Computationally Efficient Active Learning of Gaussian Graphical Models

Abrar Zahin
Electrical, Computer, and Energy Engineering
Arizona State University
Tempe, United States of America
azahin@asu.edu

Gautam Dasarathy
Electrical, Computer, and Energy Engineering
Arizona State University
Tempe, United States of America
gautamd@asu.edu

***Abstract*—The graphical model selection problem is vital in various applications and has garnered significant attention in recent years. In many applications traditional approaches face significant limitations as acquiring a large number of samples from the entire system concurrently often proves to be prohibitively expensive. For instance, in sensor networks, this requires expensive synchronization procedures across the sensors. In proteomics, it requires simultaneous tagging of a large number of proteins. While recent approaches have been proposed for efficiently and adaptively acquiring samples to overcome these difficulties, they suffer from prohibitive computational costs. Our paper introduces a novel algorithm that combines adaptive sample acquisition with a method based on the multiplicative weights update meta-algorithm [1]. We show that this algorithm enjoys significantly better computational efficiency while also being efficient in sample acquisition.**

***Index Terms*—computational efficiency, Gaussian graphical models, active learning**

I. Introduction

Probabilistic graphical models have emerged as a powerful framework to express and leverage relationships among entities in large interacting systems [27]. Their applications span various domains, including power systems [3, 6, 24], signal processing [7], (phylo)genomics [9, 10, 5], and neuroscience [4, 22]. In short, graphical models represent statistical relationships using a graph. The vertices in the graph represent random variables, and the edges in the graph represent conditional dependence between the corresponding variables. For a comprehensive introduction on graphical models, we refer the reader to [27]. An important subclass of graphical models is that of Gaussian graphical models, or Gauss Markov random fields, which is the focus of this paper. In several application domains we do not know the underlying graph structure and the goal is to learn this from data — a problem dubbed graphical model selection. This is important not only because the model can provide a succinct representation of a potentially complex multivariate distribution, but also because such models can in fact reveal important relationships among the underlying variables. This problem has been extensively studied by several authors (see e.g., [18, 29] and references therein).

Traditional algorithms for learning graphical models are *passive* and require several measurements across all the vertices in the graph. In many scenarios, acquiring such measurements could be costly or impractical. For instance, in a sensor network, obtaining samples from all sensors simultaneously requires synchronizing measurements from each sensor. Likewise, in many other fields such as neuroscience [13, 14, 15, 22] and proteomics [16], obtaining (marginalized) samples from small subsets of variables may be more feasible than capturing complete snapshots.

A recent line of work [10, 25] has pioneered a sequential and adaptive data acquisition strategies that addresses this problem. However, the benefits these algorithms deliver in data-acquisition efficiency come at the cost of prohibitvely high computational complexity. For instance, the active learning algorithm proposed in [28] incurs a computation cost of $\mathcal{O}(p^4)$ to learn a graph with p vertices. This could be prohibitive in modern problems where the number of variables (vertices) reach tens of thousands.

Contributions. In this paper, we propose a computationally efficient active learning algorithm based on the multiplicative weight update method (as pioneered in the context of graph learning in [1, 2]) incorporated into the active learning framework proposed in [28]. We demonstrate that the resulting algorithm efficiently learns the graph structure with rigorous guarantees on the required number of samples. Further, this algorithm enjoys a lower run time complexity of the order of p^3 rather than the p^4 dependency found in the algorithm by [28], where p represents the number of vertices. On the sample complexity front, while there is room for improvement, our method shares similar assumptions and qualitative dependencies with existing works (see [8] and references therein) that rely on condition-number-type assumptions. Moreover, our algorithmic approach can be applied directly in an online setting (even in the active learning framework), where samples arrive one-by-one and are processed without the need for storage before the arrival of the next sample.

II. Preliminaries

Graph theory. Let $G = (V, E)$ be an undirected graph on p vertices with the vertex set V and the edge set $E \subset \binom{V}{2}$. We

assume that G contains no self-loops. The neighborhood of a vertex $v \in V$ is given by the set $N(v) \triangleq \{u \in V : \{u, v\} \in E\}$ and the *degree* $d(v)$ of the vertex is defined as the size of $N(v)$. Let's define the closure of neighborhood $\bar{N}(v)$ as $N(v) \cup \{v\}$. Furthermore, $d_{\max} \triangleq \max_{v \in [p]} d(v)$ defines the maximum degree of a graph, and let $d^v_{\max} \triangleq \max_{j \in \bar{N}(v)} d(j)$ defines the maximum degree of v's closed neighborhood. Given a pair of vertices $u, v \in V$, a sequence of distinct vertices $v_1 = u, v_2, \ldots, v_k = v$ such that $\{v_i, v_{i+1}\} \in E$, for $1 \leq i < k$, is called a *path* between the vertices u, v. We let $\mathcal{P}_{uv}$ denote the set of all paths between u and v. If $\mathcal{P}_{uv}$ is not empty, we say those vertices are connected. The graph G is said to be connected if every pair of vertices in G is connected.

Gaussian graphical models. Let $\mathbf{X} = (X_1, X_2, \ldots, X_p) \in \mathbb{R}^p$ be a zero-mean Gaussian random vector with a covariance matrix $\Sigma \in \mathbb{R}^{p \times p}$. Compactly, $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, where $\mathbf{0}$ is the p-dimensional vector of all zeros. The density of X is given by

$$f_X(x_1, x_2, \ldots, x_p) = \frac{1}{\sqrt{(2\pi)^p |K|}} \exp\{-\frac{1}{2} x^T K x\}.$$

The distribution of $\mathbf{X}$ is said to be a *Gaussian graphical model* (or equivalently, Markov) with respect to a graph G if $(\Sigma^{-1})_{ij} = 0$ for all $\{i, j\} \notin E$. $K \triangleq \Sigma^{-1}$ is called the *precision matrix* of $\mathbf{X}$. In other words, for any $\{i, j\} \notin E$, X_i and X_j are conditionally independent given all the other coordinates of $\mathbf{X}$; we refer the reader to [27] for a more thorough exposition on graphical models. An important problem associated with Gaussian graphical models is one of structure learning where one aims to learn the edge set of the underlying graph from data. Formally, given n i.i.d samples from the distribution f_X, the problem of *model selection* or *structure learning* is to estimate the edge set E, or equivalently, the support of the precision matrix K. For recent accounts of results on structure learning, we refer the reader to [18] and [29].

III. Related Work

Several algorithms have been developed for the problem of learning Gaussian graphical models using a variety of techniques and assumptions. A comprehensive survey of all those works is beyond the scope of this work. Interested readers can take a look at [18] and the references therein. In scenarios where exhaustive sampling of the graph is infeasible or expensive, there is a recent line of work [28, 25] that considers sequential and adaptive data acquisition strategies. However, as mentioned above, they have prohibitively high computational complexities of $\mathcal{O}(d_{\max} \times p^4)$ and $\mathcal{O}(p^{d_{\max}+2})$, respectively, where $d_{\max}$ denotes the maximum degree of a graph. While there exist several other frameworks for active/sequential data acquisition that is non-exhaustive [11, 12, 17], these are only applicable in the special case of tree-structured graphs.

IV. Problem Setup

In this paper, we investigate the task of *computationally efficiently* recovering the graph structure of G using samples drawn from the underlying distribution f_X. Our focus is on a scenario described in Section I, where obtaining (joint) measurements is prohibitively expensive. Therefore, in addition to assessing the computational runtime, we adopt the total number of scalar samples as a key metric for evaluating our algorithm's performance compared to a passive counterpart (as introduced in [28]). Notably, the total number of scalar samples inherently accounts for the expenses associated with acquiring synchronous samples from a substantial subset of variables. To this end, we now define the total sample complexity, followed by our problem statement.

Definition 1 (Total Sample Complexity)**.** *Fix $\delta \in (0, 1)$. Suppose that an algorithm returns an estimate $\widehat{E}_n$ given a budget of n total scalar samples. We will say that its total sample complexity at confidence level δ is n_0 if for all $n \geq n_0$, $\mathbb{P}(\widehat{E}_n \neq E) \leq 1 - \delta$.*

Problem Statement. Let $(G, \mathcal{N}(\mathbf{0}, \Sigma))$ be a Gaussian graphical model on p vertices. Our goal is to construct an estimate $\widehat{E}$ of the edge set of the underlying graph given samples from the distribution f_X by (1) minimizing the computational resources, as measured by time complexity, and (2) maintaining a competitive (scaler) sample complexity.

V. Algorithm

In this section we present our algorithm. Before that, we will define some quantities. We start with minimum normalized edge strength: $\kappa = \min_{(i,j) \in E} \frac{K_{ij}}{\sqrt{K_{ii} \times K_{jj}}}$. This quantity appears in our theory as more data is required (i.e., longer latency is needed) to learn weaker edges. Next, we define an upper bound $k_{\max}$ on the absolute values of the entries of K, and an upper bound on the variance of any marginal variable.

$$k_{\max} = \max_{(i,j)} |K_{ij}|$$
$$\sigma_{\max} = \max_{i} \mathrm{Var}[\mathrm{X_i}]$$

We now ready present our algorithm. Recall that we adopt the algorithm in [1] in the framework of [28]. We first start with the framework.

The framework operates with a specified budget of scalar samples based on a condition dependent on graph parameters at each iteration. The algorithm relies on two subroutines, SeqNbdSearch and nbdVerify. It begins with an empty graph and progressively estimates neighborhoods using SeqNbdSearch subroutine. The algorithm iterates until it identifies the neighborhood of each vertex in $[p]$. In each iteration, the algorithm keeps track of the vertices for which the neighborhood has been found. If the neighbors of all the neighbors of a vertex i are found, the algorithm does not sample i in subsequent iterations. The SETTLED set in Algorithm 1 keeps track of such vertices. This "settling" step improves the total scalar sample complexity. In the following, we briefly discuss our SeqNbdSearch subroutine.

Sequential Neighborhood Search subroutine, SEQNBDSEARCH. We start the description of SEQNBDSEARCH subroutine (given in Subroutine 1) by noting an important property of multivariate Gaussians. For any $i \in [p]$, the conditional expectation of X_i given $\mathbf{X}_{\setminus i}$ satisfies

$$\mathbb{E}[X_i|\mathbf{X}_{\setminus i}] = \sum_{j \neq i} \frac{-K_{ij}}{K_{ii}} X_j.$$

Further, by the Gauss-Markov theorem [26], we know that X_i conditioned on $\mathbf{X}_{\setminus i}$ is in-fact normally distributed and can be written as

$$X_i = \sum_{j \neq i} w_j X_j + \eta_i, \tag{1}$$

where $w_j = -K_{ij}/K_{ii}$ and $\eta_i \sim \mathcal{N}(0, 1/K_{ii})$. Further, we know that η_i is independent of $\{X_j, j \neq i\}$. That is, if the graph G has a degree of d, then we know that X_i can be written as a noisy linear combination of $\mathbf{X}_{\setminus i}$ with at most d non-zero coefficients. Furthermore, $w^i_{\max} \triangleq \max_{j \in N(i)} |w_j|$ and $w_{\max} \triangleq \max_{i \in [p]} w^i_{\max}$. While designing our algorithm we will assume that $w_{\max}$ is known to us.

We will now (briefly) discuss the algorithm in [1], which employs a straightforward majority weighted voting mechanism. For each potential member X_j within the neighborhood of node i, the algorithm keeps track of a weight which basically records the approximate of each coordinate of the weight vector. This weight vector is initialized with an uniform weight vector and undergoes updates in a multiplicative manner, similar to the Hedge algorithm [23]. The update in each coordinate of a weight vector depends on how at a given iteration the coordinate performed in predicting the response variable. After a series of some consecutive updates, the algorithm utilizes some additional samples to empirically assess the expected risk (see Definition 2) for each of the candidates from the former stage. Subsequently, it selects the candidate with the lowest empirical risk for the final result.

Definition 2. *Let $\widehat{\mathbf{w}}^i \in \mathbb{R}^p$ and $\mathbf{w}^i \in \mathbb{R}^p$ be the candidate weight vector and true weight vector of neighborhood of marginal variable x_i, respectively. Then, the expected risk $\epsilon(\widehat{\mathbf{w}}^i)$ is defined as*

$$\epsilon(\widehat{\mathbf{w}}^i) = \mathbb{E}_X[\widehat{\mathbf{w}}^i \cdot \mathbf{X}_{-i} - \mathbf{w}^i \cdot \mathbf{X}_{-i}]^2$$

.

Once a candidate weight vector is found using proper threshold, we get a candidate neighborhood for all vertices in specific iteration. Then, NBDVERIFY() takes a vertex i, a candidate neighborhood $\widehat{N}(i)$ returned by SEQNBDSEARCH, and samples from the variables that have not been settled yet, i.e., $[p] \setminus$ SETTLED. It checks whether the returned neighborhood is indeed a potential neighborhood of i or not.

Algorithm 1 A FRAMEWORK FOR ACTIVE SEQUENTIAL LEARNING FOR GGMS

Input: $\kappa, \sigma_{max}, k_{max}, d_{max}, w_{max}$, sample complexity functions $s() \triangleq s'() + s''()$ (see [1]) and $v()$
Output: An edge set, $\widehat{E}$
Initialization: $r = 1$, NBDFOUND, SETTLED $= \emptyset$, $\widehat{N}_i \triangleq \emptyset$ for all $i \in [p]$
while NBDFOUND $\neq [p]$ or $r < 2p$ **do**
  for $i \in$ NBDFOUND^c **do**
    $\lambda_r = r \times w_{\max}$ ▷ Set λ on the $r-$th stage.
    $\widehat{N}_i, F =$ SEQNBDSEARCH$(i, s(r), \lambda_r, \{X^j_{\text{SETTLED}^C}\}_{j \in \{1,2,...}$
    if NBDVERIFY$(F, i, \widehat{N}_i, \{X^j_{\text{SETTLED}^C}\}_{j \in \{1,2,...,v(r)\}}) =$ **True then**
      NBDFOUND $\leftarrow$ NBDFOUND $\cup \{i\}$
    end if
  end for
  for $i \in$ NBDFOUND **do**
    if $\widehat{N}(i) \subseteq$ NBDFOUND **then**
      SETTLED $\leftarrow$ SETTLED $\cup \{i\}$
    end if
  end for
  $r \leftarrow r * 2$
end while

Subroutine 1 Estimating Neighborhood of a vertex i

Input: Normalizing parameter $C_n \triangleq \frac{1}{\sqrt{2\sigma_{\max}(\lambda_r+1)\log\frac{2ps'(r)}{\delta}}}$; $(\tilde{x}^t, \tilde{y}^t) = C_n(x^t, y^t)$, learning rate β; λ_r.
Output: A candidate neighborhood $\widehat{N}_i$.
Let $|\text{SETTLED}^c| = m$ Initialization: $\mathbf{v}^0 = \{\frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m}\} \in \mathbb{R}^m$
for $s = 1, \ldots, s'(r)$ **do**
  Compute $\mathbf{p}^s = \frac{\mathbf{w}^{s-1}}{\|\mathbf{w}^{s-1}\|_1}$
  $\boldsymbol{\ell}^s = (1/2)(\mathbf{1} + (\lambda_r \mathbf{p}^s \cdot \widetilde{\mathbf{x}}^s_{-i} - y^s))\widetilde{\mathbf{x}}^s_{-i})$
  $\forall i \in$ SETTLED^c, $v_i^{s-1} = v_i^{s-1}\beta^{l_i}$
end for
for $s = 1, 2, \ldots, s''(r)$ **do**
  Compute empirical risk for all $s'(r)$ weight vectors using $s''(r)$ samples
end for
$\widehat{\mathbf{w}} =$ weight vector with minimum empirical risk.
$w_i = \begin{cases} 0 & \text{if } w_i < \frac{2\kappa}{3} \\ v_i & \text{otherwise} \end{cases}$

if $|\text{supp}(\widehat{\mathbf{w}})| > r$ **then**
  $\widehat{N}_i \triangleq$ Top r coordinates of $\widehat{\mathbf{w}}_i$
  Let F be the coordinates associated with r
else
  $\widehat{N}_i \triangleq \text{supp}(\widehat{\mathbf{w}})$
end if

Subroutine 2 Verifying Neighborhood of a vertex i

1: **Input:** $F, i, \widehat{N}_i$, and SETTLED
2: **Output:** TRUE or FALSE.
3: **if** for each $j \in [p] \setminus F \cup \text{SETTLED} \cup \{i\}, |\hat{\rho}_{i,j \setminus \text{SETTLED}}| \leq \zeta$ **then** Return TRUE
4: **else** Return FALSE
5: **end if**

VI. Theoretical Guarantees

We will now discuss the main theoretical result. Before that we will state a result from [28], as a Lemma which will be instrumental for our main result on sample complexity.

Theorem 1. *Let $(G, \mathcal{N}(\mathbf{0}, \Sigma))$ be a Gaussian graphical model on p vertices with parameters $(\kappa, \sigma_{\max}, k_{\max}, d_{\max}, w_{\max})$ as defined above. Fix $\delta > 0$. There exists a constant $C > 0$ such that Algorithm 1 exactly returns the structure of G with probability at least $1 - \delta$ provided the total sample complexity is at least*

$$C \frac{1}{p} \sum_{i=1}^{p} (d^i_{\max} w_{\max})^4 \times \frac{\sigma^2_{\max} k^2_{\max}}{\kappa^4} p \log^3 \left(\frac{p d_{\max}}{\delta} \right) \quad (2)$$

Proof Sketch. Fix an arbitrary vertex $i \in [p]$. From Theorem 1 in [28] we have that in order to proof the theorem it suffices to determine the number of samples SEQNBDSEARCH requires to correctly recover the neighborhood of i with probability greater than $1 - \frac{\delta}{2pd_{\max}}$. From [1] we have that one requires $O\left(\frac{(r \times w_{\max})^4 (\sigma_{\max} \times k_{\max})^2}{\kappa^4} \log^3 (\frac{p d_{\max}}{\delta}) \right)$ samples to satisfy the above provided the degree of i is smaller than r. We then use the "waterfilling" style argument introduced by [28] to deduce the final total sample complexity. □

Theorem 2 (Main Theorem on Computational Complexity)**.** *The computational complexity of Algorithm 1 is bounded from above by $d_{\max} \times p \times C_{\text{seq}}$, where C_{seq} is the computational cost Subroutine 1 which is $O(p^2)$.*

Proof. In the for loop of Algorithm 1, our algorithm employs SPARSITRON to learn (or select) the neighborhood. This means that SPARSITRON runs in parallel for vertices in NBDFOUND^c. Note that a single instance of SPARSITRON has a time complexity on the order of p^2 in high-dimensional settings. □

Advantage over Active Algorithms in terms of Computational Complexity. Notice that [28] employs LASSO in their proposed framework which runs in parallel for vertices in NBDFOUND^c. It's important to note that a single instance of LASSO has a time complexity on the order of $O(p^3)$ in high-dimensional settings [30], whereas our Subroutine 1 has a complexity of $O(p^2)$.

Advantage over Passive Algorithms in terms of Sample Complexity. The algorithm presented in [1] can be characterized as a passive (albeit potentially online) variant of the computationally efficient algorithm proposed in this paper. The total sample complexity requirements implied in that paper essentially has the fourth power of the ℓ_1 norm of the corresponding row of the precision matrix in the place of $\frac{1}{p} \sum_{i=1}^{p} \left(d^i_{\max} w_{\max} \right)^4$ in (2). The latter can be significantly smaller in heterogeneous graphs whose degree distribution varies significantly, which results in significant savings in the total sample complexity.

VII. Conclusion

In this paper, we propose a computationally efficient active learning algorithm using the framework presented in [28]. We demonstrate that our algorithm achieves an exponential reduction in p in terms of computational complexity, where p represents the number of vertices in a graph— which can be very crucial in high-dimensional settings. Notably, our approach is suitable for direct application in an online setting within the active learning framework, handling one-by-one sample arrivals without the need for intermediate storage.

References

[1] A. Chaturvedi and J. Scarlett, *Learning Gaussian Graphical Models via Multiplicative Weights*, *arXiv preprint arXiv:2002.08663*, 2020.

[2] A. Klivans and R. Meka, "Learning graphical models using multiplicative weights," in *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 343–354, IEEE, 2017.

[3] R. Anguluri, G. Dasarathy, O. Kosut, and L. Sankar, *Grid Topology Identification With Hidden Nodes via Structured Norm Minimization*, *IEEE Control Systems Letters*, vol. 6, pp. 1244-1249, 2022.

[4] E. T. Bullmore and D. S. Bassett, *Brain Graphs: Graphical Models of the Human Brain Connectome*, *Annual Review of Clinical Psychology*, vol. 7, pp. 113-140, 2011.

[5] G. Dasarathy, E. Mossel, R. Nowak, and S. Roch, *A Stochastic Farris Transform for Genetic Data under the Multispecies Coalescent with Applications to Data Requirements*, *Journal of Mathematical Biology*, vol. 84, no. 5, pp. 1-37, 2022.

[6] D. Deka, S. Talukdar, M. Chertkov, and M. V. Salapaka, *Graphical Models in Meshed Distribution Grids: Topology Estimation, Change Detection & Limitations*, *IEEE Transactions on Smart Grid*, vol. 11, no. 5, pp. 4299-4310, 2020.

[7] M. Kim and P. Smaragdis, *Single channel source separation using smooth nonnegative matrix factorization with Markov random fields*, in *2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, 2013, pp. 1–6.

[8] T. Cai, W. Liu, and X. Luo, "A constrained ℓ_1 minimization approach to sparse precision matrix estimation," *Journal of the American Statistical Association*, vol. 106, no. 494, pp. 594–607, 2011, Taylor & Francis.

[9] Y. Zuo, Y. Cui, G. Yu, R. Li, and H. W. Ressom, *Incorporating prior biological knowledge for network-based differential gene expression analysis using differentially*

weighted graphical LASSO, *BMC bioinformatics*, vol. 18, no. 1, pp. 1–14, Springer, 2017.

[10] G. Dasarathy, R. Nowak, and S. Roch, *Data requirement for phylogenetic inference from multiple loci: a new distance method*, *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 12, no. 2, pp. 422–432, IEEE, 2014.

[11] G. Lugosi, J. Truszkowski, V. Velona, and P. Zwiernik, *Structure learning in graphical models by covariance queries*, *arXiv preprint arXiv:1906.09501*, 2019.

[12] A. Krishnamurthy and A. Singh, *Robust multi-source network tomography using selective probes*, in *2012 Proceedings IEEE INFOCOM*, pp. 1629–1637, IEEE, 2012.

[13] D. Soudry, S. Keshri, P. Stinson, M.-h. Oh, G. Iyengar, and L. Paninski, *A shotgun sampling solution for the common input problem in neural connectivity inference*, *arXiv preprint arXiv:1309.3724*, 2013.

[14] S. Turaga, L. Buesing, A. M. Packer, H. Dalgleish, N. Pettit, M. Hausser, and J. H. Macke, *Inferring neural population dynamics from multiple partial recordings of the same neural circuit*, *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[15] W. E. Bishop and B. M. Yu, *Deterministic symmetric positive semidefinite matrix completion*, *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[16] K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, and G. P. Nolan, "Causal protein-signaling networks derived from multiparameter single-cell data," *Science*, vol. 308, no. 5721, pp. 523–529, 2005. American Association for the Advancement of Science.

[17] M. J. Choi, V. Y. F. Tan, A. Anandkumar, and A. S. Willsky, *Learning latent tree graphical models*, *J. of Machine Learning Research*, vol. 12, pp. 1771–1812, Journal of Machine Learning Research, 2011.

[18] M. Drton and M. H. Maathuis, *Structure learning in graphical modeling*, *Annual Review of Statistics and Its Application*, vol. 4, pp. 365–393, Annual Reviews, 2017.

[19] S. Misra, M. Vuffray, and A. Y. Lokhov, "Information theoretic optimal learning of Gaussian graphical models," in *Conference on Learning Theory*, pp. 2888–2909, 2020. PMLR.

[20] J. Kelner, F. Koehler, R. Meka, and A. Moitra, "Learning some popular Gaussian graphical models without condition number bounds," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 10986–10998, 2020.

[21] A. Anandkumar, V. Y. F. Tan, F. Huang, and A. S. Willsky, "High-dimensional Gaussian graphical model selection: Walk summability and local separation criterion," *Journal of Machine Learning Research*, vol. 13, pp. 2293–2337, 2012.

[22] G. Vinci, G. Dasarathy, and G. I. Allen, *Graph Quilting: Graphical Model Selection from Partially Observed Covariances*, *arXiv preprint arXiv:1912.05573*, 2019.

[23] Y. Freund and R. E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, Elsevier, 1997.

[24] D. Deka, R. Baldick, and S. Vishwanath, *One Breaker is Enough: Hidden Topology Attacks on Power Grids*, in *2015 IEEE Power & Energy Society General Meeting*, pp. 1-5, 2015.

[25] D. Vats, R. Nowak, and R. Baraniuk, "Active learning for undirected graphical model selection," in *Artificial Intelligence and Statistics*, pp. 958–967, 2014. PMLR.

[26] J. A. Gubner, *Probability and Random Processes for Electrical and Computer Engineers*, *Cambridge University Press*, 2006.

[27] S. L. Lauritzen, *Graphical Models*, Clarendon Press, 1996.

[28] G. Dasarathy, A. Singh, M.-F. Balcan, and J. H. Park, *Active Learning Algorithms for Graphical Model Selection*, in *Artificial Intelligence and Statistics*, pp. 1356-1364, 2016, PMLR.

[29] M. Maathuis, M. Drton, S. Lauritzen, and M. Wainwright, *Handbook of Graphical Models*, CRC Press, 2018.

[30] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, *Least Angle Regression*, 2004.