Learning Time-Varying Graphs from Online Data

Natali, Alberto; Isufi, Elvin; Coutino, Mario; Leus, Geert

DOI
10.1109/OJSP.2022.3178901

Publication date
2022

Document Version
Final published version

Published in
IEEE Open Journal of Signal Processing

Received 22 October 2021; revised 6 May 2022; accepted 19 May 2022. Date of publication 30 May 2022;
date of current version 5 July 2022. The review of this article was arranged by Associate Editor Cedric Richard.

Digital Object Identifier 10.1109/OJSP.2022.3178901

Learning Time-Varying Graphs From Online Data

ALBERTO NATALI [1] (Student Member, IEEE), ELVIN ISUFI [1], MARIO COUTINO [2] (Member, IEEE), AND GEERT LEUS [1] (Fellow, IEEE)

[1] Faculty of Electrical Engineering, Mathematics, and Computer Science, Delft University of Technology, 2628CD Delft, The Netherlands
[2] Radar Technology, TNO, NL-2509 The Hague, The Netherlands

CORRESPONDING AUTHOR: ALBERTO NATALI (e-mail: a.natali@tudelft.nl).

An earlier version of this work was presented in ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [DOI: 10.1109/ICASSP39728.2021.9415053].

ABSTRACT This work proposes an algorithmic framework to learn time-varying graphs from online data. The generality offered by the framework renders it model-independent, i.e., it can be theoretically analyzed in its abstract formulation and then instantiated under a variety of model-dependent graph learning problems. This is possible by phrasing (time-varying) graph learning as a composite optimization problem, where different functions regulate different desiderata, e.g., data fidelity, sparsity or smoothness. Instrumental for the findings is recognizing that the dependence of the majority (if not all) data-driven graph learning algorithms on the data is exerted through the empirical covariance matrix, representing a sufficient statistic for the estimation problem. Its user-defined recursive update enables the framework to work in non-stationary environments, while iterative algorithms building on novel time-varying optimization tools explicitly take into account the temporal dynamics, speeding up convergence and implicitly including a temporal-regularization of the solution. We specialize the framework to three well-known graph learning models, namely, the Gaussian graphical model (GGM), the structural equation model (SEM), and the smoothness-based model (SBM), where we also introduce ad-hoc vectorization schemes for structured matrices (symmetric, hollows, etc.) which are crucial to perform correct gradient computations, other than enabling to work in low-dimensional vector spaces and hence easing storage requirements. After discussing the theoretical guarantees of the proposed framework, we corroborate it with extensive numerical tests in synthetic and real data.

INDEX TERMS Graph topology identification, dynamic graph learning, network topology inference, graph signal processing.

I. INTRODUCTION

Learning network topologies from data is very appealing. On the *interpretable* side, the structure of a network reveals important descriptors of the network itself, providing to humans a prompt and explainable decision support system; on the *operative* side, it is a requirement for processing and learning architectures operating on graph data, such as graph filters [2]. When this structure is not readily available from the application, a fundamental question is how to *learn* it from data. The class of problems and the associated techniques concerning the identification of a network structure (from data) are known as graph topology identification (GTI), graph learning, or network topology inference [3], [4].

While up to recent years the GTI problem has been focused on learning *static* networks, i.e., networks which do not change their structure over time, the pervasiveness of networks with a *time-varying* component has quickly demanded new learning paradigms. This is the case for biological networks [5], subject to changes due to genetic and environmental factors, or financial markets [6], subject to changes due to political factors, among others. In these scenarios, a static approach would fail in accounting for the temporal variability of the underlying structure, which is strategic to, e.g., detect anomalies or discover new emerging communities.

In addition, prior (full) data availability should not be considered as a given. In real time applications, data need to be processed on-the-fly with low latency to, e.g., identify and block cyber-attacks in a communication infrastructure, or fraudulent transactions in a financial network. Thus, another learning component to take into account, is the modality of data acquisition. Here, we consider the extreme case in which data are processed on-the-fly, i.e., a fully *online* scenario.

It is then clear how the necessity of having algorithms to learn time-varying topologies from online data is motivated by physical scenarios. For clarity, we elaborate on the three keywords - identification, time-varying and online - which constitute, other than the title of the present work, also its main pillars.

- *Identification/learning:* it refers to the (optimization) process of learning the graph topology.
- *Time-Varying/dynamic:* it refers to the temporal variability of the graph in its edges, in opposition to the static case.
- *Online/streaming:* it refers to the modality in which the data arrive and/or are processed, in opposition to a batch approach which makes use of the entire bulk of data.

This emphasis on the terminology is important to understand the differences between the different existing works, presented next.

A. RELATED WORKS

Static GTI has been originally addressed from a statistical viewpoint and only in the past decade under a graph signal processing (GSP) framework [7], in which different assumptions are made on how the data are coupled with the unknown topology; see [3], [4] for a tutorial. Only recently, dynamic versions of the static counterparts have been proposed. For instance, [8], [9] learn a sequence of graphs by enforcing a prior (smoothness or sparsity) on the edges of consecutive graphs; similarly, the work in [10] extends the graphical Lasso [11] to account for the temporal variability, i.e., by estimating a sparse time-varying precision matrix. In addition to these works, the inference of causal relationships in the network structure, i.e., directed edges, has been considered in [12], [13]. See [14] for a review of dynamic topology inference approaches.

The mentioned approaches tackle the dynamic graph learning problem by means of a two-step approach: *i)* first, all the samples are collected and split into possibly overlapping windows; *ii)* only then the topology associated to each window is inferred from the data, possibly constrained to be similar to the adjacent ones. This modus-operandi fails to address the *online* (data-streaming) setting, where data have to be processed on-the-fly either due to architectural (memory, processing) limitations or (low latency) application requirements, such as real-time decision making.

This line of work has been freshly investigated by [15], which considers signals evolving according to a heat diffusion process, and by [16], which assumes the data are graph stationary [17]. In [18], the authors consider a vector autoregressive model to learn causality graphs by exploiting the temporal dependencies, while [19] proposes an online task-dependent (classification) graph learning algorithm, in which class-specific graphs are learned from labeled signals (training phase) and then used to classify new unseen data.

Differently from these works, our goal here is to provide a general (model-independent) algorithmic framework for time-varying GTI from online data that can be specialized to a variety of static graph learning problems. In particular, the generalization given by the framework enables us to render a static graph learning problem into its time-varying counterpart and to solve it via novel time-varying optimization techniques [20], providing a trade off between the solution accuracy and the velocity of execution. We introduce ad-hoc vectorization schemes for structured matrices to solve graph learning problems in the context of the Gaussian graphical model, the structural equation model, and the smoothness based model. All in all, a mature time-varying GTI framework for online data is yet to be conceived. This is our attempt to pave the way for a unified and general view of the problem, together with solutions to solve it.

B. CONTRIBUTIONS

This paper proposes a general-purpose algorithmic blueprint which unifies the theory of learning time-varying graphs from online data. The specific contributions of this general framework are:

a) it is *model-independent*, i.e., it can be analyzed in its abstract form and then specialized under different graph learning models. We show how to instantiate three such models, namely, the Gaussian graphical model (GGM), the structural equation model (SEM) and the smoothness-based model (SBM);

b) it operates in *non-stationary* environments, i.e., when the data statistics change over time. This is possible by expressing the considered models in terms of the sample covariance matrix, which can be then updated recursively for each new streaming sample with a user-defined function, which discards past information.

c) it is *accelerated* through a prediction-correction strategy, which takes into account the time-dimension. Its iterative nature enables a trade-off between following the optimal solution (accuracy) and an approximate solution (velocity). It also exhibits an implicit regularization of the cost function due to the limited iteration budget at each time-instant, i.e., similar solutions at closed time instants are obtained.

Notation: we use $x(i)$ and $X(i, j)$ to denote the i-th entry of the column vector $\mathbf{x}$ and the ij-th entry of the matrix $\mathbf{X}$, respectively. Superscripts $^\top$ and † denote the transpose and the pseudoinverse of a matrix, respectively, while operators $\mathrm{tr}(\cdot)$ and $\mathrm{vec}(\cdot)$ denote the matrix trace and matrix vectorization, respectively. The vectors $\mathbf{0}$ and $\mathbf{1}$, and the matrix $\mathbf{I}$, denote the all-zeros vector, the all-ones vector, and the identity matrix, with dimension clarified in the context. The operators $\otimes$, $\odot$, $\oslash$ and $^\circ$ stand for Kronecker product, Hadamard (entry-wise) product, Hadamard (entry-wise) division and Hadamard (entry-wise) power, respectively. We have $[\,\cdot\,]_+ = \max(\mathbf{0}, \cdot)$, where the maximum operates in an entry-wise fashion. Also, $\iota_{\mathcal{X}}(\cdot)$ is the indicator function for the convex set $\mathcal{X}$, for which holds $\iota_{\mathcal{X}}(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathcal{X}$ and $+\infty$ otherwise. Given two functions $f(\cdot)$ and $g(\cdot)$, $f \circ g(\cdot)$ denotes their composition. A function $f(\cdot)$ with argument $\mathbf{x} \in \mathbb{R}^N$, which is parametrized by the time t, is denoted by $f(\mathbf{x}; t)$. The gradient of the function $f(\mathbf{x}; t)$ with respect to $\mathbf{x}$ at the point $(\mathbf{x}; t)$ is denoted with $\nabla_{\mathbf{x}} f(\mathbf{x}; t)$, while $\nabla_{\mathbf{xx}} f(\mathbf{x}; t)$ denotes the Hessian evaluated at the same point. The time derivative of the gradient, denoted with $\nabla_{t\mathbf{x}} f(\mathbf{x}; t)$, is the partial derivative of $\nabla_{\mathbf{x}} f(\mathbf{x}; t)$ with respect to the time t, i.e., the mixed first-order partial derivative vector of the objective. Finally, $\|\cdot\|_p$ denotes the ℓ_p norm of a vector or, for a matrix, the ℓ_p norm of its vectorization. The Frobenius norm of a matrix is denoted with $\|\cdot\|_F$. Without any subscript, the norm $\|\cdot\|$ indicates the spectral norm.

II. PROBLEM FORMULATION

In this section, we formalize the problem of learning graphs from data. In Section II-A, we introduce the static graph topology inference problem, where we also recall three well-known models from the literature. Then, in Section II-B we formulate the (online) dynamic graph topology inference problem.

A. GRAPH TOPOLOGY IDENTIFICATION

We consider data living in a non-Euclidean domain described by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{S}\}$, where $\mathcal{V} = \{1, \ldots, N\}$ is the vertex set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, and $\mathbf{S}$ is an $N \times N$ matrix encoding the topology of the graph. The matrix $\mathbf{S}$ is referred to as the graph shift operator (GSO) and typical instantiations include the (weighted) adjacency matrix $\mathbf{W}$ [7] and the graph Laplacian $\mathbf{L}$ [21]. By associating to each node $i \in \mathcal{V}$ a scalar value $x(i)$, we define $\mathbf{x} = [x(1), \ldots, x(N)]^\top \in \mathbb{R}^N$ as a *graph signal* mapping the node set to the set of real numbers.

Consider now the matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ that stacks over the columns T graph signals generated from an unknown graph-dependent process $\mathcal{F}(\cdot)$; i.e., $\mathbf{X} = \mathcal{F}(\mathbf{S})$. Then, a GTI algorithm aims to learn the graph topology, i.e., to solve the "inverse" problem (not always well defined):

$$\mathbf{S} = \mathcal{F}^{-1}(\mathbf{X}). \tag{1}$$

The function $\mathcal{F}(\cdot)$ basically describes how the data are coupled with the graph and its knowledge is crucial. The data and the graph alone are insufficient to cast a meaningful graph learning problem. On one side, we need to know how the data depends on the graph from which they are generated. On the other side, we have to enforce some prior knowledge on the graph we want to learn.

Graph-data models. The choice of a data model is the forerunner of any GTI technique and, together with the graph-data coupling priors (e.g., smoothness, bandlimitedness) differentiates the different approaches. Due to their relevance for this work, we recall three widely used topology identification methods, namely the Gaussian graphical model [22], the structural equation model [23], and the smoothness-based model [24].

1) GAUSSIAN GRAPHICAL MODEL (GGM)

assumes each graph signal $\mathbf{x}_t$ is drawn from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and positive-definite covariance matrix $\boldsymbol{\Sigma}$. By setting the graph shift operator to be the precision matrix $\mathbf{S} = \boldsymbol{\Sigma}^{-1}$, *graph learning in a GGM amounts to precision matrix estimation*, which in a maximum likelihood (MLE) sense can be formulated as:

$$\begin{aligned} &\underset{\mathbf{S}}{\text{minimize}} - \log\det(\mathbf{S}) + \operatorname{tr}\left(\mathbf{S}\hat{\boldsymbol{\Sigma}}\right) \\ &\text{s. t.} \quad \mathbf{S} \in \mathbb{S}^N_{++} \end{aligned} \tag{2}$$

where $\hat{\boldsymbol{\Sigma}} = \frac{1}{T}\mathbf{X}\mathbf{X}^\top$ is the sample covariance matrix and $\mathbb{S}^N_{++}$ is the convex cone of positive-definite matrices. In this context, matrix $\mathbf{S}$ can be interpreted as the adjacency matrix (with self loops), although the problem can also be solved under some additional constraints forcing $\mathbf{S}$ to be a Laplacian [25].

2) STRUCTURAL EQUATION MODEL (SEM)

neglecting possible external inputs, and assuming an undirected graph, the SEM poses a linear dependence between the signal value $x_t(i)$ at node i and the signal values at some other nodes $\{x_t(j)\}_{j \neq i}$, representing the endogenous variables, i.e.,:

$$x_t(i) = \sum_{j \neq i} S(i, j) x_t(j) + e_t(i), \quad t = 1, \ldots, T \tag{3}$$

where $S(i, j)$ weights the influence that node j exerts on node i, and $e_t(i)$ represents unmodeled effects. In this view, with $\mathbf{S}$ encoding the graph connectivity, model (3) considers each node to be influenced only by its one-hop neighbors. In vector form, we can write (3) as:

$$\mathbf{x}_t = \mathbf{S}\mathbf{x}_t + \mathbf{e}_t, \quad t = 1, \ldots, T, \tag{4}$$

with $S(i, i) = 0$, for $i = 1, 2, \ldots, N$. Also, we consider $\mathbf{e}_t$ white noise with standard deviation σ_e. Graph learning under a SEM implies estimating matrix $\mathbf{S}$ by solving:

$$\begin{aligned} &\underset{\mathbf{S}}{\text{minimize}} \ \ \tfrac{1}{2T}\|\mathbf{X} - \mathbf{S}\mathbf{X}\|_F^2 + g(\mathbf{S}), \\ &\text{s. t.} \quad \mathbf{S} \in \mathcal{S} \end{aligned} \tag{5}$$

where $\mathcal{S} = \{\mathbf{S} |\operatorname{diag}(\mathbf{S}) = \mathbf{0}, S(i, j) = S(j, i), i \neq j\}$, and $g(\mathbf{S})$ is a regularizer enforcing $\mathbf{S}$ to have specific properties; e.g., sparsity. In this context, matrix $\mathbf{S}$ is usually interpreted as the adjacency matrix of the network (without self loops). The first term of (5) can be equivalently rewritten as:

$$f(\mathbf{S}) = \frac{1}{2T}\|\mathbf{X} - \mathbf{S}\mathbf{X}\|_F^2 = \frac{1}{2}\left[\operatorname{tr}\left(\mathbf{S}^2\hat{\boldsymbol{\Sigma}}\right) - 2\operatorname{tr}\left(\mathbf{S}\hat{\boldsymbol{\Sigma}}\right) + \operatorname{tr}\left(\hat{\boldsymbol{\Sigma}}\right)\right]. \tag{6}$$

which highlights its dependence on $\hat{\boldsymbol{\Sigma}}$.

3) SMOOTHNESS-BASED MODEL (SBM)

assumes each graph signal $\mathbf{x}_t$ to be smooth over the graph $\mathcal{G}$, where the notion of graph-smoothness is formally captured by the Laplacian quadratic form:

$$\mathrm{LQ}_{\mathcal{G}}(\mathbf{x}_t) := \mathbf{x}_t^\top \mathbf{L}\mathbf{x}_t = \sum_{i \neq j} W(i,j)(x_t(i) - x_t(j))^2. \tag{7}$$

A low value of $\mathrm{LQ}_{\mathcal{G}}(\mathbf{x}_t)$ suggests that adjacent nodes i and j have similar values $x_t(i)$ and $x_t(j)$ when the edge weight $W(i,j)$ is high.

Thus, the quantity:

$$\overline{\mathrm{LQ}}_{\mathcal{G}}(\mathbf{X}) = \frac{1}{T}\sum_{t=1}^{T}\mathrm{LQ}_{\mathcal{G}}(\mathbf{x}_t) = \frac{1}{T}\mathrm{tr}\left(\mathbf{X}^\top\mathbf{L}\mathbf{X}\right) = \mathrm{tr}\left(\mathbf{L}\hat{\mathbf{\Sigma}}\right) \tag{8}$$

represents the average signal smoothness on top of $\mathcal{G}$, which can be rewritten as the graph-dependent function:

$$f(\mathbf{S}) = \mathrm{tr}\left(\mathrm{Diag}(\mathbf{S}\mathbf{1})\hat{\mathbf{\Sigma}}\right) - \mathrm{tr}\left(\mathbf{S}\hat{\mathbf{\Sigma}}\right) \tag{9}$$

with $\mathbf{S} = \mathbf{W}$. Building upon this quantity, graph learning under a graph smoothness prior can be casted as:

$$\begin{aligned} &\underset{\mathbf{S}}{\text{minimize}} \quad f(\mathbf{S}) + g(\mathbf{S}) \\ &\text{s. t. } \mathbf{S} \in \mathcal{S} \end{aligned} \tag{10}$$

where the term $g(\mathbf{S})$ accommodates for additional topological properties (e.g., sparsity) and also helps avoiding the trivial solution $\mathbf{S} = \mathbf{0}$. The set $\mathcal{S} = \{\mathbf{S} | \,\mathrm{diag}(\mathbf{S}) = \mathbf{0}, S(i,j) = S(j,i) \geq 0, i \neq j\}$ encodes the topological structure, which coincides with the set of hollow symmetric matrices (i.e., with zeros on the diagonal) with positive entries.

Remark 1: In [24], the authors express the smoothness quantity (8) in terms of the weighted adjacency matrix $\mathbf{W}$ and a matrix $\mathbf{Z} \in \mathbb{R}_+^{N\times N}$ representing the row-wise (squared) Euclidean distance matrix of $\mathbf{X}$; i.e., $\mathrm{tr}(\mathbf{X}^\top\mathbf{L}\mathbf{X}) = \frac{1}{2}\mathrm{tr}(\mathbf{W}\mathbf{Z}) = \frac{1}{2}\|\mathbf{W}\odot\mathbf{Z}\|_1$. This formulation mainly brings the intuition that adding explicitly a sparsity term to the objective function would simply add a constant term to $\mathbf{Z}$. We favour (9) as a measure of graph signal smoothness since it fits within our framework, as will be clear soon. We emphasize however how the two formulations are equivalent, since $\hat{\mathbf{\Sigma}}$ can be directly expressed as a function of $\mathbf{Z}$.

B. ONLINE TIME-VARYING TOPOLOGY IDENTIFICATION

When the graph topology changes over time, the changing interactions are represented by the sequence of graphs $\{\mathcal{G}_t = \{\mathcal{V}, \mathcal{E}_t, \mathbf{S}_t\}\}_{t=1}^{\infty}$, where $t \in \mathbb{N}_+$ is a discrete time index. This sequence of graphs, which is discrete in nature, can be interpreted as the sampling of some "virtual" continuous time-varying graph using the sampling period $h = 1$. To relate our expressions to existing literature, we will make the parameter h explicit in the formulas, yet it is important to remember that $h = 1$. Together with the graph sequence $\{\mathcal{G}_t\}_{t=1}^{\infty}$, we consider also streaming graph signals $\{\mathbf{x}_t\}_{t=1}^{\infty}$, such that signal $\mathbf{x}_t$ is associated to graph $\mathcal{G}_t$. At this point, we are ready to formalize the time-varying graph topology identification (TV-GTI) problem.

Problem statement. Given an online sequence of graph signals $\{\mathbf{x}_t\}_{t=1}^{\infty}$ arising from an unknown time-varying network, the goal is to identify the time-varying graph topology $\{\mathcal{G}_t\}_{t=1}^{\infty}$; i.e., to learn the graph shift operator sequence $\{\mathbf{S}_t\}_{t=1}^{\infty}$ from $\{\mathbf{x}_t\}_{t=1}^{\infty}$. On top of this, to highlight the trade-off between accuracy and low-latency of the algorithm's solution.

Mathematically, our goal is to solve the sequence of time-invariant problems:

$$\mathbf{S}_t^\star := \arg\min_{\mathbf{S}} F(\mathbf{S}; t) \quad t = 1, 2, \ldots \tag{11}$$

where function $F(\cdot; t)$ is a time-varying cost function that depends on the data model [cf. Section II-A], and the index t makes the dependence on time explicit, which is due to the arrival of new data. Although we can solve problem (11) for each t separately with (static) convex optimization tools, the need of a low-latency stream of solutions makes this strategy unappealing. This approach also fails to capture the inherent temporal structure of the problem, i.e, it does not exploit the prior time-dependent structure of the graph, which is necessary in time-critical applications.

To exploit also this temporal information, we build on recent advances of time-varying optimization [20], [26] and propose a general framework for TV-GTI suitable for non-stationary environments. The proposed approach operates on-the-fly and updates the solution as a new signal $\mathbf{x}_t$ becomes available. The generality of this formulation enables us to define a *template* for the TV-GTI problem, which can be specialized to a variety of static GTI methods. The only information required is the first-order (gradient) and possibly second-order (Hessian) terms of the function. In the next section, we lay down the mathematics of the proposed approach. The central idea is to follow the optimal time-varying solution of problem (11) with lightweight proximal operations [27], which can be additionally accelerated with a *prediction-correction* strategy. This strategy, differently from other adaptive optimization strategies such as least mean squares and recursive least squares, uses an evolution model to predict the solution, and observes new data to correct the predictions. The considerations of Section III will be then specialized to the different data models of Section II-A in Section IV, further analyzed theoretically in Section V, and finally validated experimentally in Section VI.

III. ONLINE DYNAMIC GRAPH LEARNING

To maintain our discussion general, we consider the *composite* time-varying function:

$$F(\mathbf{S}; t) := f(\mathbf{S}; t) + \lambda g(\mathbf{S}; t) \tag{12}$$

where $f : \mathbb{R}^{N\times N} \times \mathbb{N}_+ \to \mathbb{R}$ is a smooth[1] strongly convex function [28] encoding a fidelity measure and $g : \mathbb{R}^{N\times N} \times \mathbb{N}_+ \to \mathbb{R}$ is a closed convex and proper function, potentially non differentiable, representing possible regularization terms.

[1] We use the term smoothness for functions and the term graph-smoothness for graph signals.

For instance, function $f(\cdot)$ can be the GGM objective function of (2), the SEM least-squares term of (5), or the SBM smoothness measure in (8).

Solving a time-varying optimization problem implies solving the *template* problem:

$$\mathbf{S}_t^\star := \arg\min_{\mathbf{S}} f(\mathbf{S}; t) + \lambda g(\mathbf{S}; t) \quad \text{for } t = 1, 2, \ldots \tag{13}$$

In other words, the goal is to find the sequence of optimal solutions $\{\mathbf{S}_t^\star\}_{t=1}^{\infty}$ of (13), which we will also call the *optimal trajectory*. However, solving exactly problem (13) in real time is infeasible because of the computational and time constraints. The exact solution may also be unnecessary since by itself it still approximates the true underlying time-varying graph. Under these considerations, an online algorithm that updates the approximate solution $\hat{\mathbf{S}}_{t+1}$ of (13) at time $t+1$, based on the former (approximate) solution $\hat{\mathbf{S}}_t$ is highly desirable for low complexity and fast execution.[2]

A. REDUCTION

Instrumental for the upcoming analysis is to observe that the number of independent variables of the graph representation matrix plays an important role in terms of storage requirements, processing complexity and, most importantly, in the correct computations of function derivatives with respect to those variables. Thus, when considering structured matrices, such as symmetric, hollow or diagonal, we need to take into account their structure. We achieve this by ad-hoc vectorization schemes through duplication and elimination matrices, inspired by [29].

Consider a matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$ and its corresponding "standard" vectorization $\text{vec}(\mathbf{S}) \in \mathbb{R}^{N^2}$. Depending on the specific structure of $\mathbf{S}$, different reduction and vectorization schemes can be adopted, leading to a lift from a matrix space to a vector space. The following spaces are of interest.

h-space. If $\mathbf{S}$ is symmetric, the number of independent variables is $k = N(N+1)/2$, i.e., the variables in its diagonal and its lower (equivalently, upper) triangular part. We can isolate these variables by representing matrix $\mathbf{S}$ with its *half-vectorization* form, which we denote as $\mathbf{s} = \text{vech}(\mathbf{S}) \in \mathbb{R}^k$. This isolation is possible by introducing the elimination matrix $\mathbf{E} \in \mathbb{R}^{k\times N^2}$ and the duplication matrix $\mathbf{D} \in \mathbb{R}^{N^2\times k}$ which respectively selects the independent entries of $\mathbf{S}$, i.e., $\mathbf{E}\,\text{vec}(\mathbf{S}) = \mathbf{s}$, and duplicates the entries of $\mathbf{s}$, i.e, $\mathbf{D}\mathbf{s} = \text{vec}(\mathbf{S})$. We call this vector space as the half-vectorization space (h-space).

hh-space. If $\mathbf{S}$ is symmetric and hollow, the number of independent variables is $l = N(N-1)/2$, i.e., the variables on its strictly lower (equivalently, upper) triangular part. In this case, we can represent matrix $\mathbf{S}$ in its *hollow half-vectorization* form, which we denote as $\mathbf{s} = \text{vechh}(\mathbf{S}) \in \mathbb{R}^l$. This reduction is achieved by applying the hollow elimination and duplication matrices $\mathbf{E}_h \in \mathbb{R}^{l\times N^2}$ and $\mathbf{D}_h \in \mathbb{R}^{N^2\times l}$, respectively, to the vectorization of $\mathbf{S}$. In particular, $\mathbf{E}_h$ extracts the variables of the strictly lower triangular part of the matrix, i.e., $\mathbf{s} = \mathbf{E}_h\,\text{vec}(\mathbf{S})$, while $\mathbf{D}_h$ duplicates the values and fills in zeros in the correct positions, i.e., $\text{vec}(\mathbf{S}) = \mathbf{D}_h\mathbf{s}$. We refer to the associated vector space as the hollow half-vectorization space (hh-space).

With the above discussion in place, we can now illustrate the general framework in terms of vector-dependent functions $f(\mathbf{s})$ for a vector $\mathbf{s}$, in contrast to matrix-dependent functions $f(\mathbf{S})$, simplifying exposition and notation. However, we underline that the information embodied in $\mathbf{S}$ and $\mathbf{s}$ is the same.

B. FRAMEWORK

We develop a prediction-correction strategy for problem (13) that starts from an estimate $\hat{\mathbf{s}}_t$ at time instant t, and *predicts* how this solution will change in the next time step $t+1$. This predicted topology is then *corrected* after a new datum $\mathbf{x}_{t+1}$ is available at time $t+1$. More specifically, the scheme has the following two steps:

1) *Prediction:* at time t, an approximate function $\hat{F}(\mathbf{s}; t+1)$ of the true yet *unobserved* function $F(\mathbf{s}; t+1)$ is formed, using only information available at time t. Then, using this approximated cost, we derive an estimate $\mathbf{s}_{t+1|t}^\star$, of how the topology will be at time $t+1$, using only the information up to time t. This estimate is found by solving:

$$\mathbf{s}_{t+1|t}^\star := \arg\min_{\mathbf{s}} \hat{F}(\mathbf{s}; t+1). \tag{14}$$

To avoid solving (14) for each t, we find an estimate $\hat{\mathbf{s}}_{t+1|t}$ by applying P iterations of a problem-specific descent operator $\hat{\mathcal{T}}$ (e.g., gradient descent, proximal gradient) for which $\mathbf{s}_{t+1|t}^\star = \hat{\mathcal{T}}\mathbf{s}_{t+1|t}^\star$, i.e., $\mathbf{s}_{t+1|t}^\star$ is a fixed point of $\hat{\mathcal{T}}$. See Appendix A for possible instances of $\hat{\mathcal{T}}$. In other words, problem (14) is solved recursively as:

$$\hat{\mathbf{s}}^{p+1} = \hat{\mathcal{T}}\hat{\mathbf{s}}^p, \quad p = 0, 1, \ldots, P-1 \tag{15}$$

with $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_t$. Once P steps are performed, the predicted topology is set to $\hat{\mathbf{s}}_{t+1|t} = \hat{\mathbf{s}}^P$, which approximates the solution of (14) and, in turn, will be close to $\mathbf{s}_{t+1}^\star$ at time $t+1$.

For our framework, we consider a Taylor-expansion based prediction to approximate the first term of $F(\cdot; t+1)$, i.e., $f(\cdot; t+1)$ [cf. (12)], leading to the following quadratic function:

$$\hat{f}(\mathbf{s}; t+1) = \frac{1}{2}\mathbf{s}^\top \nabla_{\mathbf{ss}} f(\hat{\mathbf{s}}_t; t)\,\mathbf{s} + \big[\nabla_{\mathbf{s}} f(\hat{\mathbf{s}}_t; t) + h\nabla_{t\mathbf{s}} f(\hat{\mathbf{s}}_t; t) - \nabla_{\mathbf{ss}} f(\hat{\mathbf{s}}_t; t)\,\hat{\mathbf{s}}_t\big]^\top \mathbf{s} \tag{16}$$

where $\nabla_{\mathbf{ss}} f(\cdot) \in \mathbb{R}^{N\times N}$ is the Hessian matrix of $f(\cdot)$ with respect to $\mathbf{s}$ and $\nabla_{t\mathbf{s}} f(\cdot) \in \mathbb{R}^N$ is the partial derivative of the gradient of $f(\cdot)$ w.r.t. time t.

[2] Problem (13) also endows the constrained case, in which the function $g(\cdot)$ comprises indicator functions associated to each constraint.

Algorithm 1: Online Time-Varying Graph Topology Inference.

```
Require: Feasible $\hat{\mathbf{S}}_0$, $f(\mathbf{S}; t_0)$, $P$, $C$, operators $\hat{\mathcal{T}}$ and $\mathcal{T}$
$\hat{\mathbf{s}}_0 \leftarrow$ ad-hoc vectorization of $\hat{\mathbf{S}}_0$
for $t = 0, 1, \ldots$ do
    // Prediction
    Initialize the predicted variable $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_t$
    for $p = 0, 1, \ldots, P-1$ do
        Predict $\hat{\mathbf{s}}^{p+1}$ with (15)
    end for
    Set the predicted variable $\hat{\mathbf{s}}_{t+1|t} = \hat{\mathbf{s}}^P$.
    // Correction - time $t+1$: new data arrive
    Initialize the corrected variable $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_{t+1|t}$
    for $c = 0, 1, \ldots, C-1$ do
        Predict $\hat{\mathbf{s}}^{c+1}$ with (18)
    end for
    Set the corrected variable $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}^C$
end for
```

To approximate the second term of $F(\cdot; t+1)$, i.e., $g(\cdot; t+1)$ [cf. (12)], we use a one step-back prediction, i.e., $\hat{g}(\mathbf{s}; t+1) = g(\mathbf{s}; t)$. This implies that $\hat{g}(\cdot)$ does not depend on t, which in turn makes the constraint set and the regularization term independent of time, an assumption usually met in state-of-the-art topology identification [3]. Henceforth, we will omit this time dependency.

2) *Correction:* at time $t+1$ the new data $\mathbf{x}_{t+1}$ and hence the cost function $F(\mathbf{s}; t+1)$ becomes available. Thus, we correct the prediction $\hat{\mathbf{s}}_{t+1|t}$ by solving the correction problem:

$$\mathbf{s}^\star_{t+1} := \arg\min_{\mathbf{s}} F(\mathbf{s}; t+1). \tag{17}$$

Also in this case, we solve (17) with iterative methods to obtain an approximate solution $\hat{\mathbf{s}}_{t+1}$ by applying C iterations of an operator $\mathcal{T}$. In other words, the correction problem (17) is addressed through the recursion:

$$\hat{\mathbf{s}}^{c+1} = \mathcal{T}\hat{\mathbf{s}}^c, \quad c = 0, 1, \ldots, C-1 \tag{18}$$

with $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_{t+1|t}$. Once the C steps are performed, the correction graph $\hat{\mathbf{s}}_{t+1}$ is set to $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}^C$, which will approximate the solution $\mathbf{s}^\star_{t+1}$ of (17).

Algorithm 1 shows the pseudocode for the general online TV-GTI framework.

Remark 2: We point out that the framework can adopt different approximation schemes, such as extrapolation-based techniques, and can also include time-varying constraint sets. The choice of approximation-scheme depends on the properties of the problem itself along with the required prediction accuracy. For an in-depth theoretical discussion regarding different prediction approaches and relative convergence results, refer to [30].

IV. NETWORK MODELS AND ALGORITHMS

In this section, we specialize the proposed framework to the three static topology inference models discussed in Section II-A. Notice that the data dependency of data-driven graph learning algorithms is exerted via the empirical covariance matrix $\hat{\boldsymbol{\Sigma}}$ of the graph signals; we have already shown this for the three considered models of Section II-A. In other words, graph-dependent objective functions of the form $F(\mathbf{S})$ could be explicitly expressed through their parametrized version $F(\mathbf{S}; \hat{\boldsymbol{\Sigma}})$. This rather intuitive, yet crucial observation, is central to render the proposed framework model-independent and adaptive, as explained next.

Non-stationarity. Relying on the explicit dependence of function $F(\cdot)$ on $\hat{\boldsymbol{\Sigma}}$ and envisioning non-stationary environments, we let the algorithm be adaptive by discarding past information. That is, function $F(\mathbf{S}; t)$ in (12) can be written as $F(\mathbf{S}; \hat{\boldsymbol{\Sigma}}_t)$, with $\hat{\boldsymbol{\Sigma}}_t$ the empirical covariance matrix, up to time t, with past data gradually discarded. This makes the framework adaptive and model-independent. The adaptive behavior can be shaped by, e.g., the exponentially-weighted moving average (EWMA) of the covariance matrix:

$$\hat{\boldsymbol{\Sigma}}_t = \gamma\hat{\boldsymbol{\Sigma}}_{t-1} + (1-\gamma)\mathbf{x}_t\mathbf{x}_t^\top \quad t = 1, 2\ldots \tag{19}$$

where the forgetting factor $\gamma \in (0, 1)$ downweighs (for $\gamma \rightarrow 0$) or upweighs (for $\gamma \rightarrow 1$) past data contributions. For stationary environments, an option is the infinite-memory matrix covariance update $\hat{\boldsymbol{\Sigma}}_t = \frac{t-1}{t}\hat{\boldsymbol{\Sigma}}_{t-1} + \frac{1}{t}\mathbf{x}_t\mathbf{x}_t^\top$.

A. TIME-VARYING GAUSSIAN GRAPHICAL MODEL

The GGM problem (2), adapted to a time-varying setting following template (13) leads to:

$$f(\mathbf{S}; t) = -\log\det(\mathbf{S}) + \operatorname{tr}\left(\mathbf{S}\hat{\boldsymbol{\Sigma}}_t\right) \tag{20a}$$

$$g(\mathbf{S}; t) = \iota_{\mathcal{S}}(\mathbf{S}) \tag{20b}$$

where $\mathcal{S} = \mathbb{S}^N_{++}$. In this case $g(\cdot)$ encodes the constraint set of positive definite matrices and the regularization parameter is $\lambda = 1$.

Since $\mathbf{S}$ is symmetric, we use the half-vectorization $\mathbf{s} = \text{vech}(\mathbf{S}) \in \mathbb{R}^k$ to reduce the number of independent variables from N^2 to $k = N(N+1)/2$. Then, the gradient and the Hessian of the function $f(\cdot)$ in the h-space are respectively:

$$\nabla_{\mathbf{s}} f(\mathbf{s}; t) = \mathbf{D}^\top \text{vec}\left(\hat{\boldsymbol{\Sigma}}_t - \mathbf{S}^{-1}\right) \tag{21a}$$

$$\nabla_{\mathbf{ss}} f(\mathbf{s}; t) = \mathbf{D}^\top(\mathbf{S} \otimes \mathbf{S})^{-1}\mathbf{D}. \tag{21b}$$

Likewise, the discrete-time derivative of the gradient is given by the partial mixed-order derivative [26]:

$$\nabla_{t\mathbf{s}} f(\mathbf{s}; t) = \mathbf{D}^\top \text{vec}\left(\hat{\boldsymbol{\Sigma}}_t - \hat{\boldsymbol{\Sigma}}_{t-1}\right). \tag{22}$$

Note the Hessian term (21b) is time-independent, while the time-derivative of the gradient (22) is graph-independent.

Now, by defining $\hat{\mathbf{s}}_t := \text{vech}(\hat{\mathbf{S}}_{\text{t}}) \in \mathbb{R}^k$, we can particularize Algorithm 1 to:

- **Prediction:** with $\hat{\mathbf{s}}^0$ initialized as $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_t$, the prediction update is :

$$\hat{\mathbf{s}}^{p+1} = \mathbb{P}_{\mathcal{S}}\big[\hat{\mathbf{s}}^p - 2\alpha_t\big(\nabla_{\mathbf{s}} f(\hat{\mathbf{s}}_t;t) + \nabla_{\mathbf{ss}} f(\hat{\mathbf{s}}_t;t)\left(\hat{\mathbf{s}}^p - \hat{\mathbf{s}}_t\right) + h\nabla_{t\mathbf{s}} f(\hat{\mathbf{s}}_t;t)\big)\big] \quad (23)$$

  for $p = 0, 1, \ldots, P-1$, where α_t is a (time-varying) step size. (23) entails a descent step along the approximate function $\hat{f}(\cdot; t+1)$ in (16), followed by the projection onto the convex set $\mathcal{S}$; see Appendix A for the definition of $\mathbb{P}_{\mathcal{S}}(\cdot)$. 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 $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_{t+1|t}$, the correction update is:

$$\hat{\mathbf{s}}^{c+1} = \mathbb{P}_{\mathcal{S}}\left[\hat{\mathbf{s}}^c - \beta_t \nabla f\left(\hat{\mathbf{s}}^c; t+1\right)\right] \quad (24)$$

  for $c = 0, 1, \ldots, C-1$, where β_t is a (time-varying) step size. (24) entails a descent step along the true function $f(\cdot; t+1)$, followed by the projection onto the set $\mathcal{S}$. The correction $\hat{\mathbf{s}}_{t+1}$ is finally set to $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}^C$.

The prediction step (23) instantiates (15) to $\hat{\mathcal{T}} = \mathbb{P}_{\mathcal{S}} \circ (I - \alpha_t \nabla_{\mathbf{s}} \hat{f})(\cdot)$, where $I(\cdot)$ is the identity function $I(\mathbf{s}) = \mathbf{s}$. Similarly, the correction step (24) instantiates (18) to $\mathcal{T} = \mathbb{P}_{\mathcal{S}} \circ (I - \beta_t \nabla_{\mathbf{s}} f)(\cdot)$. The overall computational complexity of one PC iteration is dominated by the matrix inversion and matrix multiplication, incurring a cost of $\mathcal{O}(N^3)$. A correction-only algorithm would also incur a cost of $\mathcal{O}(N^3)$ per iteration. See Appendix C for details.

B. TIME-VARYING STRUCTURAL EQUATION MODEL

The SEM problem (5), adapted to a time-varying setting with sparsity-promoting regularizer, leads to [cf. (13)]:

$$f(\mathbf{S};t) = \frac{1}{2}\left[\operatorname{tr}\left(\mathbf{S}^2\hat{\boldsymbol{\Sigma}}_t\right) - 2\operatorname{tr}\left(\mathbf{S}\hat{\boldsymbol{\Sigma}}_t\right) + \operatorname{tr}\left(\hat{\boldsymbol{\Sigma}}_t\right)\right] \quad (25a)$$

$$g(\mathbf{S};t) = \|\mathbf{S}\|_1 + \iota_{\mathcal{S}}(\mathbf{S}) \quad (25b)$$

where $\mathcal{S} = \{\mathbf{S} \in \mathbb{S}^N \,|\, \operatorname{diag}(\mathbf{S}) = \mathbf{0}, S(i,j) = S(j,i), i \neq j\}$ is the set of hollow symmetric matrices, and $\|\mathbf{S}\|_1 = \|\operatorname{vec}(\mathbf{S})\|_1$. Since $\mathbf{S}$ is symmetric and hollow, we operate on the hh-space to make the problem unconstrained and reduce the number of independent variables from N^2 to $l = N(N-1)/2$, through its hollow half-vectorization form $\mathbf{s} = \operatorname{vechh}(\mathbf{S}) \in \mathbb{R}^l$. In the hh-space, (25a) and (25b) become:

$$f(\mathbf{s};t) = \frac{1}{2}\mathbf{s}^\top \mathbf{Q}_t \mathbf{s} - 2\mathbf{s}^\top \hat{\boldsymbol{\sigma}}_t + \frac{1}{2}\hat{\sigma}_t \quad (26a)$$

$$g(\mathbf{s};t) = 2\|\mathbf{s}\|_1 \quad (26b)$$

where $\mathbf{Q}_t := \mathbf{D}_h^\top(\hat{\boldsymbol{\Sigma}}_t \otimes \mathbf{I})\mathbf{D}_h$ with $\otimes$ denoting the Kronecker product, $\hat{\boldsymbol{\sigma}}_t = \operatorname{vechh}(\hat{\boldsymbol{\Sigma}}_t)$, and $\hat{\sigma}_t = \operatorname{tr}(\hat{\boldsymbol{\Sigma}}_t)$. Since $\mathbf{Q}_t \succeq 0$, (26a) is convex. To solve the time-varying SEM (TV-SEM) problem, we derive the gradient and the Hessian of function $f(\cdot)$ in the hh-space as:

$$\nabla_{\mathbf{s}} f(\mathbf{s};t) = \mathbf{Q}_t \mathbf{s} - 2\hat{\boldsymbol{\sigma}}_t \quad (27a)$$

$$\nabla_{\mathbf{ss}} f(\mathbf{s};t) = \mathbf{Q}_t \quad (27b)$$

Notice here how the Hessian is time-varying and independent on $\mathbf{s}$, differently from the GGM case. The time derivative of the gradient is given by the partial mixed-order derivative:

$$\nabla_{t\mathbf{s}} f(\mathbf{s};t) = \frac{1}{h}\left[\left(\mathbf{Q}_t - \mathbf{Q}_{t-1}\right)\mathbf{s} - 2\left(\hat{\boldsymbol{\sigma}}_t - \hat{\boldsymbol{\sigma}}_{t-1}\right)\right] \quad (28)$$

Now, by defining $\hat{\mathbf{s}}_t := \operatorname{vechh}(\hat{\mathbf{S}}_t) \in \mathbb{R}^l$, we can particularize Algorithm 1 to:

- **Prediction:** set $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_t$. Then, the prediction is the proximal-gradient update:

$$\mathbf{u}^p = \hat{\mathbf{s}}^p - \alpha_t\big[\nabla_{\mathbf{s}} f\left(\hat{\mathbf{s}}_t;t\right) + \nabla_{\mathbf{ss}} f\left(\hat{\mathbf{s}}_t;t\right)\left(\hat{\mathbf{s}}^p - \hat{\mathbf{s}}_t\right) + h\nabla_{t\mathbf{s}} f(\hat{\mathbf{s}}_t;t)\big] \quad (29a)$$

$$\hat{\mathbf{s}}^{p+1} = \operatorname{sign}(\mathbf{u}^p) \odot \left[|\mathbf{u}^p| - 2\alpha_t\lambda\mathbf{1}\right]_+ \quad (29b)$$

  for $p = 0, \ldots, P$. (29a) entails a descent step along the approximate function $\hat{f}(\cdot; t+1)$ in (16), followed by the non-negative soft-thresholding operator in (29b), which sets to zero all the (negative) edge weights of the graph obtained after the gradient descent in (29a). See Appendix A for the formal definition of proximal operator, leading to (29a) and (29b). The final prediction $\hat{\mathbf{s}}_{t+1|t}$ is set to $\hat{\mathbf{s}}_{t+1|t} = \hat{\mathbf{s}}^P$.
- **Correction:** set $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_{t+1|t}$. Then, the correction is the proximal-gradient update:

$$\mathbf{u}^c = \hat{\mathbf{s}}^c - \beta_t \nabla f\left(\hat{\mathbf{s}}^c; t+1\right) \quad (30a)$$

$$\hat{\mathbf{s}}^{c+1} = \operatorname{sign}(\mathbf{u}^c) \odot \left[|\mathbf{u}^c| - 2\beta_t\lambda\mathbf{1}\right]_+ \quad (30b)$$

  for $c = 0, \ldots, C-1$. (30a) entails a descent step along the true function $f(\cdot; t+1)$, followed by the non-negative soft-thresholding operator in (30b). Finally, $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}^C$.

The prediction step (29) instantiates (15) to $\hat{\mathcal{T}} = \operatorname{prox}_{\lambda g, \alpha_t} \circ (I - \alpha_t \nabla_{\mathbf{s}} \hat{f})(\cdot)$. Similarly, the correction step (30) instantiates (18) to $\mathcal{T} = \operatorname{prox}_{\lambda g, \beta_t} \circ (I - \beta_t \nabla_{\mathbf{s}} f)(\cdot)$. The overall computational complexity of one PC iteration is dominated by the computation of matrix $\mathbf{Q}_t$, incurring a cost of $\mathcal{O}(N^3)$. A correction-only algorithm would also incur a cost of $\mathcal{O}(N^3)$ per iteration. See Appendix C for details.

C. TIME-VARYING SMOOTHNESS-BASED MODEL

The SBM model (10) adapted to a time-varying setting is:

$$f(\mathbf{S};t) = \operatorname{tr}\left(\operatorname{Diag}(\mathbf{S}\mathbf{1})\hat{\boldsymbol{\Sigma}}_t\right) - \operatorname{tr}\left(\mathbf{S}\hat{\boldsymbol{\Sigma}}_t\right) \quad (31a)$$

$$g(\mathbf{S};t) = \frac{\lambda_1}{4}\|\mathbf{S}\|_F^2 - \lambda_2 \mathbf{1}^\top \log(\mathbf{S}\mathbf{1}) + \iota_{\mathcal{S}}(\mathbf{S}) \quad (31b)$$

where $\mathcal{S} = \{\mathbf{S} \in \mathbb{S}^N | \operatorname{diag}(\mathbf{S}) = \mathbf{0}, S(i,j) = S(j,i) \geq 0, i \neq j\}$ is the set of hollow symmetric matrices. The log barrier term $\log(\mathbf{S}\mathbf{1})$ is applied entry-wise and forces the nodes degree vector $\mathbf{d} = \mathbf{S}\mathbf{1}$ to be positive while avoiding the trivial solution. The Frobenius norm term $\|\mathbf{S}\|_F^2$ controls the sparsity of the graph.

By operating in the hh-space, (31a) and (31b) become:[3]

$$f(\mathbf{s};t)=\mathbf{s}^\top\left(\mathbf{K}^\top\hat{\boldsymbol{\sigma}}_d-2\hat{\boldsymbol{\sigma}}_t\right)-\lambda_2\mathbf{1}^\top\log(\mathbf{K}\mathbf{s})+\frac{\lambda_1}{2}\|\mathbf{s}\|^2 \quad (32a)$$

$$g(\mathbf{s};t)=\iota_{\mathbb{R}_+}(\mathbf{s}) \quad (32b)$$

where $\mathbf{K}\in\{0,1\}^{N\times l}$ is the binary matrix such that $\mathbf{d}=\mathbf{S}\mathbf{1}=\mathbf{K}\mathbf{s}$, $\hat{\boldsymbol{\sigma}}_d=\mathrm{diag}(\hat{\boldsymbol{\Sigma}}_t)$ and $\hat{\boldsymbol{\sigma}}_t=\mathrm{vechh}(\hat{\boldsymbol{\Sigma}}_t)$.

To apply the proposed framework to solve the time-varying SBM (TV-SBM) problem, we derive the gradient and the Hessian of function $f(\cdot)$ in the hh-space as follows:

$$\nabla_{\mathbf{s}}f(\mathbf{s};t)=\lambda_1\mathbf{s}-\lambda_2\mathbf{K}^\top(\mathbf{1}\oslash\mathbf{K}\mathbf{s})+\mathbf{z}_t \quad (33a)$$

$$\nabla_{\mathbf{ss}}f(\mathbf{s};t)=\lambda_1\mathbf{I}+\lambda_2\mathbf{K}^\top\mathrm{Diag}(\mathbf{1}\oslash(\mathbf{K}\mathbf{s})^{\circ 2})\mathbf{K} \quad (33b)$$

where $\oslash$ and $^\circ$ represent the Hadamard division and power, respectively. The time derivative of the gradient is given by the partial mixed-order derivative:

$$\nabla_{t\mathbf{s}}f(\mathbf{s};t)=\frac{1}{h}(\mathbf{z}_t-\mathbf{z}_{t-1}) \quad (34)$$

where $\mathbf{z}_t=\mathbf{K}^\top\hat{\boldsymbol{\sigma}}_d-2\hat{\boldsymbol{\sigma}}_t$. Now, by defining $\hat{\mathbf{s}}_t:=\mathrm{vechh}(\hat{\mathbf{S}}_t)\in\mathbb{R}^l$, we can particularize Algorithm 1 to:

- **Prediction:** with $\hat{\mathbf{s}}^0$ initialized as $\hat{\mathbf{s}}^0=\hat{\mathbf{s}}_t$, the prediction update is:

$$\hat{\mathbf{s}}^{p+1}=\mathbb{P}_{\mathbf{s}\geq\mathbf{0}}\big[\hat{\mathbf{s}}^p-2\alpha_t\big(\nabla_{\mathbf{s}}f(\hat{\mathbf{s}}_t;t) + \nabla_{\mathbf{ss}}f(\hat{\mathbf{s}}_t;t)\left(\hat{\mathbf{s}}^p-\hat{\mathbf{s}}_t\right)+h\nabla_{t\mathbf{s}}f(\hat{\mathbf{s}}_t;t)\big)\big] \quad (35)$$

for $p=0,1,\ldots,P-1$. (35) entails a descent step along the approximate function $\hat{f}(\cdot;t+1)$ in (16), followed by the projection onto the non-negative orthant. 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 $\hat{\mathbf{s}}^0=\hat{\mathbf{s}}_{t+1|t}$, the correction update is:

$$\hat{\mathbf{s}}^{c+1}=\mathbb{P}_{\mathbf{s}\geq\mathbf{0}}\left[\hat{\mathbf{s}}^c-\beta_t\nabla f\left(\hat{\mathbf{s}}^c;t+1\right)\right], \quad (36)$$

for $c=0,1,\ldots,C-1$. (36) entails a descent step along the true function $f(\cdot;t+1)$, followed by the projection onto the non-negative orthant. Finally, $\hat{\mathbf{s}}_{t+1}=\hat{\mathbf{s}}^C$.

The prediction step (35) instantiates (15) to $\hat{\mathcal{T}}=\mathbb{P}_{\mathbf{s}\geq\mathbf{0}}\circ(I-\alpha_t\nabla_{\mathbf{s}}\hat{f})(\cdot)$. Similarly, the correction step (36) instantiates (18) to $\mathcal{T}=\mathbb{P}_{\mathbf{s}\geq\mathbf{0}}\circ(I-\beta_t\nabla_{\mathbf{s}}f)(\cdot)$. The overall computational complexity per iteration is dominated by the computation of the gradient $\nabla_{\mathbf{s}}f(\mathbf{s};t)$ (or the Hessian if $P>1$), incurring a cost of $\mathcal{O}(N^2)$ (or $\mathcal{O}(N^3)$) if $P>1$). See Appendix C for details.

V. CONVERGENCE ANALYSIS

In this section, we first discuss the convergence of Algorithm 1 and the associated error bounds. As solver we consider the proximal gradient $\hat{\mathcal{T}}=\mathcal{T}=\mathrm{prox}_{g,\rho}\circ(I-\rho\nabla_{\mathbf{s}}f)(\cdot)$ [31], [32]. Then, we show how the parameters of the three introduced models are involved in the bounds. To ease notation, we use $\mathbf{s}\in\mathbb{R}^p$ to indicate the vectorization of matrix variable $\mathbf{S}\in\mathbb{R}^{N\times N}$ [cf. Section III-A].

For this analysis, we need the following mild assumptions.

Assumption 1: The function $f:\mathbb{R}^p\times\mathbb{N}_+\rightarrow\mathbb{R}$ is m-strongly convex and L-smooth uniformly in t, i.e., $m\mathbf{I}\preceq\nabla_{\mathbf{ss}}f(\mathbf{s};t)\preceq L\mathbf{I},\ \ \forall\,\mathbf{s},t$, while the function $g:\mathbb{R}^p\times\mathbb{N}_+\rightarrow\mathbb{R}\cup\{+\infty\}$ is closed convex and proper, or $g(\cdot;t)=0$, for all $t\in\mathbb{N}_+$.

This guarantees that problem (13) admits a unique solution for each time instant, which in turn guarantees uniqueness of the solution trajectory $\{\mathbf{s}_t^\star\}_{t=1}^\infty$.

Assumption 2: The gradient of function $f(\cdot)$ has bounded time derivative, i.e. $\exists\, C_0>0$ such that $\|\nabla_{t\mathbf{s}}f(\mathbf{s};t)\|\leq C_0\ \forall\,\mathbf{s}\in\mathbb{R}^p, t\in\mathbb{N}_+$.

This guarantees that the solution trajectory is Lipschitz in time.

Assumption 3: The predicted function $\hat{f}(\cdot;t+1)$ is m-strongly convex and L-smooth uniformly in t; and $\hat{g}(\cdot;t+1)$ is closed, convex and proper.

This implies that the prediction problem (14) belongs to the same class as the original problem, i.e., the functions of the two problems share the same strong convexity and Lipschitz constants m and L. Therefore, the same solver can be applied for the prediction and correction steps, i.e., $\hat{\mathcal{T}}=\mathcal{T}$.

Assumption 4: The matrix $\mathbf{S}$ of (12) has finite entries, i.e., $-\infty<S(i,j)<+\infty$, for all i,j.

This guarantees $\|\mathbf{S}\|<+\infty$, i.e., $\mathbf{S}$ is a bounded operator, and it holds in practical scenarios. In particular, it is known that (finite) weighted graphs exhibit bounded eigenvalues, see [33][34]. Notably, if $\mathbf{S}$ is a normalized Laplacian, then $\|\mathbf{S}\|=2$.

Similarly, assumptions 1-3 are mild and hold for the considered models, as we show next.

Proposition 1: The three considered models of Section IV can be m-strongly convex and L-smooth uniformly in t, for some scalar m and L, as supported by the following claims.

Claim 1: Denote with $\xi>0$ and $0<\chi<\infty$ the minimum and maximum admissible eigenvalues of the precision matrix $\mathbf{S}$, respectively; i.e., consider the set $\mathcal{S}=\{\mathbf{S}\in\mathbb{S}^N_{++}|\xi\mathbf{I}\preceq\mathbf{S}\preceq\chi\mathbf{I}\}$. Then, for the TV-GGM function $f(\cdot;t)$ in (20a), it holds:

$$m=1/\chi\qquad L=2/\xi. \quad (37)$$

Claim 2: Denote with $\lambda_{\min}$ and $\lambda_{\max}$ the smallest and highest eigenvalues for the set of empirical covariance matrices obtained with graph signals obeying (4). Then, for the TV-SEM function $f(\cdot;t)$ in (26a), it holds:

$$m=\lambda_{\min}\qquad L=2\lambda_{\max}. \quad (38)$$

Claim 3: Consider the TV-SBM function $f(\cdot;t)$ in (32a), and recall that the log-barrier term avoids isolated vertices, i.e., $\mathbf{d}\succ\mathbf{0}$. Denote with $d_{\min}>0$ the minimum degree of the GSO search space. Under these assumptions, it holds:

$$m=2\lambda_1\qquad L=2\lambda_2(N-1)d_{\min}^{-2}. \quad (39)$$

See Appendix B for a proof of Claim 1-3.

[3] We move the log-barrier and Frobenius norm terms of $g(\cdot)$ function (31b) into the $f(\cdot)$ function to fit the structure of the general template.

Thus, Assumption 1 holds since the Hessian of $f(\cdot;t)$ is bounded over time and $g(\cdot;t)$ is closed, convex and proper by problem construction; Assumption 2 holds since $\nabla_{t\mathbf{s}} f(\mathbf{s};t)$ is the difference between bounded vectors which involve covariance matrices not too different from each other (one is the rank-one update of the other), which is finite as long as the graph signals are bounded, see (19) and, e.g., (34). Assumption 3 holds since $\hat{f}(\cdot;t+1)$ is a quadratic approximation of $f(\cdot;t)$ [cf. (16)] and $\hat{g}(\cdot;t+1)=g(\cdot;t)$, thus inheriting the properties of $f(\cdot;t)$ and $g(\cdot;t)$, which satisfy Assumption 1.

With this in place, we are now ready to show two different error bounds incurred during the prediction and correction steps performed by Algorithm 1, describing its sub-optimality as function of the model and algorithm's parameters. First, we show the error bound between the optimal prediction solution $\mathbf{s}^\star_{t+1|t}$ and the associated optimal correction $\mathbf{s}^\star_{t+1}$, which solve problems (14) and (17), respectively.

Proposition 2: Let Assumptions 1-3 hold. Consider also the Taylor expansion based prediction (16) for $f(\cdot;t)$ and the one-step back prediction for $g(\cdot;t)$. Then, the distance between the optimal prediction solution $\mathbf{s}^\star_{t+1|t}$, solving problem (14), and the associated optimal correction $\mathbf{s}^\star_{t+1}$, solving problem (17), is upper bounded by:

$$\|\mathbf{s}^\star_{t+1|t}-\mathbf{s}^\star_{t+1}\| \leq \frac{2L}{m}\|\hat{\mathbf{s}}_t-\mathbf{s}^\star_t\| + \frac{2C_0\,h}{m}\left(1+\frac{L}{m}\right) \tag{40}$$

where $\hat{\mathbf{s}}_t$ is the approximate solution of the correction problem (17) at time t.

Proof: Follows from [30, Lemma 4.2] in which constant $D_0=0$ by considering a static function $g(\cdot)$. ■

This bound enables us to measure how far the prediction is from the true corrected topology at time $t+1$. It depends on the estimation error $\hat{\mathbf{s}}_t-\mathbf{s}^\star_t$ achieved at time t, the ratio L/m and the variability of the function gradient $\nabla_{t\mathbf{s}} f(\mathbf{s};t)$. The bound suggests that a small gap can be achieved if *i)* the ratio L/m is small, which for the three considered models translates in having a small condition number for the involved covariance matrices or GSOs; and *ii)* the time-gradient $\nabla_{t\mathbf{s}} f(\mathbf{s};t)$ at consecutive time steps does not change significantly, which holds when the considered models have similar covariance matrices at adjacent time instants, i.e., the data statistics do not change too rapidly (see e.g. (22) and (28)).

Finally, we bound the error sequence $\{\|\hat{\mathbf{s}}_t-\mathbf{s}^\star_t\|_2, t=1,2,\ldots\}$ achieved by Algorithm 1 by means of the following non-asymptotic performance guarantee, which is an adaptation of [30, Proposition 5.1].

Theorem 4: Let Assumptions 1 and 3 hold, and consider two scalars $\{d_t,\phi_t\}\in\mathbb{R}_+$ such that:

$$\|\mathbf{s}^\star_{t+1}-\mathbf{s}^\star_t\|\leq d_t \quad \text{and} \quad \|\mathbf{s}^\star_{t+1|t}-\mathbf{s}^\star_{t+1}\|\leq \phi_t \tag{41}$$

for any $t\in\mathbb{N}_+$. Let also the prediction and correction steps use the same step-sizes $\rho_t=\alpha_t=\beta_t$. Then, by employing P prediction and C correction steps with the proximal gradient operator $\mathcal{T}=\mathrm{prox}_{g,\rho_t}\circ(I-\rho_t\nabla_{\mathbf{s}} f)(\cdot)$, the sequence of iterates $\{\hat{\mathbf{s}}_t\}$ generated by Algorithm 1 satisfies:

$$\|\hat{\mathbf{s}}_{t+1}-\mathbf{s}^\star_{t+1}\|_2 \leq q_t^C\left(q_t^P\|\hat{\mathbf{s}}_t-\mathbf{s}^\star_t\|+q_t^P d_t+\left(1+q_t^P\right)\phi_t\right) \tag{42}$$

where $q_t=\max\{|1-\rho_t m_t|,|1-\rho_t L_t|\}\in(0,1)$ is the contraction coefficient [35].

Proof: Follows from [30, Proposition 5.1] and [30, Lemma 2.5], with variables $\lambda=q_t$ and $\chi=\beta=1$. ■

Theorem (42) states that the sequence of estimated graphs $\{\mathbf{s}_t\}_{t\in\mathbb{N}_+}$ hovers around the optimal trajectory $\{\mathbf{s}^\star_t\}_{t\in\mathbb{N}_+}$ with a distance depending on: *i)* the numbers P and C of iterations; *ii)* the estimation error achieved at the previous time instant $\|\hat{\mathbf{s}}_t-\mathbf{s}^\star_t\|$; and *iii)* the quantities d_t and ϕ_t. Moreover, (42) is a contraction (i.e., $q_t^{C+P}<1$) when $\rho_t<2/L_t$; in this case the initial starting point $\hat{\mathbf{s}}_0$ does not influence the error $\hat{\mathbf{s}}_{t+1}-\mathbf{s}^\star_{t+1}$ asymptotically, since the first term in (42) vanishes. However, the terms d_t and ϕ_t keep impacting the error also asymptotically, as long as the problem is time-varying; if the problem becomes static, i.e., the solution stops varying, then $d_t=\phi_t=0$, and the overall error asymptotically goes to zero.

VI. NUMERICAL RESULTS

In this section, we show with numerical results how Algorithm 1, specialized to the three models (TV-GGM, TV-SEM, TV-SBM), can track the offline solution (13) obtained by the respective instantiations. For all the experiments, we initialize the empirical covariance matrix $\hat{\boldsymbol{\Sigma}}_0$ with some samples acquired prior to the analysis. We consider $P=1$ prediction steps and $C=1$ correction steps, which is the challenging setting of having the minimum iteration budget for streaming scenarios. We measure the convergence of Algorithm 1 via the normalized squared error (NSE) between the algorithm's estimate $\hat{\mathbf{s}}_t$ and the optimal (offline) solution $\mathbf{s}^\star_t$:

$$\mathrm{NSE}(\hat{\mathbf{s}}_t,\mathbf{s}^\star_t)=\frac{\|\hat{\mathbf{s}}_t-\mathbf{s}^\star_t\|_2^2}{\|\mathbf{s}^\star_t\|_2^2}. \tag{43}$$

We use CVX [36] as solver for the offline computations, and report the required computational time in seconds achieved by Algorithm 1 and CVX.

A. SYNTHETIC DATA

We generate a synthetic (seed) random graph $\mathbf{S}_0$ of N nodes using the GSP toolbox [37]. Then, edges abide two different temporal evolution patterns: *i) piecewise constant*; and *ii) smooth* temporal variation. Finally, we generate the stream of data according to the three considered models [cf. Section IV] for T time instants.

Piecewise. For the piecewise constant scenario, we randomly select $\lceil N/2\rceil$ nodes of the initial graph $\mathbf{S}_0$ and double the weight of their edges, after $T/2$ samples. Then, for $t=\{1,\ldots,T\}$ we generate each graph signal $\mathbf{x}_t$ according to the three models: 1) for the TV-GGM, we use $\mathbf{x}_t\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_t)$, where $\boldsymbol{\Sigma}_t=\mathbf{S}_t^{-1}$; 2) for the TV-SEM we use $\mathbf{x}_t=(\mathbf{I}-\mathbf{S}_t)^{-1}\mathbf{e}_t$ [cf. (4)], with noise variance $\sigma_e^2=0.5$; and 3) for the TV-SBM we use $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{L}_t^{\dagger} + \sigma_e^2 \mathbf{I}_N)$ as in [38] with $\sigma_e^2 = 0.5$.

Smooth. For the smooth scenario, starting from the initial graph $\mathbf{S}_0$, the evolution pattern follows an edge-dependent behavior, $S_t(i,j) = S_0(i,j)(1 + e^{-0.01ijt})$ for $t = \{1, \ldots, T\}$. This means that each edge follows an exponential decaying behavior, with the decaying factor depending on the edge itself. The data are generated as in the piecewise constant scenario.

For the results, we will compare the following methods:

- **Prediction-correction (PC)** *red curve:* this is the proposed Algorithm 1 specialized to one of the three models, with $P = C = 1$.
- **Correction-only (CO)** *cyan curve:* this is a prediction-free algorithm which only considers the original problem (17) and applies $C = 1$ iteration of the recursion (18). It is equivalent to Algorithm 1 with $P = 0, C = 1$. We consider this algorithm to study the benefits of the prediction step performed by PC.
- **Correction-correction (CC)** *blue curve:* this is a prediction-free algorithm which only considers the original problem (17) and applies $C = 2$ iterations of the recursion (18). It is equivalent to Algorithm 1 with $P = 0, C = 2$. This is a more fair comparison than CO, since the number of iterations is the same as the one of PC.
- **Stochastic gradient descent (SGD)** *ochre curve:* this is a prediction-free and memory-less version of the algorithm which only considers the last acquired graph signal. That is, the empirical covariance matrix $\hat{\mathbf{\Sigma}}_t = \mathbf{x}_t \mathbf{x}_t^{\top}$ in (19) is just a rank-one update, achieved by setting $\gamma = 0$. We consider this to show how much the temporal variability of the function, captured by the time-derivative of the gradient in PC, affects the algorithm's convergence.
- **Prediction-correction rank-one (PC-1)** *purple curve:* this is a rank-one (stochastic) implementation of the PC algorithm; i.e., $\hat{\mathbf{\Sigma}}_t = \mathbf{x}_t \mathbf{x}_t^{\top}$ for the update in (19), and $P = C = 1$. Notice that, differently from SGD, it also uses the time-derivative of the gradient, which in this case is the difference between two rank-one covariance matrices (thus the length of the memory is equal to one). We consider this algorithm to check the impact of the prediction step in a stochastic implementation of PC;
- **Correction-correction rank-one (CC-1)** *orange curve:* this is a rank-one (stochastic) implementation of the CC algorithm; i.e., it considers $\hat{\mathbf{\Sigma}}_t = \mathbf{x}_t \mathbf{x}_t^{\top}$ for the update in (19), and $P = 0, C = 2$. It can be seen as a two-step SGD, and we consider it to study whether the prediction step of PC-1 is beneficial for stochastic implementations.

In addition, for the piecewise constant scenario, we also report (green curve) the NSE between the PC solution and the batch solution obtained having all the relevant data in advance, i.e., the solution that would be obtained with a static graph learning algorithm on the intervals where the graph remains constant. In general, a fair comparison can be made within the rank-one implementations (SGD, PC-1 and CC-1) and within the memory-aware ones (PC, CO, CC).

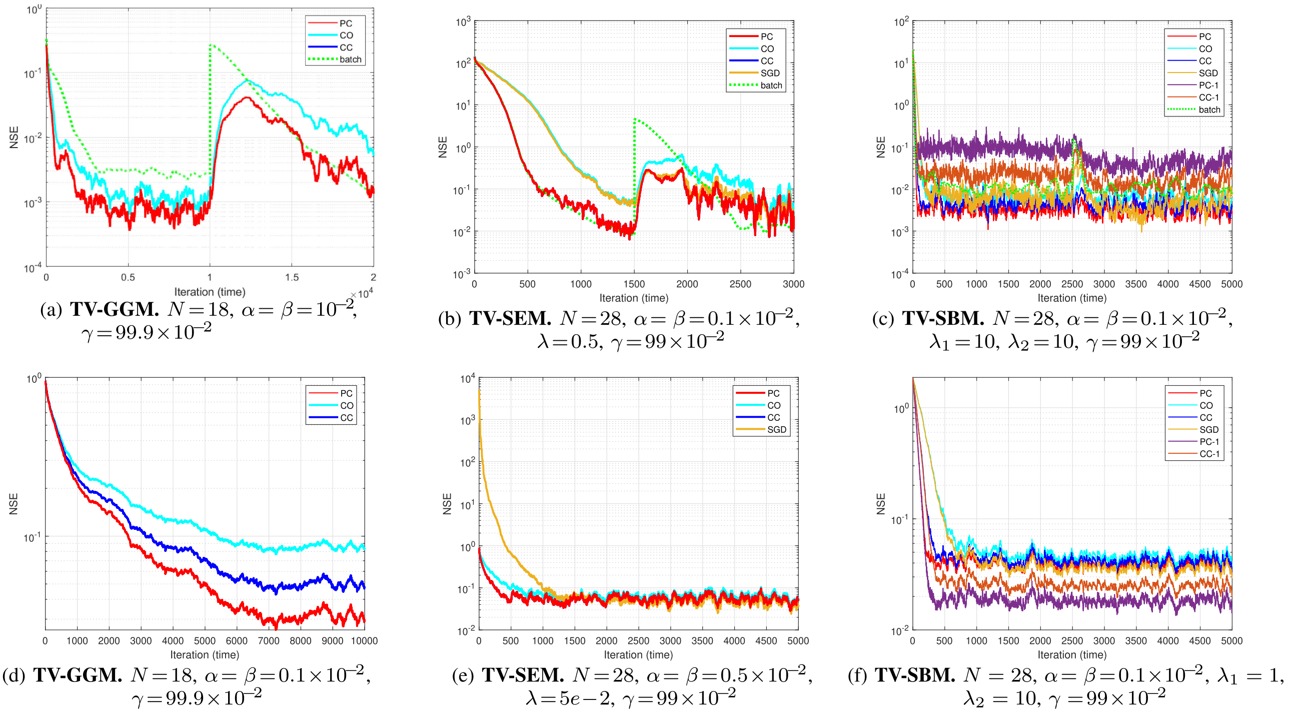
Results. The NSE achieved by Algorithm 1 for the three models is shown in Fig. 1, for both the piecewise constant (top row) and smooth (bottom row) scenarios. We use fixed step sizes for all the experiments. Notice that the only effect of the functions' hyperparameters is to shape the batch solution $\mathbf{s}_t^{\star}$ (and hence the time-varying trajectory $\hat{\mathbf{s}}_t$ at convergence). Thus, we run Algorithm 1 with different hyperparameters[4] and manually select them by ensuring that the trivial and complete graphs are excluded; the selected ones are displayed, together with the other algorithm's parameters, in the captions of Fig. 1.

GGM. Fig. 1(a) and Fig. 1(d) show the results for the piecewise constant and smooth scenarios, respectively. In both scenarios, the PC solution converges to the optimal offline counterpart and, for the piecewise constant, also to the batch solution(s). This demonstrates the adaptive nature of Algorithm 1 to react to changes in the data statistics. While for the piecewise constant scenario PC and CC offer the same convergence speed (which is expected, as explained in *"Does prediction help?"*), for the smooth scenario, the PC algorithm exhibits a faster convergence with respect to the prediction-free competitors CO and CC. This is because the temporal variability of the function (and of its gradient) is captured by the prediction step and exploited to fasten the convergence.

SEM. Similar considerations hold for the TV-SEM, whose results are illustrated in Fig. 1(b) and Fig. 1(e). In both scenarios, PC and CC offer the same convergence rate (which also converge to the batch solution for the constant scenario), faster than a CO and SGD implementation. Interestingly, after the triggering event at $T/2$, SGD can track the optimal solution faster than CO with performances similar to PC and CC. A possible justification may be the memory-less nature of SGD, i.e., it only considers the last sample for the gradient evaluation, thus discarding past data. This renders the SGD more reactive to adapt to sudden changes of the data statistics compared to the memory-aware alternatives, which however exhibit similar performances thanks to the extra iteration they can benefit.

SBM. Finally, the TV-SBM results are shown in Fig. 1(c) and Fig. 1(f). Also in this case, the PC solution converges to the offline counterpart for the two scenarios and faster than the prediction-free versions of the algorithm CC and CO. In particular, while in the piecewise constant scenario PC converges faster than CC and the rank-one implementations, in the smooth scenario the rank-one implementations exhibit faster convergent behavior with respect to the non-stochastic implementations. Similar to what has been said for the TV-SEM results, a possible reason can be the memory-aware characteristics of the non-stochastic methods; that is, while the information present in past data can be beneficial in the

[4] The search space intervals for the hyperparameters are the following: $\alpha, \beta \in (0.01, 1) \times 10^{-2}$, $\lambda \in (0.005, 5)$, $\lambda_1, \lambda_2 \in (1, 10)$, $\gamma \in \{97, 99, 99.9\} \times 10^{-2}$.



(a) **TV-GGM.** $N = 18$, $\alpha = \beta = 10^{-2}$, $\gamma = 99.9 \times 10^{-2}$

(b) **TV-SEM.** $N = 28$, $\alpha = \beta = 0.1 \times 10^{-2}$, $\lambda = 0.5$, $\gamma = 99 \times 10^{-2}$

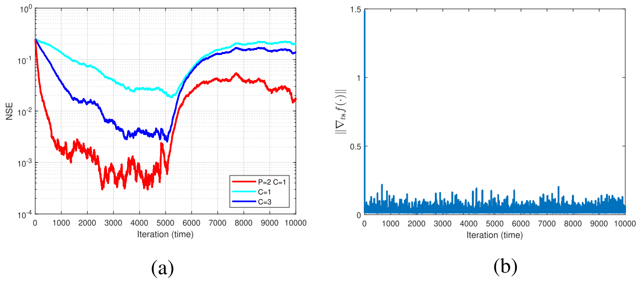
(c) **TV-SBM.** $N = 28$, $\alpha = \beta = 0.1 \times 10^{-2}$, $\lambda_1 = 10$, $\lambda_2 = 10$, $\gamma = 99 \times 10^{-2}$

(d) **TV-GGM.** $N = 18$, $\alpha = \beta = 0.1 \times 10^{-2}$, $\gamma = 99.9 \times 10^{-2}$

(e) **TV-SEM.** $N = 28$, $\alpha = \beta = 0.5 \times 10^{-2}$, $\lambda = 5e-2$, $\gamma = 99 \times 10^{-2}$

(f) **TV-SBM.** $N = 28$, $\alpha = \beta = 0.1 \times 10^{-2}$, $\lambda_1 = 1$, $\lambda_2 = 10$, $\gamma = 99 \times 10^{-2}$

FIGURE 1. Normalized squared error (NSE) for the piecewise-constant (top row) and smooth (bottom row) synthetic scenarios between our online solution $\hat{\mathbf{s}}_t$ (or the other variants reported in the legend) with respect to the offline solution $\mathbf{s}_t^\star$ obtained with CVX. For the piecewise-constant scenario, it is also illustrated the NSE between the PC solution and the batch solution (green curve). Stochastic implementations are available for a subset of methods due to numerical instabilities caused by the rank-one matrix operations involved.



(a)

(b)

FIGURE 2. (a) NSE of PC with $P = 2$ and $C = 1$, CO with $C = 1$ and CC with $C = 3$ for the piecewise constant scenario; (b) Norm of the time-derivative of the gradient as a function of the iteration index for the smooth scenario.

TABLE 1. Average Time (Expressed in Seconds) Required to Compute the PC and the CVX Solution at Each Time Instant

	PC	CVX
TV-GGM	0.110×10^{-2}	3.6
TV-SEM	0.824×10^{-2}	2.0
TV-SBM	0.023×10^{-2}	3.6

static scenario and thus help PC and CC to have a more reliable estimate of the true underlying (static) covariance matrix (and of the gradient), it may slow down the process in non-stationary environments with time-varying covariance matrices as in the smooth scenario.

Required time. An important metric to consider in time-sensitive applications is the average time per iteration. We report this information in Table 1, for the PC step and CVX, relative to the three considered models and settings in the top row of Fig. 1.

Combining the information of the table and that of the plots in Fig. 1, it is clear how trading off the knowledge of the optimal solution for savings in terms of time seems an excellent compromise. Each prediction-correction step requires indeed around three orders of magnitude less time than the CVX counterpart, leading to a NSE at least smaller than $10e - 1$.

Does prediction help? Notice how in the piecewise constant scenario, the PC strategy does not seem to offer a major advantage with respect the CC strategy. Although this behavior could be hypothesized (since the setting is static), it is here empirically confirmed. To can gain more insights we look at the structure of the prediction step (e.g., (23)), where the components playing a role in the descent direction are: the gradient $\nabla_{\mathbf{s}} f(\cdot)$; the Hessian $\nabla_{\mathbf{ss}} f(\cdot)$; and the time-derivative of the gradient $\nabla_{t\mathbf{s}} f(\cdot)$. Since we use $P = 1$, i.e., only one prediction step, the term $(\hat{\mathbf{s}}^p - \hat{\mathbf{s}}_t = \mathbf{0})$ that multiplies the Hessian does not contribute to the descent step. The added value of the prediction step with respect to a general (correction) descent method, in this case, would be only provided by the time-gradient $\nabla_{t\mathbf{s}} f(\cdot)$ (since the gradient $\nabla_{\mathbf{s}} f(\cdot)$ is common to either the prediction and the correction step). In the piecewise constant scenario, however, the underlying (true) covariance

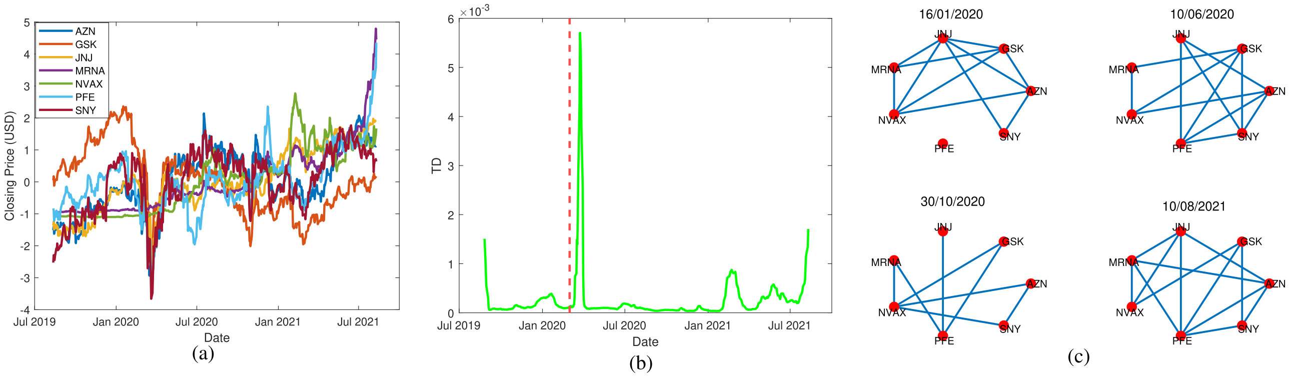


FIGURE 3. (a) Standardized time series for the period August 12th 2019 - August 10th 2021; (b) graph temporal deviation for the stock market graph inferred with TV-GGM. The sharp peaks around March 2020 and after January 2021 happen consistently with real events; (c) inferred topologies at four different dates of interest. The absence of an edge between two nodes indicates their conditional independence.

matrix is time-invariant within the two stationary intervals, leading to a zero time-derivative of the gradient (cf. (22)). This means that in static scenarios, with $P = 1$, the prediction step boils down to a correction step. Differently, for $P = 2$, the contribution of the second-order information may speed up the convergence, as illustrated in Fig. 2(a) for TV-GGM, with respect to a correction-only algorithm using $C = 3$.

In the smooth scenario, the temporal variability of the gradient captured by the time-derivative of the gradient $\nabla_{ts} f(\cdot)$, plays a role in the prediction step, which can improve the convergence speed of the algorithm. The (bounded) norm of this vector over time is illustrated in Fig. 2(b) for the TV-GGM smooth scenario of Fig. 1(d); this norm is linked to the constant C_0 introduced in Assumption 2 and the error in (40).

All in all, the results indicate the convergence of Algorithm 1 to the optimal offline counterpart and its capability to track it in non-stationary environments. The algorithm also converges to the batch solutions of the two stationary intervals, obtained with all the relevant data. A defining characteristic of Algorithm 1 is its ability to naturally enforce similar solutions at each iteration, achieved with an early stopping of the descent steps, governed by the parameters P and C. That is, the algorithm adds an implicit *temporal* regularization to the problem which needs to be explicitly added when working with the entire batch of data.

Given these results and insights, we can outline a few principles that can be adopted when considering Algorithm 1 for learning problems:

- The prediction step with $P = 1$ can be beneficial when the underlying data statistics change over time, so that the time-variability of the gradient can be exploited. Otherwise, in a complete static scenario, it coincides with a correction step.
- Increasing P can improve the convergence speed when the approximated cost function is a good surrogate of the cost function in the next time instant.
- Memory-less (stochastic) variants of the algorithm can be suitable in fast-changing environments, due to their ability to discard past information and react quickly to changes in data statistics.

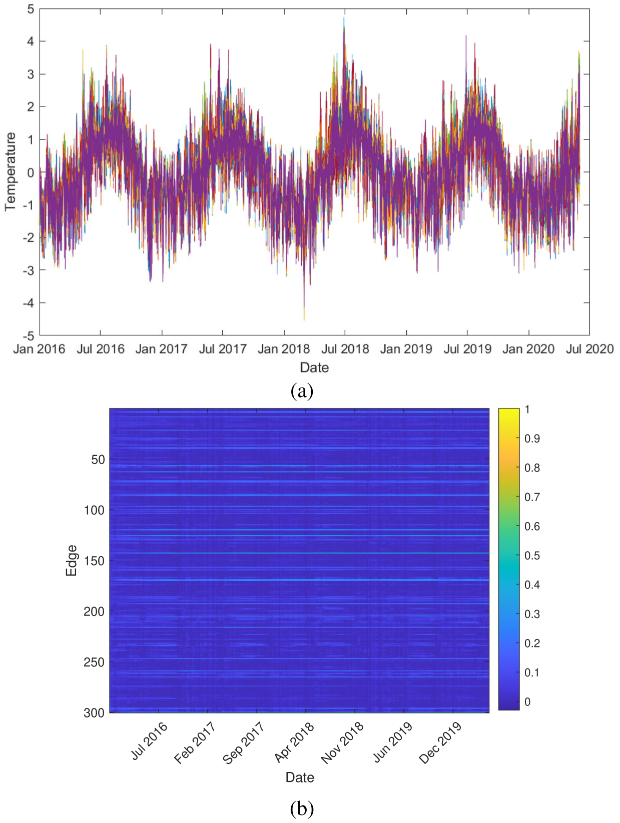
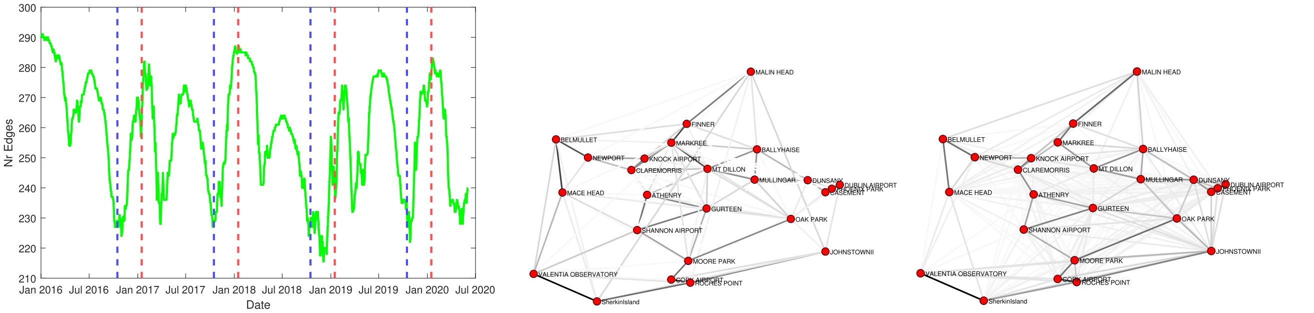


FIGURE 4. (a) Standardized time series for the 25 Irish weather stations and (b) evolution of each edge weight over time.

Being confident on the convergence of the algorithm, we now corroborate its performance with real data.

B. REAL DATA

We now test the three considered algorithms on real data. Among other indicators employed in the simulations to assess



(a) (b) Oct 2016 (c) Jan 2017

FIGURE 5. Ireland temperature dataset. (a) Number of edges of the inferred graph over time. The red vertical lines correspond to January 15 of each year (winter), while the blue vertical line correspond to October 15 (autumn); snapshot of the inferred time-varying graph during (b) October 2016 (autumn graph) and (c) January 2017 (winter graph). Notice how stations close in space tend to be connected.

the performance of the algorithm, we use the graph temporal deviation $\mathrm{TD}(t) := \|\hat{\mathbf{s}}_t - \hat{\mathbf{s}}_{t-1}\|_2$, which measures the global variability on the edges of the graph for different time instants. To gain further insights on the network evolution over time, we consider additional metrics (such as number of edges and temporal gradient norm) and visual analysis tools which will be introduced in the application-specific scenario at hand. In this case, the hyperparameters of each function are chosen in such a way that the inferred graphs are neither trivial nor complete, and interpretable patterns consistent with real events are visible from the plots of the employed metrics.

TV-GGM for Stock Price Data Analysis.

Data description: we collect historical stock (closing) prices relative to the S&P500 Index for seven pharmaceutical companies over the time period August 12th 2019 to August 10th 2021 using [39]. The collected data include the economic crisis related to the COVID-19 pandemic, followed by the vaccination campaign. The companies of interest are Pfizer (PFE), Astrazeneca (AZN), Johnson & Johnson (JNJ), GlaxoSmithKline (GSK), Moderna (MRNA), Novavax (NVAX) and Sanofi (SNY). Our goal is to leverage the TV-GGM in order to explore the relationships among these companies over time and observe the possible structural changes due to market instabilities.

Results: We consider $T = 504$ measurements (working days in August 2019 - August 2021) as graph signals $\{\mathbf{x}_t\}$ for the $N = 7$ quantities of interest, which are further standardized, i.e., each variable is centered and divided by its empirical standard deviation; see Fig. 3(a) for a plot of the standardized time series. We run the TV-GGM algorithm for different values of the forgetting factor γ, and monitor the evolution of the metrics earlier introduced. The value $\gamma = 0.75$ yielded results most consistent with the data behavior.

It is clear from Fig. 3(a) and the TD indicator in Fig. 3(b) that around March 2020 and after January 2021 the market has changed significantly, due to the instability generated by the pandemic and by the follow-up starting vaccination campaign. The sharp peaks in Fig. 3(b) around around the same period are a consequence of the dynamic inter-relationships among the companies; the inferred graph changes substantially in the two periods of interest and TD captures the market variability.

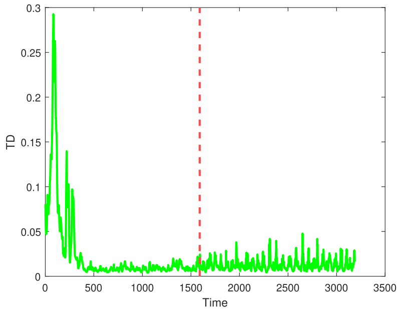


FIGURE 6. Graph temporal deviation for the epilepsy study. The red line indicates the seizure onset. During the ictal interval, a higher temporal deviation can be observed, indicating that the inferred graph is changing substantially.

To really enjoy the visualization potential offered by graphs as a tool, we show in Fig. 3(c) snapshots of the inferred time-varying graph at four different dates of interest. Common among the four graphs is the presence of the edge connecting MRNA and NVAX, and the edge connecting AZN and SNY. The pharmaceutical companies associated to the endpoints of each of these two edges also show a similar trend in Fig. 3(a). Notice moreover that since the sparsity pattern of the precision matrix reveals conditional independence among the variables indexed by its zero entries, these graphs enable us to visually inspect such independence over time. Although the information endowed in these graphs may carry a financial significance, we leave this possible knowledge-discovery task out of this manuscript, to avoid misleading or erroneous interpretations.

TV-SEM for Temperature Monitoring.

Data description: for this experiment we consider the publicly available weather dataset[5] provided by the Irish Meteorological Service, which contains hourly temperature (in

[5] [Online]. Available: https://www.met.ie/climate/available-data/historical-data

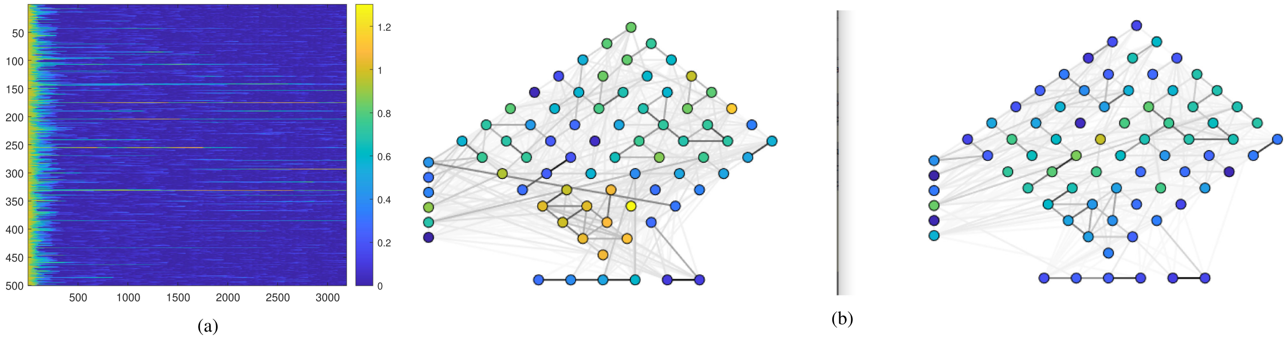


FIGURE 7. Epilepsy dataset. (a) Evolution of each edge weight over time; (b) snapshots of the inferred time-varying graph at time instant 1500 and 1800. The color of an edge indicates its weight, with darker colors indicating higher weights, while the color of a node indicates the closeness centrality of such node, with brighter colors indicating higher values of closeness centrality.

°C) data from 25 stations across Ireland. We monitor the temperature evolution over the sensor network for the period January 2016 to May 2020, and leverage the TV-SEM to infer the time-varying features of the graph learned by the algorithm.

Results: for the analysis we consider $T = 38713$ measurements as graph signals $\{\mathbf{x}_t\}$ for the $N = 25$ stations under consideration, standardizing the data as done in the previous experiment; Fig. 4(a) depicts the standardized time series. It is interesting to notice the sinusoidal-like behavior of the aggregate time-series, due to higher (lower) temperature during the summer (winter) period, resulting in a smooth signal profile.

Fig. 4(b) illustrates the sparsity pattern of the time-varying graph and the importance of the weights at every time instant. This *learn-and-show* feature offered by Algorithm 1 gives us the ability to visualize the learning behavior of the algorithm on-the-fly, a strength of low-cost iterative algorithms w.r.t. batch counterparts. From the figure (and the observed almost zero graph temporal deviation, which is not illustrated here) a consistent temporal homogeneity is visible, i.e., the graph does not change significantly over adjacent time instants. In other words, nodes influencing each other in a particular time instant, are likely to influence each other in other time instants. A reasonable explanation is given by the smooth and regular pattern exhibited by the time-series of Fig. 4(a), which is a consequence of the meteorological similarity over time, and by their high correlation coefficient.

An interesting trend arises when observing the number of edges of the graph inferred over time, shown in Fig. 5(a). Although in adjacent time instants the number does not change abruptly, a pattern can be identified over a longer time span. In particular, during winter and summer there is a sharp increment in the number of edges, with respect to autumn and spring where there is a significant reduction. To ease the visualization, the vertical red lines are placed in correspondence of the winter period of every year, while blue lines in correspondence of the autumn period. A possible reason for this phenomenon is given by the reduced variability of the temperature among the stations during summer and winter, and a higher variability during spring and autumn, leading to different graphs.

For the sake of visualization, we also report the inferred graphs for October 2016 (autumn) and January 2017 (winter). In line with our previous comments regarding Fig. 5(a), a lower number of edges is visible in the autumn graph with respect to the winter graph; in particular, edges present in the autumn graph are also present in the winter one. Finally, notice how stations close in space tend to be connected, thus showing how stations close to each other have a greater influence with respect to stations farther away in space.

TV-SBM for Epileptic Seizure Analysis.

Data description: we use electrocorticography (ECoG) time series collected during an epilepsy study at the University of California, San Francisco (UCSF) Epilepsy Center, where an 8×8 grid of electrodes was implanted on the cortical brain's surface of a 39-year-old woman with medically refractory complex partial seizure [40]. The grid was supplemented by two strips of six electrodes: one deeper implanted over the left suborbital frontal lobe and the other over the left hippocampal region, thus forming a network of 76 electrodes, all measuring the voltage level in proximity of the electrode, which is an evidence of the local brain activity. The sampling rate is 400 Hz and the measured time series contains the 10 seconds interval preceding the seizure (pre-ictal interval) and the 10 seconds interval after the start of the seizure (ictal interval). Our goal is to leverage the TV-SBM in order to explore the dynamics among different brain areas at the seizure onset.

Results: for our analysis we consider $T = 3200$ time instants as graphs signals $\{\mathbf{x}_t\}$ for the $N = 76$ electrodes, which are further filtered (over the temporal dimension) at $\{60, 180\}$Hz to remove the spurious power line frequencies, and standardized as explained in the previous experiments.

Fig. 6 shows the graph temporal deviation, where we observe an increasing and protracted variability of the TD shortly after the seizure onset (red vertical line), proving TD to serve as an indicator of network alteration suitable for time-varying scenarios. To visualize the on-the-fly learning behavior of the algorithm, in Fig. 7(a) we show the evolution of (a fraction[6] of) the edge weights over time. In the first half of the time-horizon, we notice the presence of stronger edges with respect to the second half, where the graph is sparser. We show two snapshots of the time-varying graph in Fig. 7(b), for the time instants 1500 (pre-ictal) and 1800 (ictal), where we also report the closeness centrality of each node, which expresses how "close" a node is to all other nodes in the network (calculated as the average of the shortest path length from the node to every other node in the network). During the ictal interval, the graph tends to be more disconnected and its nodes to have a lower closeness centrality value, especially in the lower part of the graph. In addition, we observe how the number of (strong) edges and the closeness centrality value drop in the ictal graph, especially in the lower part of the graph. This is consistent with the findings in [40] and indicates that, on average, signals in the pre-ictal interval behave more similar to each other as opposed to the signals in the ictal interval.

VII. CONCLUSION

In this manuscript, we proposed an algorithmic template to learn time-varying graphs from streaming data. The abstract time-varying graph learning problem, where the data influence is expressed through the empirical covariance matrix, is casted as a composite optimization problem, with different terms regulating different desiderata. The framework, which works in non-stationary environments, lies upon novel iterative time-varying optimization algorithms, which on one side exhibit an implicit temporal regularization of the solution(s), and on the other side accelerate the convergence speed by taking into account the time variability. We specialize the framework to the Gaussian graphical model, the structural equation model, and the smoothness-based model, and we propose ad-hoc vectorization schemes for structured matrices central for the gradient computations which also ease storage requirements. The proposed approach is accompanied by theoretical performance guarantees to track the optimal time-varying solution, and is further validated with synthetic numerical results. Finally, we learn time-varying graphs in the context of stock market, temperature monitoring, and epileptic seizures analysis. The current line of work can be enriched by specializing the framework to other static graph learning methods present in literature, possibly considering directed graphs, by implementing distributed versions of the optimization algorithms, and by applying the developed models in other real-world applications.

APPENDIX A

Consider the multi-valued function $\mathcal{T} : \mathbb{R}^N \rightarrow \mathbb{R}^N$, which we will refer to as *operator*. Here, we briefly review some operator theory concepts used in this manuscript; see [41].

Projection operator. Given a point $\mathbf{x} \in \mathbb{R}^N$, we define projection of $\mathbf{x}$ onto the convex set $\mathcal{C} \subseteq \mathbb{R}^N$ as:

$$\mathbb{P}_{\mathcal{C}}(\mathbf{x}) := \arg\min_{\mathbf{z}\in\mathcal{C}} \frac{1}{2}\|\mathbf{z}-\mathbf{x}\|_2 \tag{44}$$

Proximal operator. Consider the convex function $g : \mathbb{R}^N \rightarrow \mathbb{R}$. We define the proximal operator of $g(\cdot)$, with penalty parameter $\rho > 0$, as:

$$\text{prox}_{g,\rho}(\mathbf{x}) := \underset{\mathbf{z}}{argmin}\left\{g(\mathbf{z}) + \frac{1}{2\rho}\|\mathbf{z}-\mathbf{x}\|_2^2\right\} \tag{45}$$

For some functions, the proximal operator admits a closed form solution [35, Ch. 6]. In particular:

- if $g(\mathbf{x}) = \iota_{\mathcal{C}}(\mathbf{x})$ then $\text{prox}_g(\mathbf{x}) = \mathbb{P}_{\mathcal{C}}(\mathbf{x})$, i.e., it is the projection of $\mathbf{x}$ onto the convex set $\mathcal{C}$.
- if $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$ then $\text{prox}_g(\mathbf{x}) = \text{sign}(\mathbf{x}) \odot [\mathbf{x} - \lambda\mathbf{1}]_+$, i.e., it is the soft-thresholding operator.

Consider the convex minimization problem:

$$\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{x}) \tag{46}$$

with $f, g : \mathbb{R}^N \rightarrow \mathbb{R}$ convex. It can be shown that problem (46) admits at least one solution [42], which can be found by the fixed point equation:

$$\mathbf{x} = \text{prox}_{g,\rho}(\mathbf{x} - \rho\nabla f(\mathbf{x})) \tag{47}$$

APPENDIX B

Proof of Claim 1: TV-GGM: Recall the expression of the Hessian in (21b), i.e., $\mathbf{H}(\mathbf{S}) = \mathbf{D}^\top(\mathbf{S}\otimes\mathbf{S})^{-1}\mathbf{D}$ and that matrix $\mathbf{S} \in \mathcal{S}$ is the precision matrix, with $\mathcal{S} = \{\mathbf{S} \in \mathbb{S}^N_{++} | \xi\mathbf{I} \preceq \mathbf{S} \preceq \chi\mathbf{I}\}$. For the strong convexity, notice that since $\mathbf{S} \succ 0$, then also $\mathbf{H}(\mathbf{S}) \succ 0$. Indeed, by exploiting the semi-orthogonality of matrix $\mathbf{D}/\sqrt{2}$, we have:

$$\begin{aligned}
\lambda_{\min}(\mathbf{H}(\mathbf{S})) &= \min_{\|\mathbf{x}\|=1} \mathbf{x}^\top\mathbf{D}^\top(\mathbf{S}\otimes\mathbf{S})^{-1}\mathbf{D}\mathbf{x} \\
&\geq \min_{\|\mathbf{x}\|=1} \mathbf{x}^\top\frac{\mathbf{D}^\top}{\sqrt{2}}(\mathbf{S}\otimes\mathbf{S})^{-1}\frac{\mathbf{D}}{\sqrt{2}}\mathbf{x} = \min_{\|\mathbf{y}\|=1} \mathbf{y}^\top(\mathbf{S}\otimes\mathbf{S})^{-1}\mathbf{y} \\
&= \min_{\|\mathbf{z}\|=1} \sum_{i=1}^N\sum_{j=1}^N \frac{z_i z_j}{\lambda_i(\mathbf{S})\lambda_j(\mathbf{S})} \geq \frac{1}{\lambda^2_{\max}(\mathbf{S})} = 1/\chi^2
\end{aligned} \tag{48}$$

For the Lipschitz continuity of the gradient, we have

$$\begin{aligned}
\|\mathbf{D}^\top(\mathbf{S}\otimes\mathbf{S})^{-1}\mathbf{D}\| &\leq \|\mathbf{D}\|^2\|(\mathbf{S}\otimes\mathbf{S})^{-1}\| \\
&= 2\|(\mathbf{S}\otimes\mathbf{S})^{-1}\| = 2\|\mathbf{S}^{-1}\otimes\mathbf{S}^{-1}\| \\
&= 2\sqrt{\lambda_{\min}(\mathbf{S})^{-2}} = 2/\xi
\end{aligned} \tag{49}$$

■

Proof of Claim 2: TV-SEM

Denote with $\lambda_{\min}$ and $\lambda_{\max}$ the smallest and highest eigenvalues for the set of empirical covariance matrices obeying the SEM model. Recall the expression of the Hessian in (27b), i.e. $\mathbf{H}(\mathbf{S}; t) = \mathbf{Q}_t$, where $\mathbf{Q}_t := \mathbf{D}_h^\top(\hat{\mathbf{\Sigma}}_t \otimes \mathbf{I})\mathbf{D}_h$. Since $\mathbf{D}_h/\sqrt{2}$ is a

[6] For visualization, we show 500 random edges, since we recall that the number of total edges in an undirected graph of N nodes is $N(N-1)/2$.

semi-orthogonal matrix, we have:

$$\lambda_{\min}(\mathbf{H}(\mathbf{S})) = \min_{\|\mathbf{x}\|=1} \mathbf{x}^\top \mathbf{D}_h^\top \left(\hat{\boldsymbol{\Sigma}}_t \otimes \mathbf{I}\right) \mathbf{D}_h \mathbf{x}$$
$$\geq \min_{\|\mathbf{y}\|=1} \mathbf{y}^\top \left(\hat{\boldsymbol{\Sigma}}_t \otimes \mathbf{I}\right) \mathbf{y} = \min_{\|\mathbf{z}\|=1} \sum_{i=1}^{N} \lambda_i \left(\hat{\boldsymbol{\Sigma}}_t\right) z_i^2 \geq \lambda_{\min} \quad (50)$$

where $\lambda_{\min}$ is the smallest eigenvalue of $\hat{\boldsymbol{\Sigma}}_t$.

For the Lipschitz continuity of the gradient, we have:

$$\|\mathbf{D}_h^\top \left(\hat{\boldsymbol{\Sigma}}_t \otimes \mathbf{I}\right) \mathbf{D}_h\| \leq 2\|\hat{\boldsymbol{\Sigma}}_t \otimes \mathbf{I}\| = 2\lambda_{\max} \quad (51)$$

■

Proof of Claim 3: TV-SBM: For the strong convexity it suffices to notice that for $m > 0$, $f(\mathbf{s}; t) - \frac{m}{2}\|s\|^2 = \mathbf{s}^\top \mathbf{z}_t - \lambda_2 \mathbf{1}^\top \log(\mathbf{K}\mathbf{s}) + (\lambda_1 - \frac{m}{2})\|\mathbf{s}\|^2$ is convex. In turn, this implies that strong convexity of $f(\cdot; t)$ is guaranteed for $0 < m \leq 2\lambda_1$.

For the Lipschitz continuity of the gradient, recall that nodal degree vector $\mathbf{d} \succ 0$. Denote with $d_{\min}$ the minimum degree of the GSO search space. Also, recall the expression of the Hessian $\mathbf{H} = \mathbf{K}^\top \mathrm{Diag}(\mathbf{1} \oslash (\mathbf{K}\mathbf{s})^{\circ 2})\mathbf{K}$. Then:

$$\|\mathbf{K}^\top \mathrm{Diag}\left(\mathbf{1} \oslash (\mathbf{K}\mathbf{s})^{\circ 2}\right) \mathbf{K}\| \leq \|\mathbf{K}\|^2 \max\left(\mathbf{1} \oslash (\mathbf{K}\mathbf{s})^{\circ 2}\right)\Big)$$
$$= \|\mathbf{K}\|^2 d_{\min}^{-2} = 2(N-1)d_{\min}^{-2}, \quad (52)$$

where we made use of [19, Lemma 1] for the bound of $\mathbf{K}$. ■

APPENDIX C

The computational (arithmetic) complexity per iteration of Algorithm 1 is dominated by the rank-one covariance matrix update in $\mathcal{O}(N^2)$ and by the method-specific gradient computations involved in the prediction and correction steps (and eventually Hessian, if $P > 1$ [cf. Section VI "*Does prediction help?*"]). Such method-specific computational complexities are shown next, together with a discussion on the costs for the offline counterparts.

TV-GGM. The worst case scenario computational complexity of the gradient $\nabla_{\mathbf{s}} f(\mathbf{s}; t)$ in (21a) is $\mathcal{O}(N^3)$, which is due to the matrix inversion. This cost might be lowered exploiting the sparsity pattern of the sparse triangular factor of $\mathbf{S}$ or, in our case, exploiting the fact that it is a small perturbation with respect to the previous iterate. The multiplication with matrix $\mathbf{D}^\top$ has a cost of $\mathcal{O}(N^2)$, since $\mathbf{D} \in \mathbb{R}^{N^2 \times N(N+1)/2}$ has at most two 1's in each column and exactly one 1 in each row.

The worst case scenario computational complexity of the Hessian $\nabla_{\mathbf{ss}} f(\mathbf{s}; t)$ in (21b) would be $\mathcal{O}(N^3)$. However, because the Hessian is used in a matrix-vector multiplication [cf. (23)], its factorization leads to a cost for the prediction step of $\mathcal{O}(N^3)$. Indeed, exploiting the Kronecker product, the Hessian can be written as $\mathbf{D}^\top(\mathbf{S}^{-1} \otimes \mathbf{I}_N)(\mathbf{I}_N \otimes \mathbf{S}^{-1})\mathbf{D}$; then, the multiplication of the Hessian for a vector simply entails the succession of four sparse matrix-vector multiplications all with a cost of $\mathcal{O}(N^3)$.

The term $\nabla_{t\mathbf{s}} f(\mathbf{s}; t)$ in (22) has a computational complexity of $\mathcal{O}(N^2)$. Thus the overall computational complexity per iteration is $\mathcal{O}(N^3)$.

TV-SEM. The overall cost is dominated by the computation of $\mathbf{Q}_t = \mathbf{D}_h^\top(\hat{\boldsymbol{\Sigma}}_t \otimes \mathbf{I})\mathbf{D}_h$, which is present in the gradients and the Hessian. The matrix-matrix multiplication(s) have a cost of $\mathcal{O}(N^3)$, since $\mathbf{D}_h \in \mathbb{R}^{N^2 \times N(N-1)/2}$ has at most two 1's in each column and exactly one 1 in each row. Thus the overall computational complexity per iteration is $\mathcal{O}(N^3)$.

TV-SBM. Each column of $\mathbf{K}$ has exactly two non-zero entries (and each row has $N-1$ non-zero entries), thus $\mathbf{K}\mathbf{s}$ has a computational cost of $2|\mathcal{E}|$, with $|\mathcal{E}|$ the number of edges of the graph represented by $\mathbf{S}$ (in other words, $\|\mathbf{s}\|_0$). The operation $\mathbf{K}^\top(\mathbf{1} \oslash \mathbf{K}\mathbf{s})$ has a cost of $\mathcal{O}(N^2)$. The computational complexity of the Hessian is $\mathcal{O}(N^3)$, since it is the weighted sum of N outer products of vectors which are $(N-1)$-sparse in the same positions.

Thus the overall computational complexity per iteration is $\mathcal{O}(N^2)$ if $P = 0, 1$ and $\mathcal{O}(N^3)$ if $P > 1$.

Offline. The computational complexity for each time instant t incurred by an offline solver to solve instances of problem (13) depends by its algorithm-specific implementation closely related to the problem structure. The three problems we consider are (converted into) semidefinite programs (SDPs) and solved, in our case, by SDPT3, a Matlab implementation of infeasible primal-dual path-following algorithms, which involves the computation of second-order information. Since these computations are continuously repeated, for a fixed time instant t, till the algorithm convergence (say I iterations), a trivial lower bound for computing the offline solution for the three considered problems is $\boldsymbol{\Omega}(IN^3)$. To this cost must be also added the cost of other solver-specific steps which we do not explicitly consider here.

REFERENCES

[1] A. Natali, M. Coutino, E. Isufi, and G. Leus, "Online time-varying topology identification via prediction-correction algorithms," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 5400–5404.

[2] M. Coutino, E. Isufi, and G. Leus, "Advances in distributed graph filtering," *IEEE Trans. Signal Process.*, vol. 67, no. 9, pp. 2320–2333, May 2019.

[3] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 16–43, May 2019.

[4] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 44–63, May 2019.

[5] Y. Kim, S. Han, S. Choi, and D. Hwang, "Inference of dynamic networks using time-course data," *Brief. Bioinf.*, vol. 15, no. 2, pp. 212–228, 2014.

[6] R. N. Mantegna, "Hierarchical structure in financial markets," *Eur. Phys. J. B- Condens. Matter Complex Syst.*, vol. 11, no. 1, pp. 193–197, 1999.

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

[8] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 2826–2830.

[9] K. Yamada, Y. Tanaka, and A. Ortega, "Time-varying graph learning with constraints on graph temporal variation," 2020, *arXiv:2001.03346*.

[10] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, "Network inference via the time-varying graphical lasso," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 205–213.

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

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

[13] R. Money, J. Krishnan, and B. Beferull-Lozano, "Online non-linear topology identification from graph-connected time series," in *IEEE Data Sci. Learn. Workshop (DSLW)*, 2021, pp. 1–6, doi: 10.1109/DSLW51110.2021.9523399.

[14] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proc. IEEE*, vol. 106, no. 5, pp. 787–807, May 2018.

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

[16] R. Shafipour and G. Mateos, "Online topology inference from streaming stationary graph signals with partial connectivity information," *Algorithms*, vol. 13, no. 9, p. 228, 2020. [Online]. Available: https://doi.org/10.3390/a13090228

[17] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Trans. Signal Process.*, vol. 65, no. 22, pp. 5911–5926, Nov. 2017.

[18] B. Zaman, L. M. L. Ramos, D. Romero, and B. Beferull-Lozano, "Online topology identification from vector autoregressive time series," *IEEE Trans. Signal Process.*, vol. 69, pp. 210–225, 2021.

[19] S. S. Saboksayr, G. Mateos, and M. Cetin, "Online discriminative graph learning from multi-class smooth signals," *Signal Process.*, vol. 186, 2021, Art. no. 108101.

[20] A. Simonetto, E. Dall'Anese, S. Paternain, G. Leus, and G. B. Giannakis, "Time-varying convex optimization: Time-structured algorithms and applications," *Proc. IEEE*, vol. 108, no. 11, pp. 2032–2048, Nov. 2020.

[21] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[22] A. P. Dempster, "Covariance selection," *Biometrics*, vol. 28, no. 1, pp. 157–175, Mar. 1972.

[23] J. B. Ullman and P. M. Bentler, "Structural equation modeling," in *Handbook of Psychology: Research Methods in Psychology*, J. A. Schinka and W. F. Velicer, Eds., 2003, pp. 607–634.

[24] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. Artif. Intell. Statist.*, 2016, pp. 920–929.

[25] S. Kumar, J. Ying, J. V. de Miranda Cardoso, and D. P. Palomar, "A unified framework for structured graph learning via spectral constraints," *J. Mach. Learn. Res.*, vol. 21, no. 22, pp. 1–60, 2020.

[26] A. Simonetto, A. Mokhtari, A. Koppel, G. Leus, and A. Ribeiro, "A class of prediction-correction methods for time-varying convex optimization," *IEEE Trans. Signal Process.*, vol. 64, no. 17, pp. 4576–4591, Sep. 2016.

[27] B. Martinet, "Régularisation d'inéquations variationnelles par approximations successives. rev. française informat," *Recherche Opérationnelle*, vol. 4, pp. 154–158, 1970.

[28] J.-P. Vial, "Strong and weak convexity of sets and functions," *Math. Operations Res.*, vol. 8, no. 2, pp. 231–259, 1983.

[29] J. R. Magnus and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Hoboken, NJ, USA: Wiley, 2019.

[30] N. Bastianello, A. Simonetto, and R. Carli, "Primal and dual prediction-correction methods for time-varying convex optimization," 2020, *arXiv:2004.11709*.

[31] E. K. Ryu and S. Boyd, "Primer on monotone operator methods," *Appl. Comput. Math*, vol. 15, no. 1, pp. 3–43, 2016.

[32] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Berlin, Germany: Springer, 2011, pp. 185–212.

[33] X. Zhan, "Extremal eigenvalues of real symmetric matrices with entries in an interval," *SIAM J. Matrix Anal. Appl.*, vol. 27, no. 3, pp. 851–860, 2005.

[34] K. C. Das and R. Bapat, "A sharp upper bound on the spectral radius of weighted graphs," *Discrete Math.*, vol. 308, no. 15, pp. 3180–3186, 2008.

[35] A. Beck, *First-Order Methods in Optimization*. Philadelphia, PA, USA: SIAM, 2017.

[36] M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab software for disciplined convex programming, version 2.0 beta," Sep. 2013. [Online]. Available: http://cvxr.com/cvx

[37] N. Perraudin *et al.*, "GSPBOX: A toolbox for signal processing on graphs," Aug. 2014, *arXiv:1408.5781*.

[38] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, Dec. 2016.

[39] "Yahoo! finance," Accessed: Jul. 2021. [Online]. Available: https://finance.yahoo.com/lookup?s=API

[40] M. A. Kramer, E. D. Kolaczyk, and H. E. Kirsch, "Emergent network topology at seizure onset in humans," *Epilepsy Res.*, vol. 79, no. 2/3, pp. 173–186, 2008.

[41] H. H. Bauschke *et al.*, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Berlin, Germany: Springer, 2011.

[42] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Model. Simul.*, vol. 4, no. 4, pp. 1168–1200, 2005.