

Estimation of partially known Gaussian graphical models with score-based structural priors

Martín Sevilla
Rice University, USA

Antonio G. Marques
King Juan Carlos University, Spain

Santiago Segarra
Rice University, USA

Abstract

We propose a novel algorithm for the support estimation of partially known Gaussian graphical models that incorporates prior information about the underlying graph. In contrast to classical approaches that provide a *point estimate* based on a maximum likelihood or a maximum a posteriori criterion using (simple) priors *on the precision matrix*, we consider a prior *on the graph* and rely on annealed Langevin diffusion to generate *samples from the posterior distribution*. Since the Langevin sampler requires access to the score function of the underlying graph prior, we use graph neural networks to effectively estimate the score from a graph dataset (either available beforehand or generated from a known distribution). Numerical experiments demonstrate the benefits of our approach.

1 INTRODUCTION

Graphical models are useful probabilistic tools represented by graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes \mathcal{V} encode random variables, and edges \mathcal{E} encode information regarding the variables' joint distribution. Markov random fields (MRFs) – an important class of graphical models – offer the attractive property that the absence of an edge between two nodes means that the two associated random variables are conditionally independent given the rest (Bishop and Nasrabadi, 2006). When the distribution is a multivariate Gaussian, the MRF is said to be a Gaussian Markov random field (GMRF) or Gaussian graphical model (GGM) (Rue and Held, 2005). GGMs have been used to model complex relationships in a wide variety of disciplines, with relevant

examples including gene interactions (Dobra et al., 2004; Wang et al., 2020), spectrometric data (Codazzi et al., 2022), metabolic association networks (Tan et al., 2017), macroeconomic growth (Dobra et al., 2010), and social networks (Li et al., 2020).

The success of GGMs in such a broad range of datasets stems partly from their intuitive interpretability. To be specific, let $\mathbf{A} \in \{0,1\}^{n \times n}$ be the adjacency matrix of \mathcal{G} , with $n = |\mathcal{V}|$, and let $\mathbf{x} \in \mathbb{R}^n$ be a random vector such that $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$. We then say that \mathbf{x} is a GGM with respect to \mathcal{G} if and only if the precision matrix $\Theta = \Sigma^{-1}$ and \mathbf{A} have the same support (i.e., the same zero pattern) (Rue and Held, 2005). As a result, the graph associated with a GGM can be estimated from the non-zero values of Θ .

Related work and limitations. The problem of estimating the precision matrix Θ (and, consequently, its support) is classically known as *covariance selection* (Dempster, 1972). Specifically, given a set of k observations $\mathbf{x}_1, \dots, \mathbf{x}_k$, each of dimension n , the goal is to recover the $n \times n$ matrix Θ and, in particular, $\text{supp}(\Theta)$, which reveals the conditional independence relationships in the GGM. Given that $p(\mathbf{x} | \Theta)$ has a closed-form expression for a GGM, a straightforward approach is to find the maximum likelihood (ML) estimator for Θ , which is given by the inverse of the *sample* covariance \mathbf{S} (Casella and Berger, 2021). However, \mathbf{S}^{-1} typically does not contain exact zero entries. Hence, one common technique is thresholding \mathbf{S}^{-1} (Qiu and Liyanage, 2019), but this may yield unsatisfactory results if the threshold is not adequately chosen or if the number of observations k is limited.

More sophisticated techniques generally propose *penalties* in optimization problems that use the log-likelihood as cost function (Tsai et al., 2022; Williams, 2020). From a Bayesian standpoint, these penalties can be considered prior distributions $p(\Theta)$. For any penalty function $P(\Theta)$, there is an associated prior distribution $p(\Theta) \propto \exp(-P(\Theta))$ such that the penal-

ized ML estimation boils down to

$$\begin{aligned}\Theta_{\text{est}} &= \underset{\Theta \succeq 0}{\operatorname{argmax}} p(\mathbf{x}_1, \dots, \mathbf{x}_k \mid \Theta) p(\Theta) \\ &= \underset{\Theta \succeq 0}{\operatorname{argmax}} \log \det \Theta - \operatorname{tr}(\mathbf{S}\Theta) - P(\Theta),\end{aligned}\quad (1)$$

with the most popular penalty being $P(\Theta) = \lambda \sum_{i \neq j} |\Theta_{ij}|$, which translates to a Laplace prior from a Bayesian perspective. This penalty encourages sparsity in Θ and is convex, rendering (1) easy to optimize using the graphical lasso (GL) algorithm (Banerjee et al., 2008; Friedman et al., 2008). A similar approach that allows to penalize each element in Θ by a different value is the *weighted* graphical lasso (WGL), which imposes a penalty of the form $P(\Theta) = \|\Lambda \circ \Theta\|_1 = \sum_{ij} \Lambda_{ij} |\Theta_{ij}|$ (Li and Jackson, 2015; Zhuang et al., 2022; Zuo et al., 2017). Non-convex regularizers were also proposed in the literature (Williams, 2020). While more involved, they all boil down to encouraging different forms of sparsity.

Albeit less numerous, works incorporating prior distributions that do not involve sparsity also exist. In Zhou et al. (2021), a base graph structure is used as a prior, which could become too restrictive as it requires information from the specific graph whose support we want to estimate. In Wang and Li (2012), a G-Wishart prior distribution is used together with a Markov chain Monte Carlo (MCMC) sampler to determine the underlying GGM. A similar approach is taken in Friedman and Koller (2003), but using a standard Wishart instead. Another framework is given in Hosseini and Lee (2016), focusing on modularity. Even though these approaches do not assume mere sparsity, they propose pre-specified prior structures that may not apply to the graph under study.

All techniques proposed so far for GGM estimation impose limitations in the prior knowledge that can be incorporated. Importantly, the imposed priors are *too simple* (e.g., sparsity) or *restrictive* (e.g., G-Wishart), and these are *imposed on* Θ while the natural approach would be to impose the structural priors on the underlying adjacency matrix \mathbf{A} .

Addressing these limitations. This paper proposes a new approach that allows the introduction of *arbitrary* prior information *directly* on \mathbf{A} , based on a dataset of adjacency matrices \mathcal{A} (of potentially varying sizes) whose distribution we use as a prior $p(\mathbf{A})$. This is particularly useful for real-world applications where we often have datasets of graphs instead of a closed-form prior distribution $p(\mathbf{A})$. For instance, when learning brain networks, leveraging graphs from other patients is feasible (and valuable) because they often share similar structures. This idea also applies to other areas like molecular datasets or social net-

works. Additionally, many random graphs do not have a closed-form distribution, but generating samples to create a synthetic \mathcal{A} is relatively easy.

Furthermore, we allow edge-to-edge constraints (in a WGL fashion) so that edges that are *known* to either exist or be absent lead to values of 1 or 0 in \mathbf{A} , respectively. This is useful in cases where some pairs of variables are known to be conditionally independent, but we want to estimate the rest of the graph. Examples include gene expression data (Li and Jackson, 2015), neurotoxicology tests (Grzebyk et al., 2004), social networks (Wu et al., 2019), or brain networks (Simpson and Laurienti, 2015).

Our algorithm is based on Langevin dynamics, an MCMC sampler (Robert and Casella, 1999; Roberts and Tweedie, 1996). We directly sample from the posterior by defining a stochastic dynamic process whose stationary distribution matches the desired posterior distribution. If the interest is in a point estimate, we can readily use the samples to estimate, e.g., the posterior mean of the missing values in \mathbf{A} .

Contributions. Our three main contributions are:

- 1) We propose a novel GGM estimator based on *sampling* from a posterior distribution rather than finding the ML or MAP estimators and show that our estimator is consistent.
- 2) We leverage annealed Langevin dynamics to implement such an estimator, which allows us to incorporate an arbitrary prior distribution learned from data. We allow the known graphs to be of different sizes, as is the case in many practical applications.
- 3) Through numerical experiments, we show that incorporating arbitrary prior distributions outperforms estimators that only consider sparsity as prior information or are just based on the likelihood of the observations.

Notation. Scalars, vectors, and matrices are denoted by lowercase (y), lowercase bold (\mathbf{y}), and uppercase bold (\mathbf{Y}) letters, respectively. For a matrix \mathbf{Y} , Y_{ij} denotes its (i, j) -th entry. For a vector \mathbf{y} , its i -th component is represented by y_i . \mathbf{I} is the identity matrix of appropriate dimensions. The operation \circ denotes the Hadamard product. We define the element-wise indicator function as $\mathbb{I}\{\cdot\}$, and the support of a matrix as $\operatorname{supp}(\mathbf{Y}) = \mathbb{I}\{\mathbf{Y} \neq 0\}$ (i.e., a binary-valued matrix that is 0 in the (i, j) entries such that $Y_{ij} = 0$ and is 1 otherwise).

2 PROBLEM FORMULATION

We consider an unweighted and undirected graph \mathcal{G} with no self-loops that consists of n nodes and a partially known set of edges. The edge information is

encoded in the adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$. To distinguish between the entries of \mathbf{A} that are known and the ones we aim to estimate, we define two sets of indices \mathcal{O} and \mathcal{U} such that

$$\mathcal{O} = \{(i, j) : A_{ij} \text{ is observed} \wedge i < j\}, \text{ and} \quad (2)$$

$$\mathcal{U} = \{(i, j) : A_{ij} \text{ is unknown} \wedge i < j\}. \quad (3)$$

Throughout this work, we refer to the known and unknown fractions of the adjacency matrix as $\mathbf{A}^{\mathcal{O}}$ and $\mathbf{A}^{\mathcal{U}}$, respectively. The condition that $i < j$ implies that we do not take into account the diagonal (since the graph has no self-loops), and we just consider the upper-triangular part of \mathbf{A} (since it is symmetric). In case the set of edges is completely unknown, we can estimate some of its entries with high confidence and consider them known, as we show in Section 4.

Apart from the known fraction of the graph, we also assume that k independent observations are available. We arrange them as columns in the matrix $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_k] \in \mathbb{R}^{n \times k}$. Each observation \mathbf{x} follows a normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Theta}_0^{-1})$, where $\boldsymbol{\Theta}_0$ is the true precision matrix. Since $\mathbf{A}_0 = \text{supp}(\boldsymbol{\Theta}_0)$, then $\boldsymbol{\Theta}_0$ is known to be 0 where $\mathbf{A}_0^{\mathcal{O}}$ is 0 and is known to be different from 0 where $\mathbf{A}_0^{\mathcal{O}}$ is 1. We are also given a set of adjacency matrices \mathcal{A} drawn from the distribution $p(\mathbf{A})$, the same distribution from which \mathbf{A}_0 was drawn. In this setting, our problem is defined as follows:

Problem 1. *Given the k observations \mathbf{X} , a partially known adjacency matrix $\mathbf{A}_0^{\mathcal{O}}$, and structural prior information given by a set of matrices \mathcal{A} , find an estimate of $\mathbf{A}_0^{\mathcal{U}}$.*

A natural way to solve Problem 1 would be to compute the MAP, forcing the entries of the estimate to be equal to those of $\mathbf{A}_0^{\mathcal{O}}$ for all positions $(i, j) \in \mathcal{O}$. Mathematically, this is given by

$$\begin{aligned} \hat{\mathbf{A}}_{\text{MAP}} = \underset{\mathbf{A}}{\text{argmax}} \quad & p(\mathbf{X} | \mathbf{A}) p(\mathbf{A}) \\ \text{subject to } & A_{ij} = A_{0ij}^{\mathcal{O}} \quad \forall (i, j) \in \mathcal{O}. \end{aligned} \quad (4)$$

There are two main issues when solving (4), which we will describe in detail next. First, the likelihood $p(\mathbf{X} | \mathbf{A})$ is not easy to calculate, since only the expression for $p(\mathbf{X} | \boldsymbol{\Theta})$ is available, which is

$$p(\mathbf{X} | \boldsymbol{\Theta}) = \sqrt{\frac{\det \boldsymbol{\Theta}^k}{(2\pi)^{nk}}} \exp\left(-\frac{k}{2} \text{tr}(\mathbf{S}\boldsymbol{\Theta})\right), \quad (5)$$

where $\mathbf{S} = \frac{1}{k} \mathbf{X} \mathbf{X}^{\top}$ is the sample covariance. Hence, computing $p(\mathbf{X} | \mathbf{A})$ requires integrating (5) over all possible precision matrices such that $\text{supp}(\boldsymbol{\Theta}) = \mathbf{A}$, which is infeasible to do. Second, even if $p(\mathbf{X} | \mathbf{A})$ were available, carrying out the maximization in (4)

would be intractable since the feasible set contains $2^{|\mathcal{U}|}$ possible matrices.

Within the realm of point estimators, this work proposes an alternative approach to Problem 1, under which we estimate \mathbf{A}_0 as the posterior mean instead of the posterior mode. That is, we aim to compute

$$\mathbb{E}[\mathbf{A} | \mathbf{X}] = \sum_{\mathbf{A} \text{ s.t. } A_{ij} = A_{ij}^{\mathcal{O}}} \mathbf{A} \cdot p(\mathbf{A} | \mathbf{X}). \quad (6)$$

Note that the estimation of \mathbf{A}_0 can be considered a classification problem, where each edge is classified as 0 or 1. Hence, choosing a thresholded version of (6) as an estimator offers the desirable property of minimizing the edge classification error rate. However, even if we knew $p(\mathbf{A} | \mathbf{X})$, the summation in (6) requires computing $2^{|\mathcal{U}|}$ terms. Our approach to bypass this is to approximate (6) by taking the sample mean across M samples:

$$\mathbb{E}[\mathbf{A} | \mathbf{X}] \simeq \frac{1}{M} \sum_{m=1}^M \mathbf{A}^{(m)}. \quad (7)$$

The samples $\mathbf{A}^{(m)}$ should be drawn from the posterior

$$p(\mathbf{A} | \mathbf{X}) \propto p(\mathbf{X} | \mathbf{A}) p(\mathbf{A}), \quad (8)$$

where we omitted conditioning on $\mathbf{A}^{\mathcal{O}}$ to avoid cumbersome notation. As already explained, computing $p(\mathbf{X} | \mathbf{A})$ in (8) is, in general, infeasible. As a result, rather than trying to obtain $p(\mathbf{A} | \mathbf{X})$, our approach is to design an algorithm capable of sampling from (8) directly without explicitly computing the posterior. The design of such an algorithm, which has value per se and can be used to design other point estimators, is tackled in Section 3.

3 LANGEVIN FOR SUPPORT ESTIMATION

This section explains how to use annealed Langevin dynamics to solve Problem 1. In Section 3.1, we propose a distribution: i) that approximates the actual posterior (8) and ii) from which samples can be drawn by leveraging Langevin dynamics. Based on this approximate posterior, we define an estimator of \mathbf{A}_0 and show that it is consistent. Sections 3.2 and 3.3 explain how annealed Langevin dynamics works and why it provides a way to incorporate prior information in the estimation \mathbf{A}_0 via the so-called score function. Then, in Section 3.4, we study how to use the dataset \mathcal{A} as prior knowledge by training a graph neural network (GNN) whose output is directly plugged into the Langevin dynamics. Section 3.5 describes our final algorithm, which combines and summarizes the results of this section. An illustration of the overall procedure is shown in Figure 1.

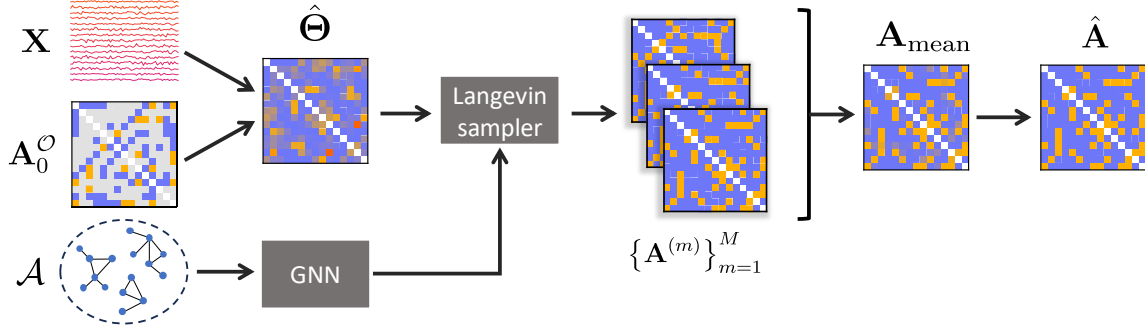


Figure 1: Illustration of our final algorithm [cf. Algorithm 1]. The grey entries in \mathbf{A}_0^O are what we aim to estimate (i.e., \mathbf{A}^U). If no entries of \mathbf{A}_0 are known, some can be estimated by bootstrapping \mathbf{X} , as shown in Section 4. By combining the GGM observations \mathbf{X} and the partially observed graph \mathbf{A}_0^O , we compute the constrained ML estimator $\hat{\Theta}$ by solving (9). This encodes information about the likelihood of \mathbf{A} given \mathbf{X} . To encode information about the prior $p(\mathbf{A})$, we process the dataset \mathcal{A} with a GNN (Section 3.4). Then, we can draw M samples from the (approximate) posterior using a Langevin sampler and build any estimator with them, such as $\hat{\mathbf{A}}$ in (11) that approximates the posterior mean.

3.1 Proposed estimator

The first step of our algorithm consists of computing the following estimator for Θ_0 ,

$$\begin{aligned} \hat{\Theta} = \underset{\Theta \succeq 0}{\operatorname{argmax}} \quad & \log \det \Theta - \operatorname{tr}(\mathbf{S}\Theta) \\ \text{s. to } \Theta_{ij} = 0 \quad & \forall (i, j) : A_{0ij}^O = 0, \end{aligned} \quad (9)$$

which corresponds to the positive definite matrix that maximizes the likelihood while respecting the zero pattern known to exist. The optimization problem in (9) can be efficiently solved by using the WGL algorithm mentioned in Section 1. The constraint is equivalent to setting a penalty Λ_{ij} to an arbitrarily large constant for those entries where \mathbf{A} is *known* to be 0, and setting $\Lambda_{ij} = 0$ otherwise.

Let $\mathcal{L}_{\mathbf{X}}(\Theta) = p(\mathbf{X} | \Theta)$ denote the likelihood of the precision matrix given the observed data. Then, based on the estimator in (9), we approximate the posterior $p(\mathbf{A} | \mathbf{X})$ in (8) as

$$\hat{p}(\mathbf{A} | \mathbf{X}) \propto \mathcal{L}_{\mathbf{X}}(\hat{\Theta} \circ (\mathbf{A} + \mathbf{I})) p(\mathbf{A}), \quad (10)$$

where we recall that \circ is the entry-wise product. Since the entries of $(\mathbf{A} + \mathbf{I})$ are binary, the entry-wise multiplication can be understood as a mask that sets to zero the entries of the precision that are not associated with an edge. Let us suppose now that we can sample from (10) and let $\{\mathbf{A}^{(m)}\}_{m=1}^M$ denote the set of M generated independent samples. Then, the set $\{\mathbf{A}^{(m)}\}_{m=1}^M$ can be used to characterize the posterior. We focus on the posterior sample mean estimator presented in (6)–(7). Then, the estimator for \mathbf{A}_0 that we

propose boils down to

$$\hat{\mathbf{A}} = \mathbb{I} \left\{ \left(\frac{1}{M} \sum_{m=1}^M \mathbf{A}^{(m)} \right) \geq \tau_k \right\}, \quad (11)$$

where τ_k is a tunable threshold that should increase with the sample size k .

We aim to prove that (11) is a consistent estimator of \mathbf{A}_0 , a fundamental result in our study. Before delving into such a proof, we establish two important intermediate results.

Lemma 1. *$\hat{\Theta}$ as defined in (9) is a consistent estimator (as $k \rightarrow \infty$) of the true precision matrix Θ_0 .*

Proof. See Section A.1 in the Supplementary Material (SM). \square

Lemma 2. *The approximate posterior $\hat{p}(\mathbf{A} | \mathbf{X})$ in (10) converges in distribution to*

$$\hat{p}(\mathbf{A} | \mathbf{X}) \xrightarrow{k \rightarrow \infty} \frac{p(\mathbf{A})}{C} \delta(\Theta_0 \circ (\mathbf{A} + \mathbf{I}) - \Theta_0), \quad (12)$$

where C is a constant and $\delta(\cdot)$ is the Dirac delta.

Proof. See Section A.2 in the SM. \square

We now leverage Lemmas 1 and 2 to show consistency of $\hat{\mathbf{A}}$.

Theorem 1. *$\hat{\mathbf{A}}$ as defined in (11) is a consistent estimator of the true adjacency matrix \mathbf{A}_0 when $M \rightarrow \infty$ and $\tau_k \xrightarrow{k \rightarrow \infty} 1$.*

Proof. According to Lemma 2, the only matrices \mathbf{A} with a positive probability of being sampled as $k \rightarrow \infty$ are those that satisfy

$$\Theta_0 = \Theta_0 \circ (\mathbf{A} + \mathbf{I}). \quad (13)$$

Let $\mathbf{A}^{(m)}$ be the m -th sample drawn from (12). The condition in (13) leads to

$$\mathbb{P}[A_{ij}^{(m)} = 0 \mid \Theta_{0_{ij}} \neq 0] = 0 \quad \forall m = 1, \dots, M. \quad (14)$$

Since the estimator $\hat{\mathbf{A}}$ from (11) is the mean of samples that follow (14), for $\tau_k > 0$ we have that

$$\mathbb{P}[\hat{A}_{ij} = 0 \mid \Theta_{0_{ij}} \neq 0] = 0. \quad (15)$$

On the other hand, false positives have a non-zero probability of being sampled:

$$\mathbb{P}[\hat{A}_{ij} = 1 \mid \Theta_{0_{ij}} = 0] = \mathbb{P}\left[\sum_{m=1}^M \frac{A_{ij}^{(m)}}{M} \geq \tau_k \mid \Theta_{0_{ij}} = 0\right]. \quad (16)$$

In the context of this proof, $\tau_k \rightarrow 1$. Additionally, the summation in (16) can be at most 1, since $A_{ij}^{(m)} \in \{0, 1\}$. Thus,

$$\mathbb{P}[\hat{A}_{ij} = 1 \mid \Theta_{0_{ij}} = 0] \xrightarrow{k \rightarrow \infty} \mathbb{P}\left[\sum_{m=1}^M \frac{A_{ij}^{(m)}}{M} = 1 \mid \Theta_{0_{ij}} = 0\right]. \quad (17)$$

Another way of writing (17) is

$$\mathbb{P}\left[\sum_{m=1}^M \frac{A_{ij}^{(m)}}{M} = 1 \mid \Theta_{0_{ij}} = 0\right] = \left(\mathbb{P}[A_{ij}^{(1)} = 1 \mid \Theta_{0_{ij}} = 0]\right)^M, \quad (18)$$

as each sample is drawn independently from the rest. Since $\mathbf{A}_0 \sim p(\mathbf{A})$, then from (12) it follows that

$$\mathbb{P}[\hat{A}_{ij}^{(1)} = 0 \mid \Theta_{0_{ij}} = 0] > 0. \quad (19)$$

Namely, given that the true adjacency matrix \mathbf{A}_0 has a prior distribution $p(\mathbf{A})$, it would not be possible for this matrix to have zero probability of being sampled from (12). Combining (19) with (18), and then taking the limit of (17) when $M \rightarrow \infty$ we get

$$\mathbb{P}[\hat{A}_{ij} = 1 \mid \Theta_{0_{ij}} = 0] \xrightarrow[M \rightarrow \infty]{k \rightarrow \infty} 0. \quad (20)$$

From (20) and (14) it follows that, if both $M \rightarrow \infty$ and $\tau_k \rightarrow 1$, then (11) converges in probability to the true adjacency matrix when $k \rightarrow \infty$. \square

In practical scenarios, infinite samples are never available, yet consistency is a desirable property for an estimator. Furthermore, even though Theorem 1 requires

$M \rightarrow \infty$, our experiments (Section 4) reveal that our method outperforms classical methods for relatively small values of M .

To compute $\hat{\mathbf{A}}$ as in (11) we need to be able to sample from the posterior distribution in (10). To this end, we utilize the stochastic diffusion process of Langevin dynamics.

3.2 Langevin dynamics

The Langevin dynamics algorithm is an MCMC method that allows us to draw samples from a distribution difficult to sample from directly (Robert and Casella, 1999; Roberts and Tweedie, 1996). This sampler's great advantage is that it does not require an expression for the target distribution but rather for the gradient of its logarithm. Generically, to sample from $p(\mathbf{w})$ via Langevin, only $\nabla_{\mathbf{w}} \log p(\mathbf{w})$ is needed. This gradient receives the name of *score function* and is of paramount relevance in the ensuing sections.

For a generic target distribution $p(\mathbf{w})$, the Langevin dynamics are given by

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \epsilon \nabla_{\mathbf{w}} \log p(\mathbf{w}_t) + \sqrt{2\epsilon} \mathbf{z}_t, \quad (21)$$

where t is an iteration index, ϵ is the step size and $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$. In each iteration, \mathbf{w}_t tends to move in the direction of the score function but is also affected by white noise that prevents it from collapsing in local maxima. Under some regularity conditions, \mathbf{w}_t converges to be a sample from $p(\mathbf{w})$ when $\epsilon \rightarrow 0$ and $t \rightarrow \infty$ (Welling and Teh, 2011).

It should be noted that, in our case, we are trying to sample a *discrete* random vector (i.e., a vectorized unweighted adjacency matrix). Hence, the gradient of the target log-density is not defined in our setting. A noisy (continuous) version of the random vector is used to circumvent this obstacle. This idea leads to the *annealed* Langevin dynamics (Kawar et al., 2021; Song and Ermon, 2019).

3.3 Annealed Langevin dynamics

To simplify the notation of what follows, we use \mathbf{A} and its half-vectorization $\mathbf{a} = \text{vech}(\mathbf{A})$ interchangeably. Consider a noisy version of \mathbf{a} ,

$$\tilde{\mathbf{a}} = \mathbf{a} + \mathbf{v}, \quad (22)$$

where \mathbf{v} represents additive Gaussian noise. More precisely, let $\{\sigma_l\}_{l=1}^L$ be a sequence of *noise levels* such that $\sigma_1 > \sigma_2 > \dots > \sigma_L > 0$. Then, for each noise level we define $\mathbf{v}_l \sim \mathcal{N}(\mathbf{0}, \sigma_l^2 \mathbf{I})$. In this setting, $\tilde{\mathbf{a}}$ is continuous, and the iterative procedure involving annealed Langevin dynamics for our problem is given by

$$\tilde{\mathbf{a}}_{t+1} = \tilde{\mathbf{a}}_t + \alpha_t \nabla_{\tilde{\mathbf{a}}} \log p(\tilde{\mathbf{a}}_t \mid \mathbf{X}) + \sqrt{2\alpha_t} \mathbf{z}_t, \quad (23)$$

where $\alpha_t = \epsilon \cdot \sigma_{l(t)}^2 / \sigma_L^2$ and $l(t)$ is an increasing function mapping time steps t to the annealing noise levels l . Note that the noise present in $\tilde{\mathbf{a}}_t$ (i.e., the variance of $\mathbf{v}_{l(t)}$) decreases with t , as given by the varying step size α_t .

The annealed version of the dynamics was initially introduced to allow the algorithm to converge faster and perform better (Song and Ermon, 2019). However, in our case, it also offers the advantage of rendering the problem differentiable. Consequently, the annealing enables the computation of the score functions and the use of Langevin dynamics to sample from an originally discrete distribution. If the noise levels $\{\sigma_l\}_{l=1}^L$ and the step size ϵ are chosen adequately (Song and Ermon, 2019), after a sufficiently large number of iterations, the sample $\tilde{\mathbf{a}}_t$ is arbitrarily close to an actual sample from the discrete distribution $p(\mathbf{a} \mid \mathbf{X})$. If an actual sample is needed, the noisy sample $\tilde{\mathbf{a}}_t$ must be projected onto the set $\{0, 1\}$.

Now we need to compute the annealed score $\nabla_{\tilde{\mathbf{a}}} \log p(\tilde{\mathbf{a}} \mid \mathbf{X})$ to sample graphs using (23). To avoid the use of cumbersome notation in what follows, from now on, we drop the reference to $\tilde{\mathbf{a}}$ in the gradients, as we always take the derivatives with respect to that vector. Using (10), we express the (approximate) annealed posterior score as

$$\nabla \log \hat{p}(\tilde{\mathbf{A}} \mid \mathbf{X}) = \nabla \log \mathcal{L}_{\mathbf{X}}(\tilde{\Theta}) + \nabla \log p(\tilde{\mathbf{A}}), \quad (24)$$

where we have defined

$$\tilde{\Theta} = \hat{\Theta} \circ (\tilde{\mathbf{A}} + \mathbf{I}). \quad (25)$$

We next discuss each of the two terms in (24). Starting with $\nabla \log \mathcal{L}_{\mathbf{X}}(\tilde{\Theta})$, referred to as the *annealed likelihood score*, we compute it as [cf. (5)]

$$\nabla \log \mathcal{L}_{\mathbf{X}}(\tilde{\Theta}) = \frac{k}{2} \nabla \log \det(\tilde{\Theta}) - \frac{k}{2} \nabla \text{tr}(\mathbf{S}\tilde{\Theta}), \quad (26)$$

with the two gradients in (26) being straightforward to compute (Petersen and Pedersen, 2012). Specifically, let us define $\Delta\tilde{\Sigma} = \tilde{\Theta}^{-1} - \mathbf{S}$ and use \mathbf{T}^{ij} to denote a matrix whose entries are all equal to zero except the (i, j) -th and the (j, i) -th ones, which are one. Then,

$$\frac{\partial \log \mathcal{L}_{\mathbf{X}}(\tilde{\Theta})}{\partial \tilde{A}_{ij}} = \frac{k}{2} \text{tr} \left[\left(2\Delta\tilde{\Sigma} + \Delta\tilde{\Sigma} \circ \mathbf{I} \right) \left(\hat{\Theta} \circ \mathbf{T}^{ij} \right) \right]. \quad (27)$$

We shift now to $\nabla \log p(\tilde{\mathbf{A}})$, the second term in (24), which is referred to as the *annealed prior score* and is more difficult to obtain. Note that computing $p(\tilde{\mathbf{A}})$ requires convolving $p(\mathbf{A})$ with the distribution of the noise [cf. (22)], which is infeasible not only because of the computational burden of that task but also because we do not know $p(\mathbf{A})$. The alternative that

we propose is to *estimate* the annealed prior score $\nabla \log p(\tilde{\mathbf{A}})$ just using samples from the prior $p(\mathbf{A})$ (i.e., the available dataset \mathcal{A}), as in Sevilla and Segarra (2023). We model this estimate as a GNN, where weights are trained on the dataset \mathcal{A} , as we explain in Section 3.4.

3.4 Learned annealed scores

Let $\mathbf{g}_{\xi}(\tilde{\mathbf{a}}, \sigma)$ be the output of the GNN we wish to train, with ξ being its trainable parameters. Ideally, the output for a given $\tilde{\mathbf{a}}$ (with the associated noise level σ_l of the current iteration) should be as close as possible to the actual score $\nabla \log p(\tilde{\mathbf{a}})$. The loss function to learn ξ should be designed to jointly minimize the mean squared error across all noise levels. To achieve this, we define the distance

$$\begin{aligned} \mathcal{D}(\tilde{\mathbf{a}} \mid \xi, \sigma_l) &= \|\mathbf{g}_{\xi}(\tilde{\mathbf{a}}, \sigma_l) - \nabla \log p(\tilde{\mathbf{a}} \mid \mathbf{a})\|_2^2 \\ &= \|\mathbf{g}_{\xi}(\tilde{\mathbf{a}}, \sigma_l) - (\mathbf{a} - \tilde{\mathbf{a}})/\sigma_l^2\|_2^2 \end{aligned} \quad (28)$$

and the associated loss function

$$\mathcal{J}(\xi \mid \{\sigma_l\}_{l=1}^L) = \frac{1}{2L} \sum_{l=1}^L \sigma_l^2 \mathbb{E}[\mathcal{D}(\tilde{\mathbf{a}} \mid \xi, \sigma_l)]. \quad (29)$$

Following the proof in Vincent (2011), it follows that the output of a GNN trained with (29) correctly estimates $\nabla \log p(\tilde{\mathbf{a}})$. It is worth pointing out that the term $(\mathbf{a} - \tilde{\mathbf{a}})/\sigma_l^2$ is known during training: \mathbf{a} is one element of \mathcal{A} and both $\tilde{\mathbf{a}}$ and σ_l are the GNN inputs.

The architecture of the GNN must account for the fact that the same graph can be represented by different adjacency matrices, depending on the node labeling. In this work, we leverage the EDP-GNN (Niu et al., 2020), designed to perform score-matching on graphs by proposing a permutation equivariant method to model the score function of interest.

3.5 Final algorithm

Now we need to put all the pieces together: the proposed (consistent) estimator $\hat{\mathbf{A}}$ (Section 3.1), the Langevin dynamics to get the samples to compute that estimator (Sections 3.2 and 3.3), and the GNN training to estimate the score needed to run the Langevin dynamics (Section 3.4). The final scheme is described in Algorithm 1. Notice that the score estimator $\mathbf{g}_{\xi}(\cdot)$ is an input. Namely, before performing any GGM estimation, a GNN has to be trained with the desired dataset \mathcal{A} in order to be able to compute $\mathbf{g}_{\xi}(\tilde{\mathbf{a}}, \sigma) \simeq \nabla \log p(\tilde{\mathbf{a}})$ for the different noise levels σ_l .

The first step in Algorithm 1 is to compute $\hat{\Theta}$ as in (9). We then draw samples from the approximate posterior distribution $\hat{p}(\mathbf{A} \mid \mathbf{X})$ by running the dynamics in (23).

Algorithm 1 Annealed Langevin for GGM estimation

Require: $\mathbf{X}, \mathbf{A}_0^{\mathcal{O}}, \mathbf{g}_{\xi}(\cdot), \{\sigma_l\}_{l=1}^L, M, T, \epsilon, \tau_k$

- 1: $\mathbf{S} \leftarrow \frac{1}{k} \mathbf{X} \mathbf{X}^{\top}$
- 2: Compute $\hat{\Theta}$ as in (9)
- 3: $\mathcal{S} \leftarrow \{\}$ ▷ Set of generated samples
- 4: **repeat**
- 5: Initialize $\tilde{\mathbf{A}}_0^{\mathcal{U}} \sim \mathcal{N}(0.5, 0.5\mathbf{I})$
- 6: $\tilde{\mathbf{A}}_0^{\mathcal{O}} \leftarrow \mathbf{A}_0^{\mathcal{O}}$ ▷ Fix the known values
- 7: **for** $l \leftarrow 1$ to L **do**
- 8: $\alpha_l \leftarrow \epsilon \cdot \sigma_l^2 / \sigma_L^2$ ▷ Change the noise level
- 9: **for** $t \leftarrow 1$ to T **do**
- 10: Draw $\mathbf{Z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 11: Compute $\nabla \log \mathcal{L}_{\mathbf{X}}(\tilde{\Theta}_t)$ as in (27)
- 12: Compute $\mathbf{g}_{\xi}(\tilde{\mathbf{A}}_{t-1}, \sigma_l)$
- 13: $\Delta_t \leftarrow \nabla \log \mathcal{L}_{\mathbf{X}}(\tilde{\Theta}_t) + \mathbf{g}_{\xi}(\tilde{\mathbf{A}}_{t-1}, \sigma_l)$
- 14: $\tilde{\mathbf{A}}_t^{\mathcal{U}} \leftarrow \tilde{\mathbf{A}}_{t-1}^{\mathcal{U}} + \alpha_l \Delta_t^{\mathcal{U}} + \sqrt{2\alpha_l} \mathbf{Z}_t$
- 15: $\tilde{\mathbf{A}}_t^{\mathcal{O}} \leftarrow \tilde{\mathbf{A}}_{t-1}^{\mathcal{O}}$
- 16: **end for**
- 17: $\tilde{\mathbf{A}}_0 \leftarrow \tilde{\mathbf{A}}_T$
- 18: **end for**
- 19: $\tilde{\mathbf{A}} \leftarrow \tilde{\mathbf{A}}_T$ ▷ A sample from $p(\tilde{\mathbf{A}} | \mathbf{X})$
- 20: $\mathbf{A} \leftarrow \mathbb{I}\{\tilde{\mathbf{A}} \geq 0.5\}$ ▷ Project onto $\{0, 1\}$
- 21: $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{A}\}$
- 22: **until** \mathcal{S} contains M samples
- 23: Store the sample mean of \mathcal{S} in \mathbf{A}_{mean}
- 24: $\hat{\mathbf{A}} \leftarrow \mathbb{I}\{\mathbf{A}_{\text{mean}} \geq \tau_k\}$
- 25: **return** $\hat{\mathbf{A}}$

Recall that this is possible because a) we count on a closed-form (approximate) expression for the annealed likelihood (27), and b) we have found a way to estimate the annealed prior score by training a GNN with \mathcal{A} . Notice that, in each step, we just update the values of $\tilde{\mathbf{A}}^{\mathcal{U}}$, leaving the known values in $\tilde{\mathbf{A}}^{\mathcal{O}}$ fixed.

After LT steps for each sample, the algorithm generates a continuous matrix $\tilde{\mathbf{A}}$. As we work with unweighted graphs, it is necessary to make the prediction binary-valued. Therefore, the algorithm draws $\mathbb{I}\{\tilde{\mathbf{A}} \geq 0.5\}$ as a sample instead, representing an element-wise projection onto the set $\{0, 1\}$. Following this procedure, we draw M samples and then compute their average. Lastly, we apply a threshold τ_k to the approximate posterior mean to compute the consistent estimator $\hat{\mathbf{A}}$.

4 NUMERICAL RESULTS

We carry out simulations in different setups ¹ to demonstrate our scheme’s practical relevance and gain

¹Source code is available at https://github.com/Tenceto/langevin_ggm.

insight regarding how informative the prior knowledge is when estimating \mathbf{A}_0 .

In all the simulations, we first generate a fully-known graph and then drop $|\mathcal{U}|$ random entries of \mathbf{a}_0 , which we then try to estimate. We generate $M = 10$ samples for each graph to compute (11). We compare our method with: ²

- **WGL** (Li and Jackson, 2015). We penalize the indices in \mathcal{U} with a parameter λ , use an arbitrarily large penalty where $\mathbf{A}_0^{\mathcal{O}}$ is 0, and do not penalize the entries where $\mathbf{A}_0^{\mathcal{O}}$ is 1.
- **Thresholding**. We compute $\hat{\Theta}$ and threshold it.
- **TIGER**. The GGM estimation method in Liu and Wang (2017) which does not require tuning.
- **GraphSAGE**. A link prediction method (not a GGM estimation method like the others) based on GNNs (Hamilton et al., 2017). We use the measurements \mathbf{X} as node features and $\mathbf{A}_0^{\mathcal{O}}$ as the training set while testing $\mathbf{A}_0^{\mathcal{U}}$.

All the thresholds (τ_k for our algorithm and those used for the thresholding and GraphSAGE methods) and λ for WGL are tuned using a training set. It is worth pointing out that the information given by $\mathbf{A}_0^{\mathcal{O}}$ cannot be used within the TIGER algorithm, as it requires fixing some entries of Θ .

Additionally, we use as a benchmark a variant of Algorithm 1. We label it as “Langevin prior” (LPr), since it consists of just using prior information (i.e., $\Delta_t = \mathbf{g}_{\xi}(\tilde{\mathbf{A}}_{t-1}, \sigma_l)$ in line 13). In other words, we test the algorithm when no observations are available, but only \mathcal{A} is. Our method is labeled as “Langevin posterior” (LPost), using both the prior and likelihood score functions.

We run simulations for three different kinds of graphs. Two of them, grid graphs and Barabási-Albert graphs (Barabási and Albert, 1999), are synthetic, while the third one consists of ego-nets of Eastern European users collected from the music streaming service Deezer (Rozemberczki et al., 2020). Next, we report and discuss the numerical results for all simulated scenarios. We report the average F1 score over 10 different train/test splits over 100 graphs in each case.

Partially unknown grids. We consider grids of different heights and widths with few additional random edges. Results are shown in Figure 2.

As k increases, the performance of the predictors that use \mathbf{X} increases, except for GraphSAGE. Recall that the presence of an edge between two nodes does not

²Additional details on hyperparameter choices, properties of datasets, and computation of the reported metrics can be found in Section B of the SM.

imply a direct correlation between the variables, but rather *conditional dependence* given the rest of the graph. Considering that this relationship is not captured by local neighborhoods, which is how GraphSAGE aggregates node data, this method is expected to not benefit from including more observations.

For both sizes of $|\mathcal{U}|$ and for the four different ratios $k/|\mathcal{U}|$, LPost outperforms all the other approaches. However, it is worth pointing out that the gap is much more prominent in Figure 2a when $|\mathcal{U}|$ is smaller. LPr’s predictions also present a higher F1 score in that case. This behavior leads to thinking that the information provided by \mathcal{A} decreases as $|\mathcal{U}|$ (and, thus, the dimensions of the space from which the Langevin process is sampling) increases. A complementary experiment on the performance dependence on $|\mathcal{A}|$ for a different type of graph is presented in Section C.1 of the SM.

When $|\mathcal{U}|$ is small, the prior probability mass is concentrated among fewer possible graphs. Intuitively, in this case, Langevin generally samples either the same graph or similar ones throughout the different M samples. Thus, the sample mean yields a satisfactory estimate. When $|\mathcal{U}|$ is large, the probability mass is spread across many adjacency matrices in the high-dimensional space of $p(\mathbf{A})$. This leads to Langevin converging to diverse graphs each time we sample, reducing the usefulness of the sample mean as an estimator. An additional experiment illustrating the performance dependence on $|\mathcal{U}|$ is presented in Section C.2 of the SM.

Partially unknown Barabási–Albert graphs.

Now we consider the dual Barabási–Albert preferential attachment model (Moshiri, 2018). All graphs in \mathcal{A} are such that $n \in \{47, 49, 51, 53\}$, while we used graphs with $n \in \{46, 48, 50, 52\}$ nodes to test the algorithm. This allows us to verify whether the EDP-GNN correctly generalizes the score estimation. The results are shown in Figure 3a.

Once again, LPost yields better results than the other algorithms, mainly when k is small. As more observations are available, all of the methods (except for LPr and GraphSAGE) have approximately the same performance – the information provided by \mathcal{A} becomes negligible compared to that offered by \mathbf{X} .

The F1 score achieved by LPr is relatively poor due to the large randomness in the graphs. Namely, it is always worse than the one obtained using WGL. On the contrary, when the underlying graph presents more structure (for instance, the grid graphs in Figure 2a), LPr was shown to outperform WGL for some values of $k/|\mathcal{U}|$. We can conclude that some priors offer more predictive power than others: the more substantial the structure of the graphs, the more useful $p(\mathbf{A})$ becomes.

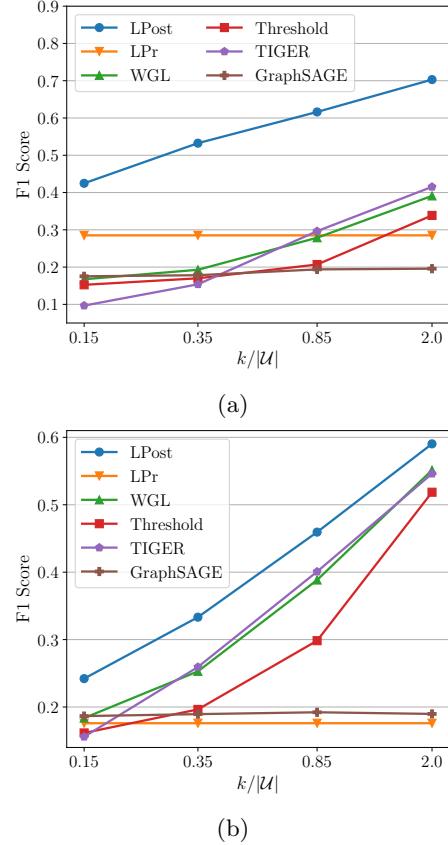


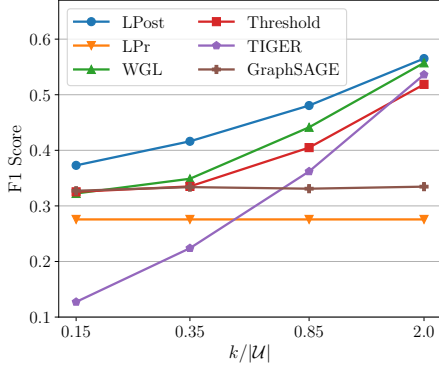
Figure 2: F1 score of several methods using grid graphs with $40 \leq n \leq 50$ where (a) 10% and (b) 20% of the values in \mathbf{a} are unknown.

Partially unknown ego-nets. Now, we consider the graphs in the Deezer dataset with $n \leq 25$. The results are shown in Figure 3b.

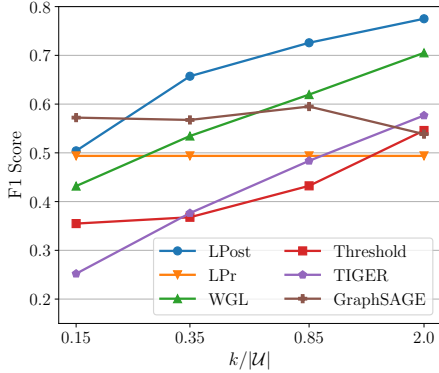
Once again, our method exhibits a higher edge prediction performance than the rest. The behavior is similar to the one observed in the previous setups: the accuracy of all GGM-based methods increases with k . GraphSAGE slightly outperforms LPost in this scenario for the smallest values of k . Ego-nets are strongly local-based, and GraphSAGE is expected to outperform the rest of the approaches when the information provided by the observations is negligible.

Ego-nets, like grids, present a strong structure, rendering the prior highly predictive. Even though half of the graph is unknown, the F1 score of LPr is the highest among all the experiments (cf. Figures 2 and 3a).

Fully unknown ego-nets. We consider the same dataset as in the last experiment, but now all the entries in \mathbf{A} (except for those in the diagonal, which are 0) are assumed unknown so that $|\mathcal{O}| = 0$. As shown in Figure 2, the prior offers less predictive power as



(a)



(b)

Figure 3: F1 score of several methods using (a) Barabási-Albert graphs with $|\mathcal{U}| = 0.1 \dim(\mathbf{a})$, and (b) ego-nets with $|\mathcal{U}| = 0.5 \dim(\mathbf{a})$.

$|\mathcal{U}|$ increases (see Section C.2 for another experiment investigating this behavior). Thus, when the entire graph is to be estimated, we propose fixing some of the entries in \mathbf{A} with a graphical version of the *random lasso* (Wang et al., 2011).

To that end, we first compute the GL solution B times (we use $B = 50$ in these experiments), where a different bootstrap sample $\mathbf{X}^{(b)}$ is used for each iteration to obtain $\hat{\Theta}_{\text{boot}}^{(b)}$. Then, the average

$$\hat{\mathbf{A}}_{\text{boot}} = \frac{1}{B} \sum_{b=1}^B \mathbb{I} \left\{ \hat{\Theta}_{\text{boot}}^{(b)} \neq 0 \right\} \quad (30)$$

can be interpreted as the probability of each A_{ij} to be 1. Thus, for some probability margin p_m , we assume known some entries A_{ij} such that

$$A_{ij}^{\mathcal{O}} = \begin{cases} 0 & \text{if } \hat{A}_{\text{boot}_{ij}} < 0.5 - p_m \\ 1 & \text{if } \hat{A}_{\text{boot}_{ij}} > 0.5 + p_m \end{cases}, \quad (31)$$

leaving as unknown all entries (i, j) such that $0.5 - p_m \leq \hat{A}_{\text{boot}_{ij}} \leq 0.5 + p_m$. As p_m increases, the confidence of the estimated fixed values is higher, and $|\mathcal{O}|$ becomes smaller.

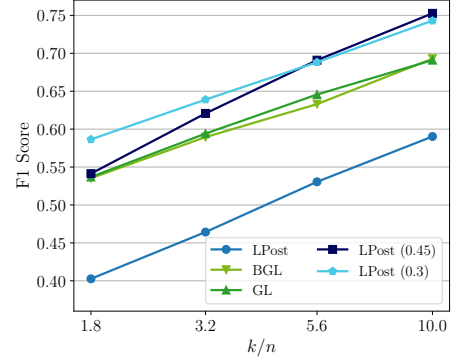


Figure 4: F1 score comparison when estimating ego-nets with no known values in \mathbf{A} . The values in parentheses correspond to the p_m used to fix values from $\hat{\mathbf{A}}_{\text{boot}}$ prior to using our method LPost.

The results for two different values of p_m are shown in Figure 4, where we also compare with GL and its bootstrapped counterpart (BGL). The latter is computed as in (30) and then thresholded. We observe that a naive implementation of our method falls behind when $|\mathcal{O}| = 0$, an expected behavior as analyzed in the experiments with grid graphs. However, by leveraging the bootstrapping procedure to fix some entries in \mathbf{A} we outperform GL and BGL, indicating that the prior distribution significantly contributes to the prediction accuracy.

Overall, our numerical experiments show that i) our approach leads to better graph estimation results than the classical alternatives considered and ii) the benefits of our approach are more significant when the number of observations is small and the graph presents marked structural features.

5 CONCLUSIONS

We proposed a GGM estimation algorithm based on annealed Langevin dynamics that allows us to leverage graph structural priors beyond sparsity. Our approach exploits a set of known graphs to extract the prior distribution. We designed an algorithm that, by combining annealed Langevin dynamics with a GNN-based annealed prior score estimator, was able to draw samples from the posterior distribution of interest, namely the distribution of the unknown edges given the known ones, the structural prior, and the GMRF observations. Finally, we proposed a consistent point estimate for the graph that underlies the GGM based on the sample posterior mean. Through numerical experiments, we showed our method outperforms classical ones, especially in cases with few observations and highly structured graphs.

Acknowledgments

This research was sponsored by the Army Research Office under Grant Number W911NF-17-S-0002; the Spanish (MCIN/AEI/10.13039/501100011033) Grants PID2019-105032GB-I00 and PID2022-136887NB-I00; the Autonomous Community of Madrid within the EL-LIS Unit Madrid framework; and the Fulbright U.S. Student Program, in turn sponsored by the U.S. Department of State and the U.S.–Argentina Fulbright Commission. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office, the U.S. Army, the Fulbright Program, the U.S.–Argentina Fulbright Commission, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

References

- Banerjee, O., El Ghaoui, L., and d’Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516.
- Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.
- Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Casella, G. and Berger, R. L. (2021). *Statistical Inference*. Cengage Learning.
- Codazzi, L., Colombi, A., Gianella, M., Argiento, R., Paci, L., and Pini, A. (2022). Gaussian graphical modeling for spectrometric data analysis. *Computational Statistics & Data Analysis*, 174:107416.
- Dempster, A. P. (1972). Covariance selection. *Biometrics*, pages 157–175.
- Dobra, A., Eicher, T. S., and Lenkoski, A. (2010). Modeling uncertainty in macroeconomic growth determinants using Gaussian graphical models. *Statistical Methodology*, 7(3):292–306.
- Dobra, A., Hans, C., Jones, B., Nevins, J. R., Yao, G., and West, M. (2004). Sparse graphical models for exploring gene expression data. *Journal of Multivariate Analysis*, 90(1):196–212.
- Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.
- Friedman, N. and Koller, D. (2003). Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Machine learning*, 50:95–125.
- Grzebyk, M., Wild, P., and Chouanière, D. (2004). On identification of multi-factor models with correlated residuals. *Biometrika*, 91(1):141–151.
- Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing systems*, 30.
- Hosseini, M. J. and Lee, S.-I. (2016). Learning sparse Gaussian graphical models with overlapping blocks. *Advances in neural information processing systems*, 29.
- Hunter, D. R. and Handcock, M. S. (2006). Inference in curved exponential family models for networks. *Journal of Computational and Graphical Statistics*, 15(3):565–583.
- Kawar, B., Vaksman, G., and Elad, M. (2021). SNIPS: Solving noisy inverse problems stochastically. *Advances in Neural Information Processing Systems*, 34:21757–21769.
- Li, T., Qian, C., Levina, E., and Zhu, J. (2020). High-dimensional Gaussian graphical models on network-linked data. *The Journal of Machine Learning Research*, 21(1):2851–2895.
- Li, Y. and Jackson, S. (2015). Gene network reconstruction by integration of biological prior knowledge. *G3-Genes Genomes Genetics*, 5:1075–1079.
- Liu, H. and Wang, L. (2017). TIGER: A tuning-insensitive approach for optimally estimating Gaussian graphical models. *Electronic Journal of Statistics*, 11(1):241 – 294.
- Moshiri, N. (2018). The dual-Barabási-Albert model.
- Niu, C., Song, Y., Song, J., Zhao, S., Grover, A., and Ermon, S. (2020). Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR.
- Petersen, K. B. and Pedersen, M. S. (2012). The matrix cookbook. Version 20121115.
- Qiu, Y. and Liyanage, J. S. (2019). Threshold selection for covariance estimation. *Biometrics*, 75(3):895–905.
- Ravikumar, P., Wainwright, M. J., Raskutti, G., and Yu, B. (2011). High-dimensional covariance estimation by minimizing l1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5(none):935 – 980.
- Robert, C. and Casella, G. (1999). *Monte Carlo Statistical Method*. Springer.

- Roberts, G. O. and Tweedie, R. L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2:341–363.
- Rozemberczki, B., Kiss, O., and Sarkar, R. (2020). Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *ACM International Conference on Information and Knowledge Management*, page 3125–3132. ACM.
- Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields: Theory and Applications*. CRC press.
- Sevilla, M. and Segarra, S. (2023). Bayesian topology inference on partially known networks from input-output pairs.
- Simpson, S. L. and Laurienti, P. J. (2015). A two-part mixed-effects modeling framework for analyzing whole-brain network data. *NeuroImage*, 113:310–319.
- Snijders, T. A. B., Pattison, P. E., Robins, G. L., and Handcock, M. S. (2006). New specifications for exponential random graph models. *Sociological Methodology*, 36(1):99–153.
- Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32.
- Sundaram, R. K. (1996). *A first course in optimization theory*. Cambridge University Press.
- Tan, L. S. L., Jasra, A., Iorio, M. D., and Ebbels, T. M. D. (2017). Bayesian inference for multiple Gaussian graphical models with application to metabolic association networks. *The Annals of Applied Statistics*, 11(4):2222–2251.
- Tsai, K., Koyejo, O., and Kolar, M. (2022). Joint Gaussian graphical model estimation: A survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, 14(6):e1582.
- Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674.
- Wang, H. and Li, S. Z. (2012). Efficient Gaussian graphical model determination under G-Wishart distributions. *Electronic Journal of Statistics*, 6:168–198.
- Wang, S., Nan, B., Rosset, S., and Zhu, J. (2011). Random lasso. *The annals of applied statistics*, 5(1):468.
- Wang, Y., Segarra, S., and Uhler, C. (2020). High-dimensional joint estimation of multiple directed Gaussian graphical models. *Electronic Journal of Statistics*, 14(1):2439 – 2483.
- Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Intl. Conf. on Machine Learning*, page 681–688.
- Williams, D. R. (2020). Beyond lasso: A survey of non-convex regularization in Gaussian graphical models.
- Wu, Q., Zhang, Z., Waltz, J., Ma, T., Milton, D., and Chen, S. (2019). Predicting latent links from incomplete network data using exponential random graph model with outcome misclassification. *bioRxiv*.
- Zhou, J., Hoen, A., Mcritchie, S., Pathmasiri, W., Viles, W., Nguyen, Q., Madan, J., Dade, E., Karagas, M., and Gui, J. (2021). Information enhanced model selection for Gaussian graphical model with application to metabolomic data. *Biostatistics*, 23.
- Zhuang, Y., Xing, F., Ghosh, D., Banaei-Kashani, F., Bowler, R. P., and Kechris, K. (2022). An augmented high-dimensional graphical lasso method to incorporate prior biological knowledge for global network learning. *Frontiers in Genetics*, page 2405.
- Zuo, Y., Cui, Y., Yu, G., Li, R., and Ressom, H. (2017). Incorporating prior biological knowledge for network-based differential gene expression analysis using differentially weighted graphical lasso. *BMC Bioinformatics*, 18.

Checklist

1. For all models and algorithms presented, check if you include:
 - (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes**.
 - (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes**. Additional analysis is provided in the SM. In particular, for a complexity analysis please refer to Section **B.3**.
 - (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **Yes**.
2. For any theoretical claim, check if you include:
 - (a) Statements of the full set of assumptions of all theoretical results. **Yes**.
 - (b) Complete proofs of all theoretical results. **Yes**. Proofs to the auxiliary lemmas are provided in the SM (Section **A**).
 - (c) Clear explanations of any assumptions. **Yes**.
3. For all figures and tables that present empirical results, check if you include:

- (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Yes.** Please refer to the source code if needed.
 - (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes.** Please refer to Sections **B.1** and **B** of the SM for more information.
 - (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes.** Additional information can be found in Section **B** of the SM.
 - (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes.** See Section **B.3** in the SM.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
- (a) Citations of the creator if your work uses existing assets. **Yes.**
 - (b) The license information of the assets, if applicable. **Yes.** Publication of our source code is available in GitHub under an MIT License.
 - (c) New assets either in the supplemental material or as a URL, if applicable. **Yes.**
 - (d) Information about consent from data providers/curators. **Not Applicable.**
 - (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not Applicable.**
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
- (a) The full text of instructions given to participants and screenshots. **Not Applicable**
 - (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable**
 - (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable**

Estimation of partially known Gaussian graphical models with score-based structural priors

Supplementary Materials

A PROOFS OF LEMMAS

A.1 Proof of Lemma 1

Let $\mathcal{S}_+^n = \{\mathbf{V} \in \mathbb{R}^{n \times n} \mid \mathbf{V} \succeq 0\}$ be the set of all positive semidefinite matrices. Then, we define a function $h : \mathcal{S}_+^n \rightarrow \mathcal{S}_+^n$ such that

$$\begin{aligned} h(\mathbf{V}) &= \operatorname{argmax}_{\boldsymbol{\Theta} \succeq 0} f(\boldsymbol{\Theta}; \mathbf{V}) \\ \text{s. to } \boldsymbol{\Theta}_{ij} &= 0 \quad \forall (i, j) : A_{ij}^{\mathcal{O}} = 0, \end{aligned} \quad (32)$$

where $f : \mathcal{S}_+^n \rightarrow \mathbb{R}$ is $f(\boldsymbol{\Theta}; \mathbf{V}) = \log \det \boldsymbol{\Theta} - \operatorname{tr}(\mathbf{V}\boldsymbol{\Theta})$. It immediately follows that the estimator in (9) of the main paper satisfies

$$\hat{\boldsymbol{\Theta}} = h(\mathbf{S}), \quad (33)$$

with $\mathbf{S} = \frac{1}{k} \mathbf{X}\mathbf{X}^\top$ being the sample covariance matrix. The estimator $\hat{\boldsymbol{\Theta}}$ is consistent if $h(\mathbf{S})$ approaches $\boldsymbol{\Theta}_0$ as k increases. Hence, this is what we want to prove next.

First, we compute $h(\boldsymbol{\Theta}_0^{-1})$. To this end, we first check what matrix maximizes f without considering the constraint in (32). The maximizer is unique since f is continuous and strictly concave in \mathcal{S}_+^n (Ravikumar et al., 2011). Taking the gradient of f with respect to $\boldsymbol{\Theta}$ yields

$$\frac{\partial f(\boldsymbol{\Theta}; \mathbf{V})}{\partial \boldsymbol{\Theta}} = 2\boldsymbol{\Theta}^{-1} - \boldsymbol{\Theta}^{-1} \circ \mathbf{I} - 2\mathbf{V} - \mathbf{V} \circ \mathbf{I} = \mathbf{0} \iff \boldsymbol{\Theta} = \mathbf{V}^{-1}. \quad (34)$$

Hence, $f(\boldsymbol{\Theta}; \boldsymbol{\Theta}_0^{-1})$ is maximized when $\boldsymbol{\Theta} = \boldsymbol{\Theta}_0$. Furthermore, $\boldsymbol{\Theta}_0$ satisfies the constraint in (32). As a result, it holds that

$$h(\boldsymbol{\Theta}_0^{-1}) = \boldsymbol{\Theta}_0. \quad (35)$$

Notice that the mapping h is continuous. This can be proven through the maximum theorem (Sundaram, 1996). Let $\mathcal{C} = \{\mathbf{V} \in \mathbb{R}^{n \times n} \mid V_{ij} = 0 \quad \forall (i, j) : A_{ij}^{\mathcal{O}} = 0\}$ be the set of constrained matrices we are interested in. Any linear combination of matrices in \mathcal{C} is still in \mathcal{C} ; thus, \mathcal{C} is a convex set. Therefore, since \mathcal{S}_+^n is convex as well, the intersection $\mathcal{I} = \mathcal{C} \cap \mathcal{S}_+^n$ (which is the set over which f is maximized in (32) to compute h) is convex too.

The function f is continuous and strictly concave in \mathcal{I} , since $\mathcal{I} \subseteq \mathcal{S}_+^n$. Hence, by the maximum theorem under convexity (Sundaram, 1996), the argmax mapping of $f(\boldsymbol{\Theta}; \mathbf{V})$ within \mathcal{I} is a continuous function of \mathbf{V} . Consequently, h is a continuous mapping.

Consider that by the law of large numbers,

$$\mathbf{S} = \frac{1}{k} \sum_{i=1}^k \mathbf{x}_i \mathbf{x}_i^\top \xrightarrow{k \rightarrow \infty} \mathbb{E} [\mathbf{x} \mathbf{x}^\top] = \boldsymbol{\Theta}_0^{-1}. \quad (36)$$

Consequently, by the continuous mapping theorem (which can be applied because h is continuous), considering (36), (33) and (35), we conclude that

$$\hat{\boldsymbol{\Theta}} = h(\mathbf{S}) \xrightarrow{k \rightarrow \infty} h(\boldsymbol{\Theta}_0^{-1}) = \boldsymbol{\Theta}_0. \quad (37)$$

A.2 Proof of Lemma 2

To study how $\hat{p}(\mathbf{A} \mid \mathbf{X})$ behaves as $k \rightarrow \infty$, we first analyze the likelihood function $\mathcal{L}_{\mathbf{X}}(\boldsymbol{\Theta})$. Using Bayes' theorem, we get

$$p(\boldsymbol{\Theta} \mid \mathbf{X}) \propto \mathcal{L}_{\mathbf{X}}(\boldsymbol{\Theta}) p(\boldsymbol{\Theta}). \quad (38)$$

By the Bernstein–von Mises theorem, we know that $p(\boldsymbol{\Theta} \mid \mathbf{X}) \xrightarrow{k \rightarrow \infty} \delta(\boldsymbol{\Theta} - \boldsymbol{\Theta}_0)$, where $\delta(\cdot)$ is the Dirac delta. Combining this with (38) it follows that

$$\mathcal{L}_{\mathbf{X}}(\boldsymbol{\Theta}) \xrightarrow{k \rightarrow \infty} 0 \quad \forall \boldsymbol{\Theta} \neq \boldsymbol{\Theta}_0. \quad (39)$$

The result in (39) holds for any prior $p(\boldsymbol{\Theta})$ since the prior can be considered a constant with respect to k .

Given (39) and the consistency of $\hat{\boldsymbol{\Theta}}$ in Lemma 1, we conclude that [cf. (10) in the main paper]

$$\hat{p}(\mathbf{A} \mid \mathbf{X}) \xrightarrow{k \rightarrow \infty} \frac{p(\mathbf{A})}{C} \delta(\boldsymbol{\Theta}_0 \circ (\mathbf{A} + \mathbf{I}) - \boldsymbol{\Theta}_0), \quad (40)$$

where C is a normalization constant.

B EXPERIMENTAL DETAILS

B.1 Hyperparameters

Regarding the Langevin sampler, in all the experiments, we use $L = 10$ noise levels, evenly spaced between $\sigma_1 = 0.5$ and $\sigma_L = 0.03$, and $T = 300$ steps per level. We set the step size at $\epsilon = 10^{-6}$.

To tune the regularization parameter λ for the graphical lasso, we fit a function of the form $\lambda(k) = a \log(k)^2 + b \log(k) + c$ using a training set and then evaluate that function at inference time for the given value of k . We selected this specific functional form as it showed a very satisfactory fit for all of our cases. An example is shown in Figure 5.

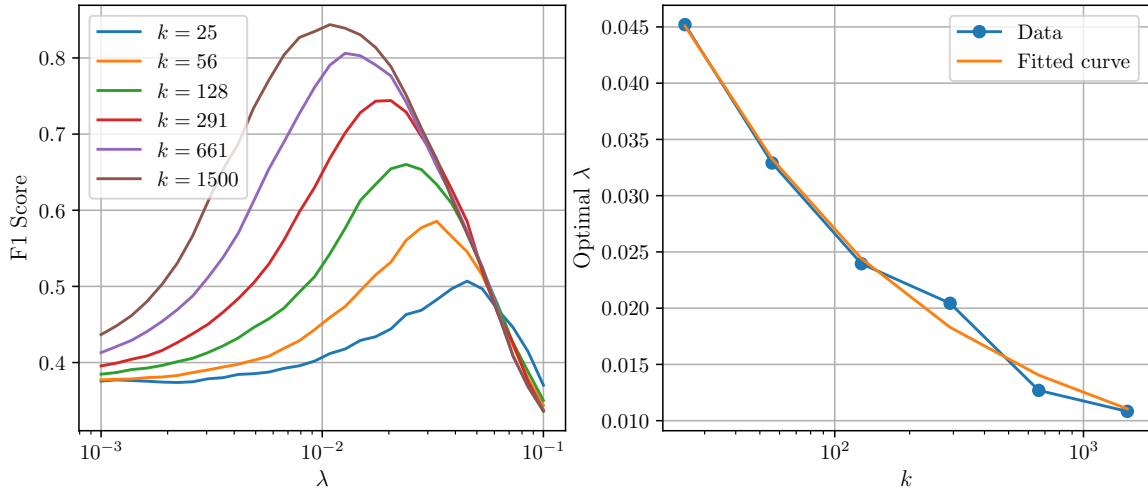


Figure 5: Fitting of $\lambda(k) = a \log(k)^2 + b \log(k) + c$ for the ego-nets dataset with $|\mathcal{U}| = 0.5 \dim(\mathbf{a})$. The orange curve on the right is the one used at inference time.

The thresholds used to obtain a binary-valued estimate of \mathbf{A}_0 when using GraphSAGE, the thresholding of $\hat{\boldsymbol{\Theta}}$ and our method (in this last case, the threshold is τ_k) are set using the same procedure. Given R simulations (in all of our experiments, $R = 100$), we take $R/2$ of those continuous-valued matrices and find the threshold within some grid \mathcal{T} that works the best for each method. Then, with those tuned values, we threshold the other $R/2$ matrices left and only evaluate performance over them. All the plots shown in Section 4 from the original

Algorithm 2 Metric computation

Require: $\{\mathbf{a}_r^{\mathcal{U}}, \tilde{\mathbf{a}}_r^{\mathcal{U}}\}_{r=1}^R, \mathcal{T}, S$

```

1:  $J \leftarrow 0$ 
2: for  $s \leftarrow 1$  to  $S$  do ▷ Repeat the process with  $S$  different splits
3:   Store  $R/2$  randomly chosen tuples  $(\mathbf{a}_r^{\mathcal{U}}, \tilde{\mathbf{a}}_r^{\mathcal{U}})$  into  $\mathcal{A}_{\text{train}}$ 
4:   Store the other  $R/2$  tuples in  $\mathcal{A}_{\text{test}}$ 
5:   for  $\tau \in \mathcal{T}$  do ▷ Get the best threshold in the grid
6:     Compute  $\text{metric}(\mathbf{a}_r^{\mathcal{U}}, \mathbb{I}\{\tilde{\mathbf{a}}_r^{\mathcal{U}} \geq \tau\})$  for each tuple in  $\mathcal{A}_{\text{train}}$ 
7:     If the average train metric is the best so far, store the current threshold in  $\tau^*$ 
8:   end for
9:   Compute  $\text{metric}(\mathbf{a}_r^{\mathcal{U}}, \mathbb{I}\{\tilde{\mathbf{a}}_r^{\mathcal{U}} \geq \tau^*\})$  for each tuple in  $\mathcal{A}_{\text{test}}$ 
10:  Store the average of the test metrics in  $J_s$ 
11:   $J \leftarrow J + J_s$ 
12: end for
13: return  $\frac{J}{S}$  ▷ Return the mean across splits

```

paper, as well as in this document, correspond to averages following this procedure S times (i.e., across $S = 10$ different train/test splits in our case), each with its threshold. Algorithm 2 provides a more precise description of our method.

B.2 Datasets’ details

For the grids, we generated graphs with $40 \leq n \leq 50$. Additionally, to introduce randomness into the graphs, we uniformly added between 2 and 5 edges at random to the edge set \mathcal{E} . We trained a GNN with $|\mathcal{A}| = 5000$ graphs generated with the described procedure.

In the case of the dual Barabási–Albert model, the generation is initialized with an empty graph with $\max(n_1, n_2)$ nodes. Then, the remaining $n - \max(n_1, n_2)$ vertices are added iteratively. For each new node, n_1 edges are added with probability π , and n_2 edges are added with probability $1 - \pi$. Edges are added following a preferential attachment criterion. In our case, we set $n_1 = 2$, $n_2 = 4$ and $\pi = 0.5$. We simulated $|\mathcal{A}| = 1000$ graphs in this setting.

Regarding the ego-nets, we used $|\mathcal{A}| = 2926$ of the graphs available in the whole dataset (i.e., only those such that $n \leq 25$) to train the EDP-GNN, leaving 100 graphs for testing.

B.3 Implementation details

All experiments shown in the main paper and in this document and the training of the EDP-GNNs were run on an NVIDIA DGX A100 system. Our code is provided as part of the supplementary material and not in a GitHub repository for the sake of anonymity. The code corresponding to the EDP-GNN implementation was taken from the original repository of [Niu et al. \(2020\)](#) under a GPL-3.0 license.

Regarding the computational complexity, the key steps to consider from Algorithm 1 are those in lines 2, 11, and 12. The computation time of $\hat{\Theta}$ is $\mathcal{O}(n^3)$ for dense problems, but much less in sparse ones ([Friedman et al., 2008](#)), and even less if some of the entries are fixed as is the case of (9). However, here, we consider the worst-case scenario. After computing $\hat{\Theta}$, we evaluate (27), which requires to compute the inverse of $\hat{\Theta}$, leading to a time complexity of $\mathcal{O}(n^3)$ as well. Feed-forwarding the GNN consists of just matrix and vector multiplications that scale as $\mathcal{O}(n^2)$. Therefore, the dominating steps are the first two. The step in 11 is repeated $L \cdot T$ times per sample, and since the sampling of each \mathbf{A}_i is completely parallelizable, the total complexity of Algorithm 1 is $\mathcal{O}(L \cdot T \cdot n^3)$, considering that typically $LT \gg 1$.

C ADDITIONAL EXPERIMENTS

This section provides more experiments to gain additional insights into our method.

C.1 Performance dependence on $|\mathcal{A}|$

We want to analyze how the predictive power of the prior learned by the EDP-GNN changes when different dataset sizes $|\mathcal{A}|$ are used for training. To this end, we use a different family of graphs, known as *exponential random graph models* (ERGMs), which has a closed-form distribution (up to a normalization constant) that relies on a set of network statistics and parameters (Hunter and Handcock, 2006; Snijders et al., 2006). Namely, for a vector of r statistics $\psi(\mathbf{A}) = [\psi_1(\mathbf{A}) \ \psi_2(\mathbf{A}) \ \cdots \ \psi_r(\mathbf{A})]^\top$ and a set of parameters $\beta \in \mathbb{R}^r$, the distribution of \mathbf{A} is given by

$$p(\mathbf{A}) = \frac{1}{C_\psi(\beta)} \exp\left(\beta^\top \psi(\mathbf{A})\right). \quad (41)$$

The statistics can be, e.g., the number of triangles in the graph, the number of d -stars, or the number of edges, among many others. In the simulations, we consider an ERGM distribution with statistics

$$\psi(\mathbf{A}) = [\text{AKS}_\gamma(\mathbf{A}) \ \frac{1}{2} \sum_{ij} A_{ij}]^\top. \quad (42)$$

The first one corresponds to the alternated d -stars statistic (Hunter and Handcock, 2006), defined as

$$\text{AKS}_\gamma(\mathbf{A}) = \sum_{d=2}^{p-1} (-1)^d \frac{S_d(\mathbf{A})}{\gamma^{d-2}}, \quad (43)$$

with $S_d(\cdot)$ being the number of d -stars in the given graph and γ a constant. In our simulations, we set $\gamma = 0.3$. The second statistic corresponds to the number of edges $|\mathcal{E}|$. We use $\beta = [0.7 \ -2]^\top$ as coefficients. The graphs we generate have $n = 50$ nodes, and we remove $|\mathcal{U}| = 30$ values from \mathbf{A}_0 . We drop zeros and ones with equal probability so that we can use accuracy instead of F1 score for this experiment. We generated $|\mathcal{A}| = 1000$ graphs from this distribution to train the EDP-GNN and, to evaluate the impact the dataset size $|\mathcal{A}|$ has on the estimation performance, we compare the edge prediction accuracy when fewer training samples are used. The reported accuracies correspond to the average over 25 runs, following the procedure described in Section B.1.

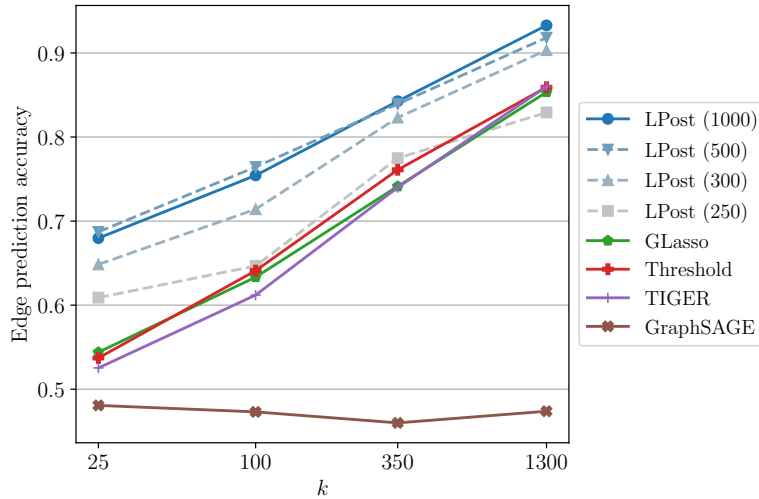


Figure 6: Prediction accuracy of several methods using ERGM graphs with $|\mathcal{U}| = 30$. The comparison includes five versions of Algorithm 1, each considering a prior that was learned with datasets \mathcal{A} of different sizes (the value of $|\mathcal{A}|$ is indicated in the legend), as well as three benchmarks available in the literature.

Figure 6 shows that the accuracy improves drastically as $|\mathcal{A}|$ increases, as expected. However, this enhancement seems to saturate as the dataset size approaches $|\mathcal{A}| = 1000$. All the other GGM estimation methods perform similarly, logically presenting a higher prediction accuracy as k increases. Notably, whenever $|\mathcal{A}| \geq 300$, all other methods underperform our algorithm.

GraphSAGE performs poorly – it predicts 0 or 1 uniformly at random. Additionally, the incorporation of additional observations does not increase its accuracy. As discussed in detail in the main paper, this behavior is

expected and associated with GraphSAGE aggregating data using local neighborhoods, which is not a good fit for the setup at hand.

C.2 Performance dependence on $|\mathcal{U}|$

In order to understand how $|\mathcal{U}|$ impacts the estimation performance, we run an additional experiment that complements the results presented in Section 4 when using grid graphs with different $|\mathcal{U}|$. We use the test set of the ego-nets and fix $k = 100$, varying the percentage of unknown values in \mathbf{a} . We run the simulations using three versions of Algorithm 1:

- **Posterior (LPost)**. Our original approach, implementing all the steps in Algorithm 1.
- **Prior (LPr)**. This simplified version of Algorithm 1 only exploits prior information. This corresponds to running Algorithm 1 where in line 11 we set $\nabla \log \mathcal{L}_{\mathbf{X}}(\tilde{\Theta}_t) = 0$.
- **Likelihood (LL)**. This simplified version of Algorithm 1 omits the learned prior and uses only the observations \mathbf{X} . More specifically, this entails to run a version of Algorithm 1 where in line 13 we set $\mathbf{g}(\hat{\mathbf{A}}_{t-1}, \sigma_t) = 0$.

We use the AUC score instead of F1 in this experiment to assess the prediction performance. This allows us to avoid the tuning of τ_k , which is unnecessary for this analysis since we are comparing three versions of Algorithm 1 that return continuous predictions if no thresholding is applied. Results are shown in Figure 7.

The main observations are: i) the information provided by \mathcal{A} decreases as $|\mathcal{U}|$ increases, and ii) LPost consistently outperforms LPr and LL. Both are expected results that were observed in previous experiments. Analyzing more specific details, we note that the performance of LPr drops rapidly as $|\mathcal{U}|$ increases, while that of LPost decays more slowly. The same is true for LL, whose AUC levels are stable and decay slowly. Finally, it is worth noticing that when a large portion of the graph is unknown, LPost approaches LL since the information provided by $|\mathcal{A}|$ becomes less valuable.

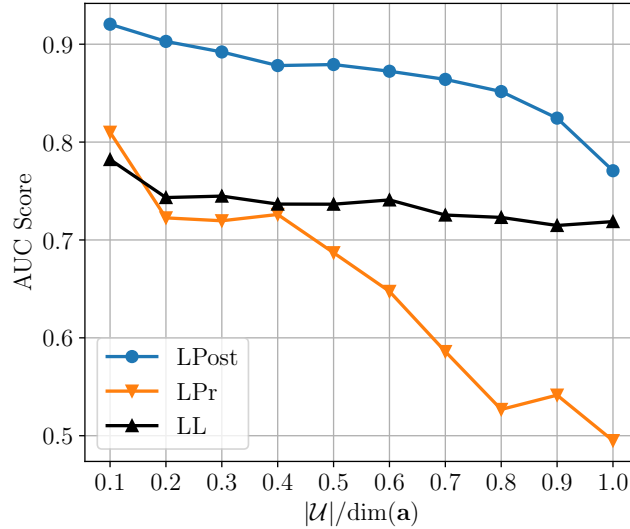


Figure 7: AUC score achieved by different versions Algorithm 1 when using an ego-net prior and $k = 100$ observations. The horizontal axis represents different values of $|\mathcal{U}|$. For this experiment we skip the thresholding implemented in line 24 of Algorithm 1.