes or edges in the graphs, and we can solve it efficiently in practice when the number of graphs is small. In addition, we show an alternative ILP in the Appendix, which does not have an exponential number of variables but does depend on the number of nodes in the graphs. For our experiments, the above version is adequate.

Greedy Algorithm for SSBM-FIXED

As an alternative to integer linear programming approaches, we propose a greedy algorithm that iteratively picks shared blocks consisting of one block from each graph. In each iteration, the greedy algorithm chooses the shared block that yields the highest log-likelihood with the previously chosen shared blocks. The pseudocode for the greedy algorithm is given in Algorithm 1.

Algorithm 1 Greedy algorithm for SSBM-FIXED.

```
Input: number of shared blocks $s$, graphs $G_1, \dots, G_n$ and
    their block assignments.
1: $S \leftarrow \emptyset$; $\mathcal{T} \leftarrow [B_1] \times \cdots \times [B_n]$
2: while $|S| < s$ do
3:     $r \leftarrow$ vector in $\mathcal{T}$ with the smallest decrease in log-
           likelihood when added to shared blocks $S$
4:     Add $r$ to $S$
5:     Delete any vector from $\mathcal{T}$ sharing blocks with $r$
6: return $S$
```

To compute the change in log-likelihood for a single candidate shared block, we update the parameters for edge probabilities between it and the other currently shared blocks. Calculating the log-likelihood change for each of the n graphs for the at most $\mathcal{O}(s)$ updated parameters thus takes time $\mathcal{O}(ns)$. In each of the s iterations, we consider $|\mathcal{T}|$ candidates, which leads to a total running time of $\mathcal{O}\left(ns^2 B_1 \cdots B_n\right)$. Note that this is polynomial when the number of graphs n is constant, and does not depend on the number of edges or vertices. However, the number of possible shared blocks grows exponentially with the number of graphs, so future work may design different algorithms for cases where n is large.

Related Work

The Stochastic Block Model (SBM) has been extensively studied and adapted to capture various complexities in network structures. The classic SBM, introduced by Holland, Laskey, and Leinhardt (1983), provides a foundational framework for modeling community structures within networks through probabilistic block assignments. This model has since been expanded to address different types of network characteristics, including degree heterogeneity through the Degree-Corrected SBM (Karrer and Newman 2011), the incorporation of temporal dynamics into SBMs enabling the analysis of evolving networks (Matias and Miele 2017), and the ability to model multi-layer networks with SBMs (Han, Xu, and Airoldi 2015).

Sharing. Sharing in the context of the SBM framework has focused on capturing overlapping communities. Airoldi et al. (2008) developed the mixed-membership stochastic block model (MMSBM), which allows nodes to belong to multiple communities. While Consensus SBM (CSBMs) (Faskowitz et al. 2018) aim to create a single unified SBM from a collection of hundreds of individual SBMs, they cannot distinguish between shared and specific blocks. This approach is informed by methods developed for community detection in multiplex networks and overlapping community models. The work of Yang and Leskovec (2012) on overlapping community detection models how nodes can belong to multiple communities simultaneously, which they efficiently put into practice using Non-Negative Matrix Factorization (Yang and Leskovec 2013). Recent work by Cohen-Addad et al. (2024) introduces multi-view SBMs to estimate a shared SBM jointly for multiple graphs. To date, research on fine-grained block-sharing in the context of multiple graphs is lacking.

Inferring. Markov Chain Monte Carlo (MCMC) methods are widely used to estimate posterior distributions over block assignments and model parameters (Peixoto 2014). Latouche, Birmele, and Ambroise (2012) pioneered the use of variational inference (Blei, Kucukelbir, and McAuliffe 2017) to efficiently approximate the posterior distributions in large-scale SBM applications (Tabouy, Barbillon, and Chiquet 2020), enhancing computational feasibility and offering a more scalable inference. Peixoto (2017) further refined these techniques by employing nonparametric Bayesian inference to better capture complex community structures without requiring predefined numbers of blocks. Additionally, clustering techniques, such as spectral clustering (Lei and Rinaldo 2015), and graph partition heuristics, such as Kernighan-Lin (Kernighan and Lin 1970), have been used to efficiently estimate block structure. Funke and Becker (2019) provide a comparative study of the experimental efficacy of many inference algorithms.

Comparing Graphs. Graph kernels (Vishwanathan et al. 2010; Kriege, Johansson, and Morris 2020), such as the graphlet kernel (Shervashidze et al. 2009), the Weisfeiler-Lehman (WL) kernel (Shervashidze et al. 2011), and Wasserstein WL kernel (Togninalli et al. 2019), can be used to compare structural properties of *pre-existing* communities and blocks, but fail to capture the high-level commonalities between distributions and blocks that are specific to our use case. While divergences and metrics over distributions, such as Wasserstein kernels (De Plaen, Fanuel, and Suykens 2020), may perform better for many needs, they also require pre-existing communities which we aim to find.

Experiments

Next, we present our experimental evaluation.

Methodology. To discover block assignments—while ignoring the shared block requirement—we consider three approaches: *Single* independently fits a single SBM to graph using MCMC. The algorithm samples a new assignment by moving a random node between blocks (Peixoto 2019). Alternatively, *Multilevel* fits single SBMs for each graph using an MCMC algorithm with an agglomerative heuristic that starts with all nodes in separate blocks and uses a series of block merges and individual node movements to reach a stable node assignment for the desired number of blocks (Peixoto 2014). Third, *ML+Single* first runs the *Multilevel* algorithm as a starting point for *Single* to refine the node assignment further.

To find block assignments with shared blocks, we consider two approaches: *Shared* fits SBMs with shared parameters using the MCMC algorithm for SSBM. Finally, *ML+Shared* uses *Multilevel* similarly as a starting point for *Shared*. However, since *Shared* assumes that the first s blocks in each graph share parameters, we use the ILP, given in Eqs. 4–10, to select which blocks to share and reorder them to be first. We use the last 3 methods directly to find SBMs with shared blocks. In addition, we use all 5 methods to compare the ILP and Greedy algorithms for SSBM-FIXED, with a *Random* baseline that randomly selects the shared blocks.

We implemented *Single* and *Shared* in C++ using graph-tool,[1] which includes *Multilevel*. Our experiments and other algorithms are implemented in Python, and we use Gurobi for solving the ILP instances.[2] The experiments were run on an AMD EPYC 7452 32-core processor in a high-performance computing environment. All code for reproducing the experiments is publicly available.[3]

Experiments on Synthetic Graphs

For each experiment on synthetic data, we create n graphs using stochastic block models with 200 to 500 vertices with a fixed number of blocks and shared blocks. The vertices are randomly partitioned into the blocks, edge probabilities between blocks are sampled from a Beta distribution with parameters $\alpha = 0.5, \beta = 1.0$, and the probabilities for shared blocks are set to be equal. Each pair of vertices then has an edge between them based on the edge probability between the corresponding blocks.

We evaluate how accurately our algorithms for SSBM-FIXED can find the ground truth shared blocks by computing the Shared ARI, which we define as the adjusted Rand index (ARI) (Hubert and Arabie 1985) score between an inferred binary partition of vertices into shared or non-shared blocks and the corresponding ground truth partition.

Random Noise. Since solving SSBM via SSBM-FIXED requires a separate inference algorithm to provide block assignments that may be imperfect, we assess how mistakes in the input block assignment reflect in the Shared ARI scores for our algorithms for optimizing the shared blocks. We add noise to the ground truth block assignment by independently moving vertices to random blocks with increasing probability. Figure 1 shows that ILP outperforms Greedy and Random in finding the ground truth shared blocks when provided with the true assignment of blocks and how increasing levels of random noise slowly degrades performance.

Block Assignment Accuracy. We then compare the algorithms for inferring the block assignments by using the ARI score to measure the similarity between the inferred and ground truth partitions of vertices into blocks. Figure 2 (left) shows the average partition ARI over 10 runs on 3 graphs with 500 vertices and 5 blocks each, with 3 shared blocks between the graphs. All algorithms find partitions highly similar to the ground truth, and we note that *Multilevel* is more consistent at finding partitions close to the ground truth than the simpler *Single* and *Shared* algorithms. *ML+Shared* performs the best by slightly improving upon *Multilevel* and

[1]https://graph-tool.skewed.de/

[2]https://www.gurobi.com/solutions/gurobi-optimizer/

[3]https://version.helsinki.fi/dacs/SharedSBM/

finding the ground truth partition nearly every time.

Accuracy of Discovering Shared Blocks. Combining the inference algorithms with the algorithms for selecting which blocks to share, we obtain a pipeline for solving SSBM. Figure 2 (right) shows the Shared ARI scores for different pairs of algorithms for inferring the block assignments and for choosing the shared blocks. When selecting the s first blocks to be shared, *Shared* is notably better than random, but the multi-step approaches involving *Multilevel* perform better, with results close to ground truth when using ILP. Notably, the *ML+Shared* combination slightly improves upon *Multilevel*, while *ML+Single* does not.

Information Gain. To evaluate if sharing the parameters between the blocks leads to a better model, we use the Bayesian information criterion (BIC) for model selection. In Figure 3, we compare the differences in BIC and log-likelihood for *Multilevel*, *ML+Single*, and *ML+Shared* when using no shared blocks compared to when using the ILP for selecting shared blocks and sharing parameters. When sharing blocks, the BIC slightly decreases for all three, implying that having shared blocks leads to a better model despite small decreases in the log-likelihood. The improvement in BIC is highest for *ML+Shared*, whereas for *ML+Single* the loss in log-likelihood nearly offsets the information gain from decreasing the number of parameters.

Inferring the Number of Shared Blocks. While our experiments use fixed numbers of blocks and shared blocks, in practical applications these may have to be inferred from the data. Figure 4 shows the BIC scores for *ML+Shared* with ILP assuming different numbers of shared blocks for two graphs with 400 vertices and 8 blocks each and 3 shared blocks between them. The BIC score is lowest when the assumed number of shared blocks matches the ground truth, showing we can infer it from the data. In future work, the numbers of blocks and shared blocks could be inferred automatically using Bayesian inference (Peixoto 2019).

Running Time. We compare the running times of ILP, Greedy, and Random for an increasing number of graphs, with each graph having 4 blocks, and the number of shared blocks set to 2. We show the results in Fig. 5. Greedy is significantly faster than ILP, taking less than two minutes for six graphs. The exponential trend matches the theory.

Finally, we analyze the total running times for the different MCMC-based inference algorithms. We run *Single* and *Shared* for 500 iterations over all vertices, while *Multilevel* uses graph-tool's default parameters. Figure 6 shows the running times as a function of the total number of edges in two graphs with 4 blocks each and 2 shared blocks. The running times of all algorithms scale linearly with the edge count, but *Single* and *Shared* are significantly faster, and combining them with *Multilevel* adds almost no overhead.

Use Case: Brain Networks

To assess if our algorithms can find shared block structures that can be meaningful in practice, we test them on brain network samples from the NITRC ADHD-200 (Bellec et al. 2017) resting-state dataset available in the nilearn library.[4]

[4]https://nilearn.github.io/

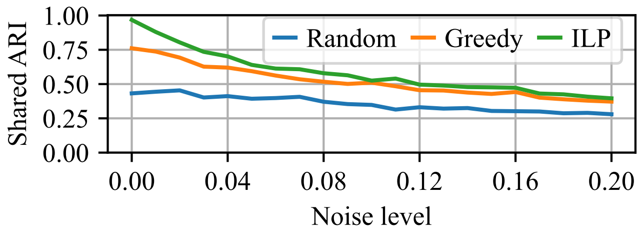

Figure 1: Comparison of ARI scores for the partitions of vertices into shared or non-shared blocks by different algorithms for selecting shared blocks, with increasing levels of random noise in the input block assignment.

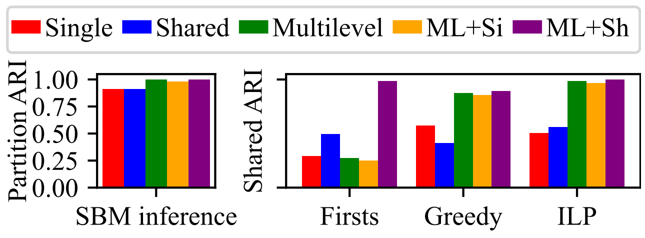

Figure 2: (Left): Average ARI scores measuring the similarity between the inferred partitions of vertices into blocks with the ground truth partitions. (Right): ARI scores between ground truth and inferred partitions of vertices into shared or non-shared blocks for different algorithms for inferring the block assignments paired with different methods for choosing the shared blocks. The Firsts method represents choosing the first s blocks in each graph to be shared, which is effectively random for *Single*, *Multilevel*, and *ML+Single*.

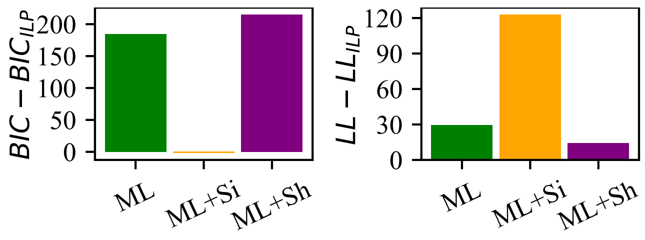

Figure 3: Decrease in BIC (higher is better) and in log-likelihood (lower is better) for *Multilevel* (ML), *ML+Single* (ML+Si), and *ML+Shared* (ML+Sh) when using ILP shared blocks compared to not using any shared blocks.

We obtain one graph per patient, with each graph having 1000 nodes representing brain regions of interest, and we construct an undirected edge whenever the functional connectivity between the corresponding brain regions has a correlation coefficient of at least 0.5.

We note that the brain networks between patients vary greatly. In Figure 7 we highlight a set of three patients with attention deficit and hyperactivity disorder (ADHD) whose brains display similar connectivity patterns. We set the number of blocks to 10 per graph of which we seek to share 2. By using the Greedy algorithm, *ML+Shared* identifies shared

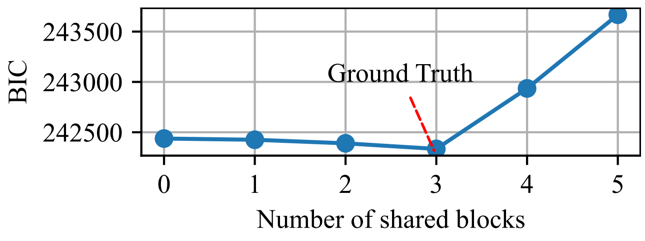

Figure 4: BIC scores for different numbers of shared blocks for *ML+Shared* with ILP. The BIC is lowest at 3 shared blocks matching the ground truth value used for generating the graphs.

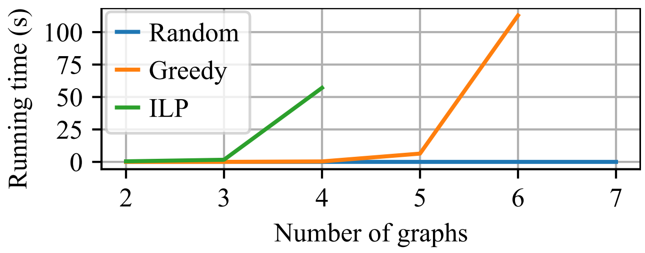

Figure 5: Average running times of different algorithms for optimizing which blocks to share as a function of the number of graphs.

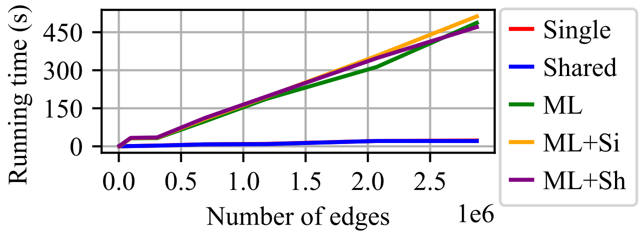

Figure 6: Average running times of different SBM inference algorithms as a function of the number of edges in randomly generated synthetic graphs.

blocks that correspond to approximately the same brain region. Being able to discover shared structures solely based on the connectivity patterns, our method does not use the dataset-inherent node-alignments.

Use Case: Wikipedia Networks

To test the scalability of our methods on real-world networks, we run an experiment on the extensive English and Chinese Wikipedia link networks from the Koblenz network collection (Kunegis 2013). Each node represents an article on Wikipedia with directed edges representing links from one article to another. The graph for English Wikipedia has 13 593 032 nodes and 437 217 424 edges, while the Chinese Wikipedia has 1 786 381 nodes and 72 614 837 edges. Since many Wikipedia articles exist in both languages, we expect that the two networks include similar connectivity patterns that shared blocks could represent.

For the large Wikipedia networks, *Multilevel* took too long to execute, exceeding our time limit of 8 hours. However, with 100 iterations over all nodes, *Single* and *Shared* were completed in approximately 5.5 hours. With the number of blocks set at 20 per graph and having 2 shared blocks between the graphs, the ILP took prohibitively long while the Greedy algorithm took less than one second. *Shared* resulted in a better log-likelihood than *Single* (-5.036×10^9 versus -5.050×10^9), and the BIC scores improved in both cases when incorporating the shared blocks chosen by Greedy compared to when not sharing any parameters between blocks. This suggests our models with shared blocks may fit the data better. In this instance, further analysis of the shared blocks would be difficult due to the lack of node labels in the dataset.

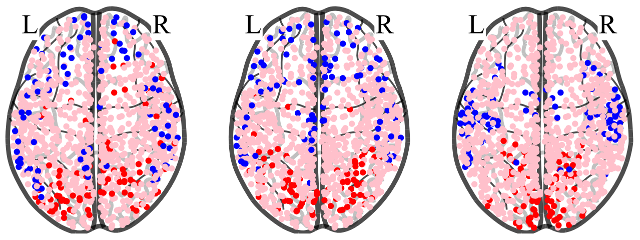

Figure 7: Brain networks for three patients with ADHD at the Oregon Health & Science University (ages 9, 9, and 11). Brain regions highlighted in red and blue correspond to shared blocks between the graphs having similar connectivity patterns identified by *ML+Shared* with Greedy.

Conclusions

In this paper, we studied the problem of common structure in multiple graphs using the stochastic block model. We showed that the related optimization problems are **NP**-hard and inapproximable. To fit the model we considered several approaches, based on a Markov chain Monte Carlo (MCMC) approach, integer linear programming, and greedy optimization. In our experiments, the best approach was to first discover block structure using MCMC, and then refine the result by optimizing the shared blocks with an integer linear program. We showed that our algorithms can find shared structure between graphs that may be of practical interest and that our methods can be applied to large graphs with over 10^8 edges.

Developing even better or faster algorithms for finding shared blocks, especially when the number of input graphs is large, is an interesting direction for future work. Alternatively, other definitions for shared blocks may be considered, such as allowing the shared blocks to repeat within the graphs by using many-to-many instead of injective mappings to indicate which blocks are shared. Furthermore, the models for shared blocks could be used in Degree-Corrected SBMs—for example, by requiring that the distributions of node degrees are also shared—or other variants of the SBM.

Acknowledgements

This research is supported by the Academy of Finland project MALSOME (343045). This research is supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

Airoldi, E. M.; Blei, D.; Fienberg, S.; and Xing, E. 2008. Mixed membership stochastic blockmodels. *Advances in Neural Information Processing Systems*, 21.

Bellec, P.; Chu, C.; Chouinard-Decorte, F.; Benhajali, Y.; Margulies, D. S.; and Craddock, R. C. 2017. The Neuro Bureau ADHD-200 Preprocessed repository. *NeuroImage*, 144: 275–286. Data Sharing Part II.

Blei, D. M.; Kucukelbir, A.; and McAuliffe, J. D. 2017. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518): 859–877.

Cohen-Addad, V.; d'Orsi, T.; Lattanzi, S.; and Nasser, R. 2024. Multi-View Stochastic Block Models. In *Forty-first International Conference on Machine Learning*.

De Plaen, H.; Fanuel, M.; and Suykens, J. A. 2020. Wasserstein exponential kernels. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–6. IEEE.

Faskowitz, J.; Yan, X.; Zuo, X.-N.; and Sporns, O. 2018. Weighted stochastic block models of the human connectome across the life span. *Scientific reports*, 8(1): 12997.

Funke, T.; and Becker, T. 2019. Stochastic block models: A comparison of variants and inference methods. *PloS one*, 14(4): e0215296.

Han, Q.; Xu, K.; and Airoldi, E. 2015. Consistent estimation of dynamic and multi-layer block models. In *Proceedings of the 32nd International Conference on Machine Learning*, 1511–1520. PMLR.

Holland, P. W.; Laskey, K. B.; and Leinhardt, S. 1983. Stochastic blockmodels: First steps. *Social Networks*, 5(2): 109–137.

Hubert, L.; and Arabie, P. 1985. Comparing partitions. *Journal of classification*, 2: 193–218.

Karrer, B.; and Newman, M. E. 2011. Stochastic blockmodels and community structure in networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 83(1): 016107.

Kernighan, B. W.; and Lin, S. 1970. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2): 291–307.

Kriege, N. M.; Johansson, F. D.; and Morris, C. 2020. A survey on graph kernels. *Applied Network Science*, 5: 1–42.

Kunegis, J. 2013. Konect: the koblenz network collection. In *Proceedings of the 22nd international conference on world wide web*, 1343–1350.

Latouche, P.; Birmele, E.; and Ambroise, C. 2012. Variational Bayesian inference and complexity control for stochastic block models. *Statistical Modelling*, 12(1): 93–115.

Lei, J.; and Rinaldo, A. 2015. Consistency of spectral clustering in stochastic block models. *The Annals of Statistics*, 215–237.

Matias, C.; and Miele, V. 2017. Statistical clustering of temporal networks through a dynamic stochastic block model. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(4): 1119–1141.

Peixoto, T. P. 2014. Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 89(1): 012804.

Peixoto, T. P. 2017. Nonparametric Bayesian inference of the microcanonical stochastic block model. *Phys. Rev. E*, 95: 012317.

Peixoto, T. P. 2019. Bayesian stochastic blockmodeling. *Advances in Network Clustering and Blockmodeling*, 289–332.

Reis, V.; and Rothvoss, T. 2023. The subspace flatness conjecture and faster integer programming. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, 974–988. IEEE.

Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9).

Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. 2009. Efficient graphlet kernels for large graph comparison. In van Dyk, D.; and Welling, M., eds., *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, 488–495. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR.

Tabouy, T.; Barbillon, P.; and Chiquet, J. 2020. Variational inference for stochastic block models from sampled data. *Journal of the American Statistical Association*, 115(529): 455–466.

Togninalli, M.; Ghisu, E.; Llinares-López, F.; Rieck, B.; and Borgwardt, K. 2019. Wasserstein weisfeiler-lehman graph kernels. *Advances in neural information processing systems*, 32.

Vishwanathan, S. V. N.; Schraudolph, N. N.; Kondor, R.; and Borgwardt, K. M. 2010. Graph kernels. *The Journal of Machine Learning Research*, 11: 1201–1242.

Yang, J.; and Leskovec, J. 2012. Community-affiliation graph model for overlapping network community detection. In *2012 IEEE 12th International Conference on Data Mining*, 1170–1175. IEEE.

Yang, J.; and Leskovec, J. 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM International Conference on Web Search and Data Mining*, 587–596.

Appendix — Alternative Integer Linear program for SSBM-FIXED

In this section we will describe alternative definition for ILP. This definition avoids having exponential number of parameters but it is most likely impractical due to a large number of parameters required.

The key observation is that the optimal parameters are rational numbers of certain forms. Let Λ be the set of all rational numbers x/y, where x, y are integers such that

$$0 \leq x \leq y \leq \sum_{k=1}^{n} \binom{|V(G_k)|}{2}.$$

We will index Λ with a, that is, we will write $\Lambda = \{\lambda_a\}$. It follows that Λ is of polynomial size and the optimal parameters are in Λ.

In our ILP we will map each block in each graph to an index. Let x_{kir} be a binary variable indicating that the ith block in G_k is mapped to index r. We will use the first s indices as shared, while the remaining indices can be unique. Consequently, we need $q = s + \sum (B_k - s)$ indices, at most.

We force that that each block is mapped to one index with

$$\sum_{r=1}^{q} x_{kir} = 1, \qquad k \in [n], i \in [B_k]. \qquad (11)$$

We also force that each graph has s shared blocks with

$$\sum_{i=1}^{B_k} \sum_{r=1}^{s} x_{kir} = s, \qquad k \in [n] \qquad (12)$$

Next we introduce a binary variable $z_{r\ell a}$ which indicates that block pair with indices (r, ℓ) uses λ_a. Every block pair should use one λ_a which we force by

$$\sum_{a} z_{r\ell a} = 1, \qquad r, \ell \in [q]. \qquad (13)$$

Finally, we introduce a binary variable y_{kija} that states that block pair (i, j) in graph G_k uses λ_a. To make y consistent with x and z, we require that

$$y_{kija} \geq \sum_{r,\ell=1}^{q} x_{kir} + x_{kj\ell} + z_{r\ell a} - 2 \qquad (14)$$

that is $y_{kija} = 1$ if block i is mapped to index r and block j is mapped to index ℓ and this index pair using λ_a.

Finally, we maximize

$$\sum_{k=1}^{n} \sum_{i,j=1}^{B_k} \sum_{a} y_{kija} (C_{ij}^{k} \log \lambda_a + F_{ij}^{k} \log(1 - \lambda_a)). \qquad (15)$$

We should point out that we did not need to specify upper bounds for y_{kija} (in addition to the lower bound given in Eq. 14) since the logarithm terms in Eq. 15 are negative.

The constraints and the objective in Eqs. 11–15 are linear so the problem is solvable with ILP. The number of needed variables and equations is in

$$\mathcal{O}\left((\sum_{k=1}^{n} B_k)^2 \sum_{k=1}^{n} |V(G_k)|^2 \right).$$