ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3503-4485-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICASSP48485.2024.10447619

RAPID CHANGE LOCALIZATION IN DYNAMIC GRAPHICAL MODELS

Abrar Zahin, Weizhi Li, and Gautam Dasarathy

ECEE, Arizona State University
{azahin, weizhili, gautamd}@asu.edu

ABSTRACT

Gaussian graphical models have emerged as a powerful tool for modeling and understanding multivariate data across various domains. In this paper, we consider the problem of change localization in the Gaussian graphical model, where it is known that a change has occurred in the underlying graph structure, and the goal is to localize the change rapidly. This paradigm occurs in various applications, from cyber-physical systems and biological networks to social networks and epidemiology. We introduce a novel algorithm, dubbed FOLk-DGM (Fast Online Localization in Dynamic Graphical Models), that is both computationally efficient and performs change localization with provably low latency (time elapsed before the change is localized). We present the theoretical properties of the algorithm and complement our theoretical results with experimental results.

Index Terms— change localization, Gaussian graphical models, sequential learning

1. INTRODUCTION

Gaussian graphical models (GGM) have emerged as a powerful framework for capturing relationships and dependencies among multivariate data. These models have found applications in diverse domains ranging from social networks and genomics to finance and cyber-physical systems. Consequently, it is imperative to learn the structure of the graph that underlies the Gaussian graphical model. Unsurprisingly, learning the structure of GGM has been extensively studied [1, 2].

In this paper, we consider an important generalization of this problem where the underlying system is not static and evolves over time. In particular, we consider the problem of *change localization*, where we know a change has occurred in the system, and our goal is to localize it in the graph swiftly. This problem has received much less attention despite its relevance to several real-world applications. Considering a sprawling infrastructure network where a fault is detected by installed sensors, one needs to rapidly locate the source of the fault to maintain the operational integrity of the system. In epidemiology, the sudden contagion spread may be observed (e.g., via wastewater analysis [3]); however, identifying the portion of the network that gave rise to the contagion needs to be identified rapidly. Similarly, disruptions in global supply chains may be evident, but pinpointing the network location of such disruptions is key to swift resolution. Addressing this challenge, this paper makes the following contributions:

1. **A Novel Online Change Localization Algorithm.** Our algorithm, dubbed FOLk-DGM (Fast Online Localization in Dynamic Graphical Models) performs change localization in an online setting and is therefore naturally able to handle data in a streaming fashion without storing vast amounts of sensor data. This results in significant memory efficiency.

2. **Low Latency.** Our algorithm also enjoys provably low latency. In particular, we show that FOLk-DGM provably localizes changes and recovers the new graph in time that scales as $\widetilde{O}(\Delta^4)$, where Δ captures the magnitude of the change and $\widetilde{O}$ suppresses log factors. Δ is typically small if the change is not systemic in the underlying network. This is in contrast with the streaming algorithms of [4, 5] whose latency depends on the absolute value of the new edge weights.

3. **Computational Efficiency.** Our algorithm is also naturally highly computationally efficient. The online change detection and estimation algorithms proposed in [6, 7] can be naturally adapted to the change localization setting; however, they have a computational complexity of $\mathcal{O}(p^3)$. Our algorithm enjoys a computational complexity of $\widetilde{O}(p^2\Delta^4)$, where Δ and $\widetilde{O}$ are as above. So, when the change is abrupt and not systemic, our algorithm localizes the changes (and learns the new graph) computationally efficiently.

2. RELATED WORK

Existing work associated with dynamic graphical models roughly falls into two classes: one is based on time-varying graphical models, and the other is based on piece-wise constant graphical models. Researchers have developed algorithms for both offline and online data for both classes. In a

This work was supported by the National Science Foundation under award number CCF-2048223, and the Office of Naval Research (ONR) under award number N00014-21-1-2615.

time-varying graphical model, the graph topology gradually changes over time; the curious reader may read [8, 9, 10] for different algorithms developed for learning the structure of such time-varying graphical models. In this paper, we consider abrupt changes, as is typical in modeling anomalous or phase-change events in networked systems. To that end, [11] considered covariance selection models for multivariate time series where (at most one edge) changes in the dependence structure occur at random times, and [12] also considered learning time-indexed graphs for directed graph. Both these papers require the full dataset as input. In a different line of work, algorithms for directly learning the differential graph were proposed for undirected graphical models in the offline setting (see, e.g., [13, 14, 15]). [16, 17, 18] proposed methods for jointly learning multiple graphical models without considering the possibility of the existence of any change points. The work most closely related to ours is [6, 7]. The algorithm in [7] is based upon monitoring the conditional log-likelihood of all nodes in the network. The algorithm proposed in [6] is an approximate majorize-minimize algorithm for fitting piecewise constant high-dimensional models. Both of these works are focused on sequential change detection rather than localization. Further, it is important to note that the localization step in [7] is a batch method, whereas the method we propose is sequential. Finally, as previously mentioned, these methods are computationally impractical due to their high computational complexity.

3. NOTATION AND PRELIMINARIES

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, $\mathbf{X} \sim f_{\mathbf{X}} \triangleq \mathcal{N}(\mathbf{0}, \Sigma)$, where $\mathbf{0}$ is the p-dimensional vector of all zeros. Let $G = ([p], E)$ be a graph on the vertex set $[p] \triangleq \{1, 2, \ldots, p\}$ representing the coordinates of $\mathbf{X}$ with edge set $E \subset \binom{[p]}{2}$. The distribution of $\mathbf{X}$ is said to be Markov with respect to G if $(\Sigma^{-1})_{ij} = 0$ for all $\{i, j\} \notin E$. $K \triangleq \Sigma^{-1}$ is called the *precision matrix* of $\mathbf{X}$. Therefore, $\mathbf{X}$ is Markov with respect to G only if for every pair $i, j \in [p]$ such that $\{i, j\} \notin E$, we have that $K_{ij} = 0$. Recall that since $\mathbf{X}$ is Gaussian, $K_{ij} = 0$ if and only if X_i and X_j are conditionally independent given all the other coordinates of $\mathbf{X}$. We refer to the pair $(G, \mathcal{N}(\mathbf{0}, \Sigma))$ as a (Gaussian) graphical model; we invite the interested reader to read [19] for a more thorough exposition of graphical models. In what follows, we write $\mathbf{X}_{\setminus i}$ to denote the sub-vector of $\mathbf{X}$ with i-th coordinate removed.

The goal of the (static) structure learning problem is to recover the structure of the graph G given samples from the distribution $f_{\mathbf{X}}$. That is, we would like to construct an estimate $\widehat{E}$ of the edge set E from samples. In this paper, we are interested in the setting where the underlying graph is allowed to vary with time. That is, we will suppose that $G_t = ([p], E_t)$, $t \in [T]$ is a sequence of graphs with a fixed node set $[p]$ and T denotes the total number of independent samples one has access to from the system. Corresponding to each time $t \in [T]$, we suppose that we have a covariance matrix $\Sigma_t \in \mathbb{R}^{p\times p}$ and that we have access to a sample $\mathbf{X}_t \sim f_{\mathbf{X},t} \triangleq \mathcal{N}(\mathbf{0}, \Sigma_t)$. It is standard to adopt (see [7, 6]) a piecewise constant model for Σ_t (or equivalently K_t) to model abrupt changes in the dependency structure of the underlying graph. That is, we will suppose that there is a set of change points $\mathcal{T}_c \triangleq \{t_0^c, t_1^c, t_c^2, \ldots\} \subset [T]$ sorted in ascending order with $t_0^c = 1$, such that $\Sigma_t = \sum_{j=1}^{|\mathcal{T}_c|} \Sigma^{(j)} \mathbb{1}\left(t_j^c \le t < t_{j+1}^c\right)$. In the change localization problem, we suppose that we are given access to the set of changepoints $\mathcal{T}_c$, and our goal is to (rapidly) learn the changed covariance matrices $\Sigma(j)$ and thereby localize where the change occurred. Notice that this is different from the typical change detection problem, where the goal is to detect *when a change has occurred*. It is also different from the problem of model selection in time-varying graphical models, where the graph is assumed to vary smoothly. In order for our change localization procedure to work, we need to make the following assumptions.

Assumption 3.1. We will suppose that there exists a known $\Delta > 0$ such that

$$\max_j \left\| K^{(j)} - K^{(j-1)} \right\|_{1,\infty} \le \Delta.$$

That is the amount by which any one neighborhood in the graph changes is bounded by some known parameter Δ.

Assumption 3.2. We will suppose that there is a $t_* > 0$ such that $|t_{j+1}^c - t_j^c| \ge t_*$.

Assumption 3.2 is common in change detection and estimation literature [6, 7] and allows us not to miss any change events. This can of course, be relaxed if one is allowed to combine closely occurring changes.

Problem Setup. Let $\mathcal{T}_c$ be the set of given change points. For each changepoint $t_i^c \in \mathcal{T}_c$, let $\widehat{\Delta T}_i$ be the time elapsed since t_i^c before the algorithm correctly learns the changed graph (and hence localizes the change point); we will take $\widehat{\Delta T}_i = \infty$ if the algorithm fails to learn the i-th change point. For a change point localization algorithm, we define the *latency* to be $\max_i \widehat{\Delta T}_i$. The goal of (rapid) changepoint localization is to design an algorithm that, given any $\delta > 0$, terminates with low latency with probability at least $1 - \delta$.

4. THE RAPID CHANGE LOCALIZATION ALGORITHM

Before we present our algorithm, 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.

$$K_{\max} = \max_{(i,j)} |K_{ij}| \quad \text{and} \quad \sigma_{\max} = \max_{i} \text{Var}[\text{X}_\text{i}]$$

Our algorithm generalizes [4, 5] and operates in p parallel instances, one for each vertex (and is hence highly parallelizable). In order to understand the rationale behind our proposed algorithm, we start 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 [20], 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.

In an important paper [5], the authors make a similar observation for the Ising model and design a (static) graph learning algorithm that uses an adaptation of the Multiplicative Weights Update (MWU) framework [21] to learn the coefficients w_j for each vertex i, and therefore the neighborhood of the vertex i. The authors in [4] adapt this procedure to the setting of Gaussian graphical models. In this paper, we adapt this framework to perform rapid change localization in a dynamic setting. Our theory (see Section 5) shows that the algorithm proposed localizes the change with a lag that depends on the magnitude (ℓ_1 norm) of the change of a vector in the precision matrix K and only mildly (polylogarithmically) on the number of variables p. If one simply applies the algorithm from [4] to this setting, given that it is able to be run online, one can only establish a lag guarantee that depends on the ℓ_1 norm of the new weight vector – a quantity that could be significantly higher than the magnitude of the change vector. In fact, in our experiments (see 6), we show that this is not simply a weakness in theory and that it appears in practice as well.

Key Idea of the FOLk-DGM Algorithm. We present the key idea of FOLk-DGM by fixing an arbitrary change point in $\mathcal{T}_c$. Now, without loss of generality, we suppose that we are looking at the first change point t_1^c. Further, fix an arbitrary vertex $i \in [p]$. We suppose that $\mathbf{w}^*$ is the true weight vector corresponding to vertex i in the sense of (1) before the change occurs. For the sake of clarity, we will suppose that we know $\mathbf{w}^*$. Indeed, our method (and theory) works if one had to estimate $\mathbf{w}^*$ and condition on this estimate being accurate. We will suppose that $\mathbf{w}_1^*$ is the (unknown) weight-vector corresponding to vertex i post-change. *One can localize the change if one were to estimate* $\mathbf{w}_1^*$ *(using MWU based algorithm in [4]) accurately. Instead, our algorithm FOLk-DGM will estimate the difference vector* $d_1^* = \mathbf{w}^* - \mathbf{w}_1^*$ *accurately.* By our assumption, we know that $\|d_1^*\|_1 \leq \Delta$. In designing the algorithm, we will suppose that we know (an upper-bound on) Δ. Now, building on the algorithmic framework proposed in [4, 5], our algorithm operates in two phases for all the change points. In the first phase, the algorithm learns a *difference vector* for which the empirical loss is small and returns the post-change weight vector. Then, similarly, in the next phase, it learns the post-change weight vector for all the vertices in parallel. The outer-most loop in Algorithm 2 applies the two former steps for all the change points in $\mathcal{T}_c$.

Algorithm 1 ESTIMATING POST-CHANGE WEIGHT VECTOR BY LEARNING THE DIFFERENCE VECTOR

Input: Parameter tuple $(\delta, \sigma_{max}, \Delta)$; $N = N_{\text{tr}} + N_{\text{ts}}$ normalized samples; Normalizing parameter $C_n \triangleq \sqrt{2\log(\frac{2pN}{\delta})\sigma_{max}(2\Delta+1)}$; Learning rate β; A change point t_n^c; $\mathbf{w}_n^* \triangleq$ true weight vector before t_n^c .
Output: Post-change weight vector $\widehat{w}_{n+1}$.
Initialization: $\mathbf{d}^0 = \{\frac{1}{p}, \frac{1}{p}, \ldots, \frac{1}{p}\} \in \mathbb{R}^p$.
for $t = t_n^c + 1$ to $t_n^c + N_{\text{tr}}$ **do**
    Compute $\widetilde{\mathbf{d}}^t = \frac{\mathbf{d}^t}{\|\mathbf{d}^t\|_1}$ and $\Delta \times \widetilde{\mathbf{d}}^t$.
    $\boldsymbol{\ell}^t = (1/2)(\mathbf{1} + (\Delta\widetilde{\mathbf{d}}^t \cdot \widetilde{\mathbf{x}}_{-i}^t - (\widetilde{x_i}^t - \mathbf{w}_n^* \cdot \widetilde{\mathbf{x}}_{-i}^t))\widetilde{\mathbf{x}}_{-i}^t)$
    $\forall i \in [p]$, $d_i^t = d_i^{t-1}\beta^{l_i^t}$.
end for
Obtain N_{tr} candidate difference vectors.
Use N_{ts} samples for each candidate vector to compute empirical risk, and **return** $\widehat{\mathbf{d}}$ with the minimum empirical risk and $\widehat{\mathbf{w}}_{n+1} = \mathbf{w}_n^* + \widehat{\mathbf{d}}$

Algorithm 2 FOLK-DGM

Input: Set of change points $\mathcal{T}_c$; Same parameters as Algorithm 1.
for 1 to $|\mathcal{T}_c|$ **do**
    for $i = 1$ to p **do**
        Run Algorithm 1 to get the updated weight vectors for all vertex $i \in [p]$.
    end for
    Output post-change weight vector for all $i \in [p]$.
    $\forall$ pair (i, j), and their weight vectors w_i and w_j **declare** $\exists$ an edge (i, j), if $max(|w_i, |w_j|) \geq \frac{2\kappa}{3}$
end for

5. THEORETICAL GUARANTEES

We will now discuss the main theoretical result.

Theorem 5.1. *Let* $(G, \mathcal{N}(\mathbf{0}, \Sigma)$ *be a Gaussian graphical model on* p *vertices with parameters* $(\kappa, \sigma_{\max}, k_{\max}, \Delta)$ *as defined above. Let* $\mathcal{T}_c$ *be the set of given change points. Then,*

for any $\delta \in (0,1)$, FOLk-DGM terminates with a latency of at most

$$O\left(\frac{\Delta^4 \sigma_{\max}^2 (K_{\max})^2}{\kappa^4} \log^3\left(\frac{p \times |\mathcal{T}_c|}{\delta}\right)\right)$$

with probability at least $1-\delta$.

Proof Sketch. We will now provide a high-level proof sketch of the theorem. For a fixed change point and a fixed coordinate, let us denote $\widehat{\mathbf{d}}$ and $\mathbf{d}^*$ as the estimated and true difference vector, respectively. Further let us denote $\varepsilon(\widehat{\mathbf{d}}) \triangleq \mathbb{E}[(\widehat{\mathbf{d}} \cdot \mathbf{X} - \mathbf{d}^* \cdot \mathbf{X})^2]$ as the expected risk of the estimated difference vector. The proof shows that using N_{tr} samples FOLk-DGM guarantees that $\min_{n \in [N_{\text{tr}}]} \varepsilon(\widehat{\mathbf{d}}^n)$ is small. Next, this guarantee on minimal expected risk further guarantees that $\|\widehat{\mathbf{d}} - \mathbf{d}^*\|_\infty$ is upper bounded by a small amount. Finally, we will use the remaining samples to select a candidate difference vector with smallest empirical risk and show that expected risk of this candidate is close to the expected risk of the candidate difference vector with minimum expected risk. Thus $N_{\text{tr}} + N_{\text{ts}}$ samples will guarantee that $\widehat{\mathbf{d}}$ is a good estimate of $\mathbf{d}^*$ with high probability. ■

Remark 5.2. Theorem 5.1 shows that the amount of time one needs to wait before FOLk-DGM correctly localizes the detected change in the underlying graph scales as $\widetilde{\mathcal{O}}\left(\max_{t_j^c \in \mathcal{T}_c} \left\|K^{(j)} - K^{(j-1)}\right\|_{1,\infty}^4\right)$, where $\widetilde{O}$ suppresses logarithmic factors. Therefore, the slighter the change, the easier it is to localize. We suspect that the quadratic dependence on the magnitude of change is not fundamental and can be improved with a different analytical strategy; this is an interesting avenue for future work.

Remark 5.3. Our algorithm enjoys a low runtime $\mathcal{O}(Np)$, where N is the (random) latency before FOLk-DGM correctly identifies the changed weight vector. If we don't account for parallelization (which this algorithm is highly suited for), this implies a total runtime of $\mathcal{O}(Np^2)$. By Theorem 5.1, we know that with high probability this means that the runtime of FOLk-DGM scales as $\mathcal{O}(\Delta^4 p^2 \log^3 p)$. In comparison, our closest competitor has a runtime of $\mathcal{O}(p^3)$, which stems from the fact that they employ a computationally intensive procedure to do edge estimation.

6. EXPERIMENTS

We perform experiments on synthetic graphs to assess the validity of our theory. For a given number of vertices and the allowed number of non-zero entries per row, i.e. sparsity of the graph, we generate synthetic graphs, associated covariance matrices, and generate samples from these to set up our simulation. We report two results: (1) run-time comparison with GLASSO [22] and CLIME [23] as the learning algorithm (which is a reasonable adaptation of [7] to our setting), and (2) weight vector recovery comparison in ℓ_∞-norm with the multiplicative weights update (MWU) algorithm [4, 5]. Table 1 presents the run time comparison (with the TIC operation from MATLAB) for $p = 20, 40,$ and 60 vertices. We can see that FOLk-DGM performs better than both GLASSO and CLIME. We expect that as p increases the gap between the runtimes will increase as well. Next, in Fig. 1, for graphs with $p = 40$ and $p = 60$, we present the performance comparison between FOLk-DGM and the MWU algorithm in recovering the correct weight vector in the left subfigure and right subfigure, respectively. In order to accomplish this, we ran these algorithms for 15 independent trials following the introduction of the change. FOLk-DGM exhibits superior performance in recovering the accurate weight vectors once the change has been introduced with low latency.

Run time comparison in seconds			
Dimension	FOLk-DGM	GLASSO [22]	CLIME [23]
$p = 20$	70.82	233.51	245.3
$p = 40$	1203.5	4751.37	6406.48
$p = 60$	2800.31	8242.1	10328.35

Table 1: Run-time comparison for FOLk-DGM, GLASSO, and CLIME for graphs on 20, 40, and 60 vertices.

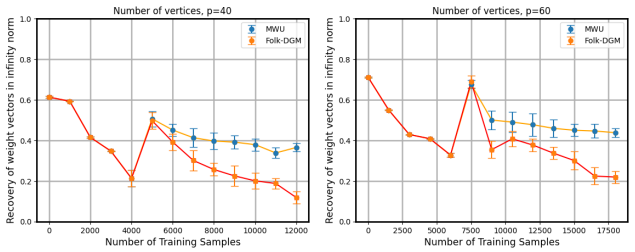

Fig. 1: Performance comparison of recovered weight vectors in ℓ_∞ norm between MWU algorithm [4] and FOLk-DGM. We see that for both graphs after the change occurs (4000 (resp. 6000) samples for $p = 40$ (resp. $p = 60$)), FOLk-DGM was able to recover the correct weight vector quickly.

7. CONCLUSION

We consider the problem of change localization in graphical models whose underlying graph undergoes abrupt changes. We propose a novel algorithm for this problem, dubbed FOLk-DGM, that is online and hence memory efficient. Further, the algorithm we propose has provably low latency for change localization and has low computational complexity compared to natural competitors. We supplement our theory with experimental results that corroborate our findings. In our future work we will integrate sequential online hypothesis testing methods for change detection.

8. REFERENCES

[1] Marloes Maathuis, Mathias Drton, Steffen Lauritzen, and Martin Wainwright, *Handbook of graphical models*, CRC Press, 2018.

[2] Mathias Drton and Marloes H Maathuis, "Structure learning in graphical modeling," *Annual Review of Statistics and Its Application*, vol. 4, pp. 365–393, 2017.

[3] Christian G Daughton, "Wastewater surveillance for population-wide covid-19: The present and future," *Science of the Total Environment*, vol. 736, pp. 139631, 2020.

[4] Anamay Chaturvedi and Jonathan Scarlett, "Learning gaussian graphical models via multiplicative weights," *arXiv preprint arXiv:2002.08663*, 2020.

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

[6] Leland Bybee and Yves Atchadé, "Change-point computation for large graphical models: A scalable algorithm for gaussian graphical models with change-points," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 440–477, 2018.

[7] Hossein Keshavarz, George Michailidis, and Yves Atchadé, "Sequential change-point detection in high-dimensional gaussian graphical models," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 3125–3181, 2020.

[8] Jilei Yang and Jie Peng, "Estimating time-varying graphical models," *Journal of Computational and Graphical Statistics*, vol. 29, no. 1, pp. 191–202, 2020.

[9] Erich Kummerfeld and David Danks, "Tracking time-varying graphical structure," *Advances in neural information processing systems*, vol. 26, 2013.

[10] Mladen Kolar, Le Song, Amr Ahmed, and Eric P Xing, "Estimating time-varying networks," *The Annals of Applied Statistics*, pp. 94–123, 2010.

[11] Makram Talih and Nicolas Hengartner, "Structural learning with time-varying components: tracking the cross-section of financial time series," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 3, pp. 321–341, 2005.

[12] Michael Siracusa and John Fisher III, "Tractable bayesian inference of time-series dependence structure," in *Artificial Intelligence and Statistics*. PMLR, 2009, pp. 528–535.

[13] Song Liu, Kenji Fukumizu, and Taiji Suzuki, "Learning sparse structural changes in high-dimensional markov networks," *Behaviormetrika*, vol. 44, no. 1, pp. 265–286, 2017.

[14] Song Liu, John A Quinn, Michael U Gutmann, Taiji Suzuki, and Masashi Sugiyama, "Direct learning of sparse changes in markov networks by density ratio estimation," *Neural computation*, vol. 26, no. 6, pp. 1169–1197, 2014.

[15] Sihai Dave Zhao, T Tony Cai, and Hongzhe Li, "Direct estimation of differential networks," *Biometrika*, vol. 101, no. 2, pp. 253–268, 2014.

[16] Jian Guo, Elizaveta Levina, George Michailidis, and Ji Zhu, "Joint estimation of multiple graphical models," *Biometrika*, vol. 98, no. 1, pp. 1–15, 2011.

[17] Jing Ma and George Michailidis, "Joint structural estimation of multiple graphical models," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 5777–5824, 2016.

[18] Karthik Mohan, Palma London, Maryam Fazel, Daniela Witten, and Su-In Lee, "Node-based learning of multiple gaussian graphical models," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 445–488, 2014.

[19] Steffen L Lauritzen, *Graphical models*, vol. 17, Clarendon Press, 1996.

[20] John A Gubner, *Probability and random processes for electrical and computer engineers*, Cambridge University Press, 2006.

[21] Sanjeev Arora, Elad Hazan, and Satyen Kale, "The multiplicative weights update method: a meta-algorithm and applications," *Theory of computing*, vol. 8, no. 1, pp. 121–164, 2012.

[22] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[23] Tony Cai, Weidong Liu, and Xi 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.