ONLINE TIME-VARYING TOPOLOGY IDENTIFICATION VIA PREDICTION-CORRECTION ALGORITHMS

Alberto Natali, Mario Coutino, Elvin Isufi and Geert Leus

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, Delft, The Netherlands

ABSTRACT

Signal processing and machine learning algorithms for data supported over graphs, require the knowledge of the graph topology. Unless this information is given by the physics of the problem (e.g., water supply networks, power grids), the topology has to be learned from data. Topology identification is a challenging task, as the problem is often ill-posed, and becomes even harder when the graph structure is time-varying. In this paper, we address the problem of dynamic topology identification by building on recent results from time-varying optimization, devising a general-purpose online algorithm operating in non-stationary environments. Because of its iteration-constrained nature, the proposed approach exhibits an intrinsic temporal-regularization of the graph topology without explicitly enforcing it. As a case-study, we specialize our method to the Gaussian graphical model (GGM) problem and corroborate its performance.

Index Terms— dynamic topology identification, online algorithm, graphical models, graph learning, time-varying optimization.

1. INTRODUCTION

The knowledge of the graph topology is a *requirement* for processing tools operating on data residing on top of networks, often conceptualized as graph signals [1, 2]. Unless the topology is known beforehand like in water supply networks or power grids [3], it has to be inferred from the available networked data. This challenging task is known in literature as *network topology inference* or *graph learning* [4, 5], and it becomes even harder when the graph topology is dynamic, i.e., it changes over time, like in brain functional connectivity [6] and biological networks [7].

When the network is dynamic, the dependence of the graph structure is reflected on the underlying time-varying distribution of the observable data. Thus, the task of dynamic network topology identification is to use the stream of data to infer the evolution of the network structure in an online fashion. This online inference task allows a network invigilator to promptly detect anomalies, e.g., uncommon financial transactions among users, or to perform effective decision making, e.g., load redistribution in smart grids.

Previous works in this area, e.g. [8, 9], learn a sequence of graphs by enforcing a prior (smoothness or sparsity) on the edges of consecutive graphs. Adopting a similar approach, the work in [10] extends the graphical Lasso [11] to estimate the dynamic topology using the alternating direction method of multipliers (ADMM) [12].

In addition to these works, the inference of causal relationships in the network structure, i.e., directed edges, has been considered in [13]. For a complete review of causal dynamic topology inference, the reader is referred to [14].

The aspect that brings all previous approaches together is the following same *processing chain*: i) data-collection; ii) data-splitting (windowing); iii) batch-processing with structural constraints between consecutive graph topologies. This processing chain fails to address the *online (data-streaming) setting*, recently investigated in [15], which assumes graph data generated by a heat diffusion process, and by [16], which assumes data to be graph stationarity, i.e., the covariance matrix of the data and the matrix representation of the network commute. Differently from these works, our framework is more general as it does not require such assumptions.

To address the dynamic network topology inference problem, we develop an adaptive algorithm based on the time-varying convex optimization framework [17, 18]. This framework operates on-the-fly, thus adapting to non-stationary dynamics, and runs without requiring the entire sequence of observations. The devised adaptive algorithm does not assume knowledge of the time instants at which the topology changes and it implicitly applies a temporal regularization due to its early-stopping behavior, i.e., limited iteration budget. We demonstrate the validity of our approach by focusing on the well-known Gaussian graphical model (GGM) problem. Although the GGM problem does not consider the causal scenario, i.e., non-symmetric network topology, we stress that the proposed dynamic network topology tracking framework can be applied to this scenario as well.

2. PRELIMINARIES

2.1. Graphs and Signals over Graphs

Consider a scenario in which the data of interest reside in a non-Euclidean domain described by the undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{S}\}$, where $\mathcal{V} = \{1, \ldots, N\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are the vertex and edge set, respectively; and $\mathbf{S}$ is an $N \times N$ symmetric matrix that represents the graph structure, which entries $[\mathbf{S}]_{i,j}$ are nonzero only if $(j, i) \in \mathcal{E}$, for $i \neq j$. Matrix $\mathbf{S}$ is typically called the graph shift operator (GSO) [2], and examples include the (weighted) adjacency matrix $\mathbf{W}$ [2] and the graph Laplacian $\mathbf{L}$ [1]. By associating to each node $i \in \mathcal{V}$ a scalar value x_i, we can define $\mathbf{x} \in \mathbb{R}^N$ as a *graph signal* mapping the node set to the set of real vectors.

2.2. Topology Identification

Static. When the knowledge of the topology of the network is not available, we encounter the problem of learning the network structure from data, a problem known as *topology identification or graph learning*. Formally, consider the matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ that

stacks T graph signals of dimension N arising from an unknown graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{S})$. The goal is then to infer the *latent* underlying topology of the graph by these data, which amounts to estimating the GSO. In this work, we focus particularly on the Gaussian graphical model (GGM) problem, which we recall, to pave the way for the illustration of our (dynamic) network inference framework.

Gaussian Graphical Model. Assume each graph signal $\mathbf{x}$ to be drawn from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{S}_{++}^N$. By denoting with $\mathbf{S} = \boldsymbol{\Sigma}^{-1}$ the *precision* matrix, *graph learning in a GGM amounts to precision matrix estimation*. This is because a missing edge in $\mathcal{G}$ corresponds to a zero entry in the precision matrix, and consequently to the conditional independence of the related entities of the graph. Formally, the maximum likelihood estimation (MLE) for $\mathbf{S}$ reads as:

$$\underset{\mathbf{S} \in \mathcal{S}}{\text{minimize}} - \log \det(\mathbf{S}) + \text{tr}(\mathbf{S}\hat{\boldsymbol{\Sigma}}), \qquad (1)$$

where $\hat{\boldsymbol{\Sigma}}$ is the sample covariance matrix and $\mathcal{S} = \mathbb{S}_{++}^N$.

Dynamic. Our focus is, however, on scenarios where the graph topology changes over time, e.g., social interaction or brain connections. These changing interactions are represented by a time-varying topology, which we express by the sequence of graphs $\{\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t, \mathbf{S}_t)\}_{t=1}^\infty$ with associated GSOs $\{\mathbf{S}_t\}_{t=1}^\infty$. This formulation can be seen as the sampling of a time-continuous dynamic topology using a sampling period of h.

We consider time-varying graph signals $\{\mathbf{x}_t\}_{t=1}^\infty$ arising from an unknown time-varying graph $\{\mathcal{G}_t\}_{t=1}^\infty$. Akin to the static case, now *the goal is to infer the underlying time-varying topology of the graph from $\{\mathbf{x}_t\}_{t=1}^\infty$*, which is tantamount to the estimation of the matrix sequence $\{\mathbf{S}_t\}_{t=1}^\infty$.

Using the above formalism, most of the current works have relied on batch approaches to tackle the dynamic topology inference setting. That is, after all possible data have been collected, pre-processed, and then split into (possibly overlapping) temporally-contiguous windows, each window is associated with a graph topology. This topology has been further constrained to be *close* to those of temporally-adjacent windows, by solving a regularized version of (1), see, e.g., [10]. These approaches $i)$ require all windows at once to form the dynamic topology estimate and $ii)$ need to select the time instances where to split the data into windows, i.e., the approximate time instances where the topology changes. These form a limitation because in many scenarios the memory storage is an issue, batch processing is computationally expensive, or a control agent may need to take real-time decisions.

To alleviate these issues, we devise an on-the-fly adaptive algorithm, using time-varying optimization [18], that updates the solution as samples come into the system, thus avoiding assumptions on the time instants at which the topology changes. Because of its iteration-budget constraints, the adaptive method exhibits intrinsic regularization properties without the need of explicit regularizers. This is advantageous since we do not need to rely on a specific regularization assumption such as smoothness or sparsity, which may not hold in practice.

In Section 3, we introduce the time-varying optimization framework for the dynamic graph learning problem and, in Section 4, we specialize it to the GGM problem.

3. TIME-VARYING TOPOLOGY IDENTIFICATION

Conventional optimization methods can no longer be taken for granted when dealing with dynamic systems whose internal parameters are continuously evolving and for which fast decision-making

tasks are required. Fortunately, advances have been made in the field of time-varying optimization [17, 18]. We leverage these ideas and propose a framework that given a time-varying cost function $f(\mathbf{S}, t)$ maintains an estimate solution $\hat{\mathbf{S}}_t$, at time instant t, of the time-varying optimization problem $\min_{\mathbf{S} \in \mathcal{S}} f(\mathbf{S}, t)$ for some feasible set $\mathcal{S}$, and then *predict* how this solution evolves in the next time step $t + 1$. This prediction is then *corrected* after a new datum is available at time $t + 1$.

Formally, we aim to solve the sequence of time-invariant problems of the form:

$$\mathbf{S}_t^* := \underset{\mathbf{S} \in \mathcal{S}}{\text{argmin}} f(\mathbf{S}; t), \quad t \in \mathbb{N} \qquad (2)$$

where $\mathcal{S}$ is a convex set defining the feasible set of GSOs[1]. As the process is sampled at given intervals and (2) is typically solved by iterative methods, in most instances, the sampling period will only allow for few iterations of the selected solver, (possibly) leading to a suboptimal solution. Due to its practical relevance, this is the setting of our interest.

It can be shown that solving problem (2), at time t, is equivalent to solving the generalized equation:

$$\mathbf{R}(t) := \nabla_{\mathbf{S}} f(\mathbf{S}_t^*; t) + \mathcal{N}_{\mathcal{S}}(\mathbf{S}_t^*) \ni \mathbf{0}, \qquad (3)$$

where $\nabla_{\mathbf{S}} f(\cdot) \in \mathbb{R}^N$ is the gradient of f with respect to $\mathbf{S}$ and $\mathcal{N}_{\mathcal{S}}(\cdot) : \mathbb{R}^{N \times N} \to \mathbb{R}^{N \times N}$ is the normal cone operator (the subdifferential of the indicator function).

3.1. Prediction

Suppose now that $\hat{\mathbf{S}}_t$ is an approximate solution for (3) at time t, i.e. a point estimate for which the residual $\mathbf{R}(t) \approx 0$. The prediction step seeks an approximate optimizer for (2) at $t + 1$, with only the information available up to t.

Denoting by $\hat{\mathbf{S}}_{t+1|t}$ the output of the prediction step, the goal is then to solve:

$$\nabla_{\mathbf{S}} f\left(\hat{\mathbf{S}}_{t+1|t}; t + 1\right) + \mathcal{N}_S\left(\hat{\mathbf{S}}_{t+1|t}; t + 1\right) \ni \mathbf{0} \qquad (4)$$

which is impossible at time t. However, we can perturb the above generalized equation, by means of backward Taylor expansion, and obtain the generalized system:

$$\nabla_{\mathbf{S}} f(\hat{\mathbf{S}}_t; t) + \nabla_{\mathbf{SS}} f(\hat{\mathbf{S}}_t; t) : (\hat{\mathbf{S}}_{t+1|t} - \hat{\mathbf{S}}_t)$$
$$+ \nabla_{t\mathbf{S}} f(\hat{\mathbf{S}}_t; t)h + \mathcal{N}_S\left(\hat{\mathbf{S}}_{t+1|t}\right) \ni \mathbf{0}, \qquad (5)$$

where $\nabla_{\mathbf{SS}} f(\cdot) \in \mathbb{R}^{N \times N \times N \times N}$ is the Hessian tensor of f with respect to $\mathbf{S}$, $\nabla_{t\mathbf{S}} f(\cdot) \in \mathbb{R}^N$ is the partial derivative of the gradient of f w.r.t. the time t, and : denotes the (scalar) inner product between arguments. We can notice that the formulation in (5) is equivalent to the constrained optimization problem (6), where the normal cone operator leads to the definition of the constraint set over which the computation is performed. Intuitively, setting to zero the gradient of the cost function in (6), and taking into account its feasible set, leads to (5).

Although (6) is a constrained quadratic optimization problem, and its solution can be efficiently found, its exact solution might incur high computational costs. Therefore, when solving (6) we approximate its solution by means of a few projected gradient (PG)

---

[1]Note that the temporal variability of the function is due to the graph signals acquired up to the time instant of interest.

Algorithm 1 Time-Varying Topology Identification

Require: Feasible $\hat{\mathbf{S}}_0$, $f(\mathbf{S}; t_0)$, P, C
1: **for** $t = 0, 1, \ldots$ **do**
2:     // *Prediction*
3:     Initialize $\hat{\mathbf{S}}^0 = \hat{\mathbf{S}}_t$
4:     **for** $p = 0, 1, \ldots, P - 1$ **do**
       Predict $\hat{\mathbf{S}}^{p+1}$ with (7)
5:     **end for**
       Set the predicted variable $\hat{\mathbf{S}}_{t+1|t} = \hat{\mathbf{S}}^P$.
6:     *time $t + 1$: new data arrive*
7:     // *Correction*
8:     Initialize the corrected variable $\hat{\mathbf{S}}^0 = \hat{\mathbf{S}}_{t+1|t}$
9:     **for** $c = 0, 1, \ldots, C - 1$ **do**
       Predict $\hat{\mathbf{S}}^{c+1}$ with (9)
10:    **end for**
       Set the corrected variable $\hat{\mathbf{S}}_{t+1} = \hat{\mathbf{S}}^C$
11: **end for**

---

descent steps. So, letting $\hat{\mathbf{S}}^0$ be a dummy variable initialized as $\hat{\mathbf{S}}^0 = \hat{\mathbf{S}}_t$, we perform the following steps:

$$\hat{\mathbf{S}}^{p+1} = \mathbb{P}_{\mathcal{S}}[\hat{\mathbf{S}}^p - \alpha(\nabla_{\mathbf{S}} f(\hat{\mathbf{S}}_t; t) + \nabla_{\mathbf{SS}} f(\hat{\mathbf{S}}_t; t) : (\hat{\mathbf{S}}^p - \hat{\mathbf{S}}_t)$$
$$+ \nabla_{t\mathbf{S}} f(\hat{\mathbf{S}}_t; t)h)] \tag{7}$$

for $p = 0, 1, \ldots, P - 1$, where P is a predefined number of PG steps, $\alpha > 0$ is the stepsize, and $\mathbb{P}_{\mathcal{S}}$ it the projection operator over the convex set $\mathcal{S}$. Once the P steps are performed, the prediction $\hat{\mathbf{S}}_{t+1|t}$ is set to:

$$\hat{\mathbf{S}}_{t+1|t} = \hat{\mathbf{S}}^P \tag{8}$$

which is expected to approximate the optimal solution $\mathbf{S}^*_{t+1}$ at time $t + 1$.

3.2. Correction

When new graph signals are acquired, i.e. the new cost function $f(\cdot; t+1)$ becomes available, the correction step refines the estimate of the optimal solution $\hat{\mathbf{S}}_{t+1|t}$. To do so, we also perform a set of PG steps. In particular, by setting to $\hat{\mathbf{S}}^0 = \hat{\mathbf{S}}_{t+1|t}$, we consider the following PG steps:

$$\hat{\mathbf{S}}^{c+1} = \mathbb{P}_{\mathcal{S}} \left[\hat{\mathbf{S}}^c - \beta \nabla_{\mathbf{S}} f(\hat{\mathbf{S}}^c; t + 1) \right], \tag{9}$$

for $c = 0, 1, \ldots, C - 1$. The refined estimate of the optimal solution $\mathbf{S}^*_{t+1}$ is then set as $\hat{\mathbf{S}}_{t+1} = \hat{\mathbf{S}}^C$. Notice that we can also use a (quasi)-Newton update as an alternative to the PG step in (9) if we can afford the related computational cost.

    The general time-varying optimization framework, adapted to our case, is summarized in Algorithm 1.

4. TIME-VARYING GAUSSIAN GRAPHICAL MODELS

The previous problem formulation has been developed without a particular specification of the cost function $f(\cdot; t)$. This enables us to extend a large class of "static" topology identification algorithms. That is, given a cost function, the framework only requires access to its first- and second-oder derivatives. In the following, we specialize this framework to an online GGM setting.

    Denote with $\mathcal{X}_t = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ the set of graph signals available up to time t. Let also $\hat{\mathbf{\Sigma}}_t$ be the empirical covariance matrix

computed with the signals up to time t in a weighted moving average fashion, i.e., $\hat{\mathbf{\Sigma}}_t = \gamma \hat{\mathbf{\Sigma}}_{t-1} + (1 - \gamma)\mathbf{x}_t\mathbf{x}_t^\top$, where $\gamma \in (0, 1)$ plays the role of forgetting factor, required in non-stationary environments. Then the GGM problem, adopted to a time-varying setting, reads as:

$$\mathbf{S}^*_t := \underset{\mathbf{S} \in \mathcal{S}}{\operatorname{argmin}} \{ f(\mathbf{S}; t) := -\log \det(\mathbf{S}) + \operatorname{tr}(\mathbf{S}\hat{\mathbf{\Sigma}}_t) \}, \tag{10}$$

where $\mathbf{S}$ is the precision matrix, and $\mathcal{S}$ defines the (convex) constraint set of valid solutions for the problem, in this case coinciding with the set of positive definite matrices. For other commonly-used sets for GSOs and/or graph matrices see, e.g., [19].

    While problem (10) can be solved at each time instant from scratch, this induces a large computational overhead. Thus, an approach which *tracks* the optimizer $\mathbf{S}^*_t$ based on previous estimates, and the available graph signals, is highly attractive.

4.1. Implementation

To apply the presented time-varying topology identification method to the GGM problem, we need to compute the expressions of the gradients $\nabla_{\mathbf{S}} f(\mathbf{S}; t)$, $\nabla_{t\mathbf{S}} f(\mathbf{S}; t)$, and the Hessian $\nabla_{\mathbf{SS}} f(\mathbf{S}; t)$ for the prediction step. Since $\mathbf{S}$ is symmetric, we can reduce the number of independent variables from N^2 to $N(N + 1)/2$. The latter is enforced by means of representing the matrix $\mathbf{S}$ with its half-vectorization form, i.e., $\mathbf{s} = \operatorname{vech}(\mathbf{S}) \in \mathbb{R}^{N(N+1)/2}$. We further introduce the elimination matrix $\mathbf{E} \in \mathbb{R}^{[N(N+1)/2] \times N}$ and the duplication matrix $\mathbf{D} \in \mathbb{R}^{N \times [N(N+1)/2]}$ which selects the unique entries of $\mathbf{S}$, i.e., $\mathbf{E}\operatorname{vec}(\mathbf{S}) = \mathbf{s}$, and duplicates the entries of $\mathbf{s}$, i.e, $\mathbf{D}\mathbf{s} = \operatorname{vec}(\mathbf{S})$, respectively.

    The gradient and the Hessian of the function f in the half-vectorization space can be derived as:

$$\mathbf{h}(\mathbf{S}; t) = \frac{\partial f(\mathbf{S}; t)}{\partial \mathbf{s}} = \mathbf{D}^\top \operatorname{vec}(\hat{\mathbf{\Sigma}}_t - \mathbf{S}^{-1})$$

$$\mathbf{H}(\mathbf{S}) = \frac{\partial^2 f(\mathbf{S}; t)}{\partial^2 \mathbf{s}} = \mathbf{D}^\top (\mathbf{S} \otimes \mathbf{S})^{-1} \mathbf{D},$$

respectively. Because we are in a discrete-time setting, the time derivative of the gradient is given by the partial mixed-order derivative:

$$\mathbf{g}_t = \mathbf{D}^\top \operatorname{vec}(\hat{\mathbf{\Sigma}}_t - \hat{\mathbf{\Sigma}}_{t-1}).$$

Now, by defining $\hat{\mathbf{s}}_t := \operatorname{vech}(\hat{\mathbf{S}}_t) \in \mathbb{R}^{N(N+1)/2}$, $\mathbf{h}_t := \mathbf{h}(\hat{\mathbf{S}}_t; t)$ and $\mathbf{H}_t := \mathbf{H}(\hat{\mathbf{S}}_t)$, we have that:

- **Prediction:** with $\hat{\mathbf{s}}^0$ a dummy variable initialized as $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_t$:

$$\hat{\mathbf{s}}^{p+1} = \mathbb{P}_{\mathcal{S}} [\hat{\mathbf{s}}^p - 2\alpha (\mathbf{h}_t + \mathbf{H}_t (\hat{\mathbf{s}}^p - \hat{\mathbf{s}}_t) + h\mathbf{g}_t)] \tag{11}$$

  for $p = 0, 1, \ldots, P - 1$. Then, the prediction $\hat{\mathbf{s}}_{t+1|t}$ is set to $\hat{\mathbf{s}}_{t+1|t} = \hat{\mathbf{s}}^P$.

- **Correction**: by setting $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_{t+1|t}$:

$$\hat{\mathbf{s}}^{c+1} = \mathbb{P}_{\mathcal{S}} [\hat{\mathbf{s}}^c - \beta \mathbf{h}_{t+1}], \tag{12}$$

  for $c = 0, 1, \ldots, C - 1$ and $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}^C$.

$$\hat{\mathbf{S}}_{t+1|t} = \underset{\mathbf{S} \in \mathcal{S}}{\operatorname{argmin}} \left\{ \frac{1}{2}\mathbf{S} : \nabla_{\mathbf{SS}} f\left(\hat{\mathbf{S}}_t; t\right) : \mathbf{S} + \left[\nabla_{\mathbf{S}} f\left(\hat{\mathbf{S}}_t; t\right) + h\nabla_{t\mathbf{S}} f\left(\hat{\mathbf{S}}_t; t\right) - \nabla_{\mathbf{SS}} f\left(\hat{\mathbf{S}}_t; t\right) : \hat{\mathbf{S}}_t \right] : \mathbf{S} \right\} \tag{6}$$

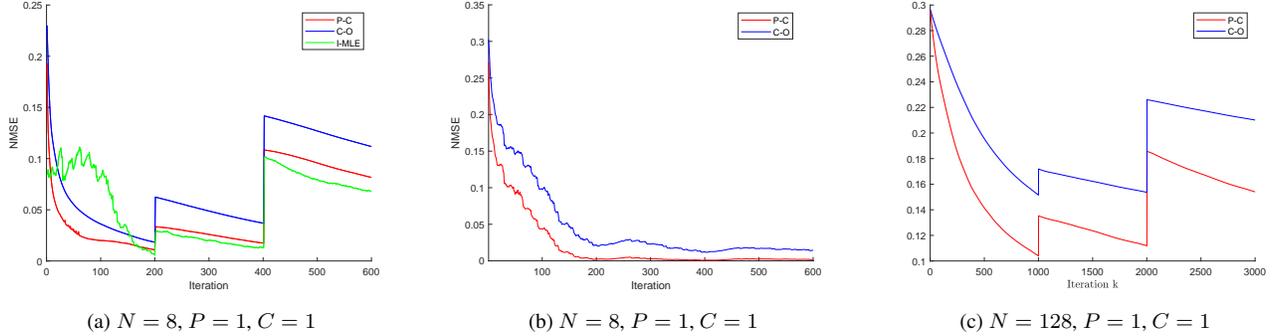(a) $N = 8, P = 1, C = 1$ (b) $N = 8, P = 1, C = 1$ (c) $N = 128, P = 1, C = 1$

Fig. 1: (a) NMSE of the (P-C), (C-O), (I-MLE) solutions w.r.t. the (B-MLE) solution; (b) NMSE of the (P-C), (C-O) solutions w.r.t. the (I-MLE) solution; (c) NMSE of the (P-C), (C-O) solutions w.r.t. the true underlying graph topology.

5. NUMERICAL RESULTS

We consider a simple graph of $N = 8$ nodes undergoing two triggering events. For the graph and data generation, we adopt the following procedure: first, we generate a positive definite matrix representing the precision matrix $\mathbf{S}_0$ at time $t = 0$, i.e., the starting graph topology, that remains constant for the first 200 time instants. Then, we perturb the graph topology obtaining a new GSO, that, again, remains constant for 200 time indices. We repeat this again, obtaining a third GSO. Each time, we perturb the graph by randomly selecting one node, and increasing the weight of its adjacent edges with 20%. Finally, we draw $T = 600$ graph signals following $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_t)$, where $\boldsymbol{\Sigma}_t = \mathbf{S}_t^{-1}$, for $t = \{1, \ldots, T\}$.

We assume that each graph signal arrives in an online fashion one at the time. In this way, our cost function changes at every new sample that arrives. However, the distribution of the data, reflected in the precision matrix $\mathbf{S}_t$, changes only every 200 samples.

Analysis. We are interested in tracking: *i)* the optimal batch MLE (**B-MLE**) solution that solves the standard GGM (1) for the three stationary intervals of 200 graph signals (should be close to the generative precision matrix $\mathbf{S}_t$); and *ii)* the instantaneous MLE (**I-MLE**) solution that solves (10) and makes use of the streaming graph signals up to a particular time instant. We denote the B-MLE and I-MLE solutions as $\mathbf{S}_B$ and $\mathbf{S}_I$, respectively. Hence, we have 3 B-MLE and 600 I-MLE solutions, equal to the number of stationary intervals and time instants, respectively. If, at the end of each stationary period, the empirical covariance matrices coincide, then $\mathbf{S}_I$ should coincide with $\mathbf{S}_B$. However, since for the I-MLE solution we use a forgetting factor, they will not.

To assess the validity of our approach, we compare our prediction-correction (**P-C**) solution, and the correction-only (**C-O**) solution with respect to : *i)* the I-MLE solution $\mathbf{S}_I$ for each time instant $t \in \{1, \ldots, T\}$; *ii)* the B-MLE solution $\mathbf{S}_B$ for the three stationary intervals. We also compare how $\mathbf{S}_I$ deviates from $\mathbf{S}_B$ at each time instant. We evaluate the performance of the algorithm by means of the normalized MSE (NMSE), computed as:

$$\text{NMSE} = \frac{\|\hat{\mathbf{S}} - \mathbf{S}_{\text{MLE}}\|_F^2}{\|\mathbf{S}_{\text{MLE}}\|_F^2} \tag{13}$$

where $\hat{\mathbf{S}}$ is either the P-C or the C-O solution (or $\mathbf{S}_I$ when comparing to $\mathbf{S}_B$), and $\mathbf{S}_{\text{MLE}}$ is either $\mathbf{S}_I$ or $\mathbf{S}_B$.

Results. Fig. 1a shows the NMSE of the P-C solution (for $P = 1$, $C = 1$, $\alpha = \beta = 1e{-}3$), the C-O solution ($C = 1, \alpha = \beta = 1e{-}3$) and I-MLE solution $\mathbf{S}_I$ with respect to the optimal B-MLE $\mathbf{S}_B$, for a forgetting factor of $\gamma = 0.97$. The triggering effect is visible for

$t = \{200, 400\}$, where the NMSE has a sharp increase. All the three solutions show a convergence behavior to the B-MLE $\mathbf{S}_B$ and we especially observe how the prediction step improves the performances w.r.t. a correction-only algorithm. This is also visible in Fig. 1b, that shows the NMSE of the P-C and C-O solution w.r.t. the I-MLE solution for each time instant. Here we see also how at $t = \{200, 400\}$, the iterative solutions move apart from the I-MLE solution, and then follow it again, probably due to the forgetting factor. We can conclude, based on these two figures, that the algorithm naturally enforces a similarity of the solutions at each iteration, differently from the MLE solution. In other words, the algorithm adds a regularization to the problem without any regularizer in the cost function. This is achieved with an *early stopping* of the descent steps, governed by the parameters P and C.

A more challenging scenario is considered in Fig. 1c, where the algorithm runs on a graph of $N = 128$ nodes, and follows the same perturbation model as the previous one, yet this time with a 50% dilation. Due to the high-computational cost involved for the MLE solutions, we show the NMSE of the iterative solutions with respect to the true underlying matrix $\mathbf{S}_t$, where we can still see the trajectory tracking of the algorithm.

Experimentally, we also noticed (not shown) how we can tune the behavior of the algorithm between following the I-MLE solution (increasing C) or the B-MLE solution (using a low P and C). We also observed how, for a low value of C, the value of P has a an impact on the performance, especially w.r.t. the B-MLE solution.

6. CONCLUSION

We proposed an online algorithm operating in non-stationary environments to learn a dynamic graph topology from observed data. The proposed approach, built on the prediction-correction framework, does not require the knowledge of the time instants in which the topology changes. It further implicitly regularizes the problem due to the early-stopping behavior in the iterative process, enforcing similarity between solution iterates. Because the algorithmic formulation is developed without explicitly specifying the cost function, it can be used to extend a variety of (static) topology identification algorithms. We detailed its application to the Gaussian graphical model (GGM) problem, where numerical results show the tracking capabilities of the algorithm with respect to the optimal solution(s). As future works, we will validate our algorithm for different perturbation models and different cost functions, as well as on real data sets. Building on our recent work [20], we also envision to extend it to cases where the sparsity pattern of the graph is assumed to be known, yet not the importance of the edge weights.

7. REFERENCES

[1] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[2] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, April 2013.

[3] Mark Newman, *Networks*, Oxford university press, 2018.

[4] Gonzalo Mateos, Santiago Segarra, Antonio G Marques, and Alejandro Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

[5] Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 44–63, 2019.

[6] Maria Giulia Preti, Thomas AW Bolton, and Dimitri Van De Ville, "The dynamic functional connectome: State-of-the-art and perspectives," *Neuroimage*, vol. 160, pp. 41–54, 2017.

[7] Yongsoo Kim, Seungmin Han, Seungjin Choi, and Daehee Hwang, "Inference of dynamic networks using time-course data," *Briefings in bioinformatics*, vol. 15, no. 2, pp. 212–228, 2014.

[8] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2826–2830.

[9] Koki Yamada, Yuichi Tanaka, and Antonio Ortega, "Time-varying graph learning with constraints on graph temporal variation," *arXiv preprint arXiv:2001.03346*, 2020.

[10] David Hallac, Youngsuk Park, Stephen Boyd, and Jure Leskovec, "Network inference via the time-varying graphical lasso," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 205–213.

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

[12] Stephen Boyd, Neal Parikh, and Eric Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Now Publishers Inc, 2011.

[13] B. Baingana and G. B. Giannakis, "Tracking switched dynamic network topologies from information cascades," *IEEE Transactions on Signal Processing*, vol. 65, no. 4, pp. 985–997, 2017.

[14] Georgios B Giannakis, Yanning Shen, and Georgios Vasileios Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 787–807, 2018.

[15] S. Vlaski, H. P. Maretić, R. Nassif, P. Frossard, and A. H. Sayed, "Online graph learning from sequential data," in *2018 IEEE Data Science Workshop (DSW)*, 2018, pp. 190–194.

[16] Rasoul Shafipour and Gonzalo Mateos, "Online topology inference from streaming stationary graph signals with partial connectivity information," *Algorithms*, vol. 13, no. 9, pp. 228, 2020.

[17] Andrea Simonetto, Aryan Mokhtari, Alec Koppel, Geert Leus, and Alejandro Ribeiro, "A class of prediction-correction methods for time-varying convex optimization," *IEEE Transactions on Signal Processing*, vol. 64, no. 17, pp. 4576–4591, 2016.

[18] Andrea Simonetto, Emiliano Dall'Anese, Santiago Paternain, Geert Leus, and Georgios B Giannakis, "Time-varying convex optimization: Time-structured algorithms and applications," *Proceedings of the IEEE*, 2020.

[19] Santiago Segarra, Antonio G Marques, Gonzalo Mateos, and Alejandro Ribeiro, "Network topology inference from spectral templates," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 467–483, 2017.

[20] Alberto Natali, Mario Coutino, and Geert Leus, "Topology-aware joint graph filter and edge weight identification for network processes," *arXiv preprint arXiv:2007.03266*, 2020.