Online Graph Learning under Smoothness Priors

Seyed Saman Saboksayr*, Gonzalo Mateos*† and Mujdat Cetin*†
*Dept. of Electrical and Computer Engineering, University of Rochester, Rochester, NY, USA
†Goergen Institute for Data Science, University of Rochester, Rochester, NY, USA

***Abstract*—The growing success of graph signal processing (GSP) approaches relies heavily on prior identification of a graph over which network data admit certain regularity. However, adaptation to increasingly dynamic environments as well as demands for real-time processing of streaming data pose major challenges to this end. In this context, we develop novel algorithms for online network topology inference given streaming observations assumed to be smooth on the sought graph. Unlike existing batch algorithms, our goal is to track the (possibly) time-varying network topology while maintaining the memory and computational costs in check by processing graph signals sequentially-in-time. To recover the graph in an online fashion, we leverage proximal gradient (PG) methods to solve a judicious smoothness-regularized, time-varying optimization problem. Under mild technical conditions, we establish that the online graph learning algorithm converges to within a neighborhood of (i.e., it tracks) the optimal time-varying batch solution. Computer simulations using both synthetic and real financial market data illustrate the effectiveness of the proposed algorithm in adapting to streaming signals to track slowly-varying network connectivity.**

***Index Terms*—Graph learning, graph signal processing, online optimization, smooth signals, network topology inference.**

I. Introduction

Making sense of relational datasets from a network-centric perspective is essential to obtain actionable information in various fields of science and engineering. Graph signal processing (GSP) has played a key role to that end, underscoring the value of graphs as models of complex signals with irregular structures [1], [2]. However, said graphs are often not readily available and a first crucial step is to use nodal observations (i.e., measurements of graph signals) to identify the network structure, or, a useful graph model that facilitates signal representations and downstream learning tasks; see [3], [4] for tutorials on recent network topology inference advances. Acknowledging that many of these networks are also dynamic (e.g., in applications involving financial markets), there is a growing need to develop online graph learning algorithms that can process network data streams in an efficient manner [5].

Given a set of graph signal observations, the network topology inference problem is to find the graph (represented through e.g., an adjacency or Laplacian matrix) that is optimal in some sense. The optimality criterion is usually dictated by the adopted network-dependent model for the measurements, often augmented by structural (e.g., edge sparsity) priors motivated by physical characteristics to effect statistical regularization or favor interpretability. Network data models are often given by probabilistic priors such as Gaussian Markov random fields (GMRFs), where the graph learning problem becomes one of graphical model (here covariance) selection [2, Ch. 7]. Other models are specified via signal parsimony-related properties with respect to the underlying graph, including stationarity [6], [7] and smoothness (i.e., bandlimitedness, linked to GMRF selection under Laplacian constraints) [8]–[10].

Contributions in context of related prior work. In this paper, we develop an online algorithmic framework to estimate (possibly dynamic) graphs under smoothness priors. Many real-world signals are smooth over judicious networks, including temperature recordings [11], movie ratings, and natural images [12], to name a few. Exploitation of this cardinal property is at the heart of several graph-based statistical learning tasks including nearest-neighbor prediction (also known as graph smoothing), denoising, semi-supervised learning, and spectral clustering [1], [2]. Revisiting the general graph learning framework in [8], [10] – but with streaming data – we develop online proximal-gradient (PG) iterations to solve the resulting smoothness-regularized, time-varying optimization problem. There are noteworthy recent works on time-varying network topology inference from observations of smooth signals [12]–[14]. Unlike our online algorithm, those in [12]–[14] operate in batch mode, they are non-recursive, and hence their computational complexity and memory storage grow linearly with time. Borrowing techniques from [15], we establish that the online PG algorithm converges to within a neighborhood of (i.e., it tracks) the optimal time-varying batch solution. To the best of our knowledge, this is the first work that addresses the problem of online graph learning from streaming smooth signals with quantifiable performance guarantees. Different from our signal smoothness assumption, the online graph learning scheme in [16] uses observations from a Laplacian-based, continuous-time graph process and [7] relies on stationarity. Numerical tests using both synthetic and real financial market data corroborate the efficiency and effectiveness of the proposed PG algorithm in adapting to streaming signals and tracking changes in the sought (slowly-varying) dynamic network.

II. Graph Signal Processing Preliminaries

Consider a weighted, undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges, and $\mathbf{W} \in \mathbb{R}_+^{N \times N}$ is the symmetric adjacency matrix. In the absence of connection [i.e., $(i,j) \notin \mathcal{E}$] one has $W_{ij} = 0$. Graph $\mathcal{G}$ is devoid of self-loops, which implies

Work in this paper was supported by the NSF awards CCF-1750428, CCF-1934962 and ECCS-1809356. Author emails: ssaboksa@ur.rochester.edu, gmateosb@ece.rochester.edu, and mujdat.cetin@rochester.edu.

$W_{ii} = 0$, $\forall i \in \mathcal{V}$. We will henceforth assume nodal degrees $\mathbf{d} := \mathbf{W}\mathbf{1}$ are uniformly lower bounded away from zero, i.e., $\mathbf{d} \succeq d_{\min}\mathbf{1}$ (entry-wise inequality) for some prescribed $d_{\min} > 0$. Else if degrees become arbitrarily small, it is prudent to apply a threshold and remove the loosely connected nodes from $\mathcal{G}$. We study graph signals $\mathbf{x} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^N$ defined on $\mathcal{G}$, where x_i is the signal value at node $i \in \mathcal{V}$. Directed graphs could be useful models [17], but are beyond the scope of this paper. Complex-valued signals can be accommodated as well [1].

Signal smoothness with respect to $\mathcal{G}$. The adjacency matrix $\mathbf{W}$ encodes the graph's topology. Beyond $\mathbf{W}$, advances in spectral graph theory often motivate choosing the combinatorial graph Laplacian $\mathbf{L} := \text{diag}\,(\mathbf{d}) - \mathbf{W}$. In particular, $\mathbf{L}$ plays a central role in defining a graph Fourier transform (GFT) [1] as well as a measure of signal variability with respect to $\mathcal{G}$ [18]. Focusing on the latter, the total variation (TV) of the graph signal $\mathbf{x}$ with respect to the Laplacian $\mathbf{L}$ (also known as Dirichlet energy) is defined as the following quadratic form:

$$\text{TV}(\mathbf{x}) := \mathbf{x}^\top \mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{i \neq j} W_{ij}\left(x_i - x_j\right)^2. \tag{1}$$

The $\text{TV}(\mathbf{x})$ is a smoothness measure, quantifying how much the graph signal $\mathbf{x}$ changes with respect to $\mathcal{G}$'s topology. Smaller values of $\text{TV}(\mathbf{x})$ are indicative of limited signal variability, with $\text{TV}(\alpha\mathbf{1}) = 0$ as an extreme. More germane to the graph-learning theme of this paper is to use smoothness as the criterion to construct graphs on which network data admit certain regularity, the subject dealt with next.

III. Graph learning from smooth signals

Consider the following network topology identification problem. Given a set $\mathcal{X} := \{\mathbf{x}_t\}_{t=1}^T$ of possibly noisy graph signal observations acquired at time t, the goal is to learn an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{W})$ with $|\mathcal{V}| = N$ nodes such that the observations in $\mathcal{X}$ are smooth on $\mathcal{G}$. The graph can be dynamic with a slowly time-varying adjacency matrix $\mathbf{W}_t$, $t = 1, 2, \ldots$ (see Section IV), but for now we omit any form of temporal dependency to simplify exposition. In this section, we briefly review the general graph learning framework proposed in [8], [10], that we build on in the rest of the paper.

Graph learning under smoothness priors. Given $\mathcal{X}$ one can form the data matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T] \in \mathbb{R}^{N \times T}$, and let $\bar{\mathbf{x}}_i^\top \in \mathbb{R}^{1 \times T}$ denote its i-th row collecting those T measurements at vertex i. The neat idea in [8] is to establish a link between smoothness and sparsity, namely

$$\sum_{t=1}^{T} \text{TV}(\mathbf{x}_t) = \text{trace}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) = \frac{1}{2}\|\mathbf{W} \circ \mathbf{Z}\|_1, \tag{2}$$

where $\circ$ stands for the Hadamard (element-wise) product and the Euclidean-distance matrix $\mathbf{Z} \in \mathbb{R}_+^{N \times N}$ has entries $Z_{ij} := \|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\|^2$, $i, j \in \mathcal{V}$. The intuition is that when the given distances in $\mathbf{Z}$ come from a smooth manifold, the corresponding graph has a sparse edge set, with preference given to edges (i, j) associated with smaller distances Z_{ij}.

Leveraging (2) a general graph-learning framework was put forth in [8], which advocates solving the convex smoothness-regularized inverse problem

$$\begin{aligned} \min_{\mathbf{W}} \ & \|\mathbf{W} \circ \mathbf{Z}\|_1 + g(\mathbf{W}) \\ \text{s. t.} \quad & \text{diag}(\mathbf{W}) = \mathbf{0},\ W_{ij} = W_{ji} \geq 0,\ i \neq j. \end{aligned} \tag{3}$$

The convex objective function $g(\mathbf{W})$ encodes additional structural properties of $\mathcal{G}$. Several choices for $g(\mathbf{W})$ have been proposed to, e.g., recover graphs based on the Gaussian kernel [19], accommodate time-varying graphs [12], or to scale other related graph learning algorithms [9]. Identity (2) offers a favorable way of formulating the inverse problem (3), because the space of adjacency matrices can be described via simpler (meaning entry-wise decoupled) constraints relative to its Laplacian counterpart. As a result, (3) can be solved efficiently with complexity $\mathcal{O}(N^2)$ per iteration, by leveraging provably-convergent primal-dual solvers. Next, we present a different optimization approach based on PG methods [20], which offers an (equally) efficient alternative in the batch setting while it lends itself naturally to online operation.

Batch proximal gradient algorithm. To make (3) amenable to the PG method, recall first that the adjacency matrix $\mathbf{W}$ is symmetric with diagonal elements equal to zero. Thus, the independent decision variables are effectively the upper-triangular elements $[\mathbf{W}]_{ij}$, $j > i$, which we collect in the vector $\mathbf{w} \in \mathbb{R}_+^{N(N-1)/2}$. Second, it will prove convenient to enforce the non-negativity constraints via a penalty function augmenting the original objective. Just like [8] we add an indicator function $\mathbb{I}\{\mathbf{w} \succeq \mathbf{0}\} = 0$ if $\mathbf{w} \succeq \mathbf{0}$, and $\mathbb{I}\{\mathbf{w} \succeq \mathbf{0}\} = \infty$ otherwise. Given these definitions we recast the objective in (3) as the function $F(\mathbf{w})$ of a vector variable, and write the equivalent composite, non-smooth optimization problem

$$\min_{\mathbf{w}} F(\mathbf{w}) := \underbrace{\mathbb{I}\{\mathbf{w} \succeq \mathbf{0}\} + 2\mathbf{w}^\top \mathbf{z}}_{h(\mathbf{w})} + g(\mathbf{w}), \tag{4}$$

where $\mathbf{z}$ is a vector containing the upper-triangular entries of $\mathbf{Z}$, and $\mathbf{S} \in \{0, 1\}^{N \times N(N-1)/2}$ is such that $\mathbf{d} = \mathbf{W}\mathbf{1} = \mathbf{S}\mathbf{w}$.

A useful choice is $g(\mathbf{w}) = \beta 2\|\mathbf{w}\|^2 - \alpha\mathbf{1}^\top \log(\mathbf{S}\mathbf{w})$, where $\alpha, \beta > 0$ are tuning parameters [8]. The logarithmic barrier on the nodal degree sequence $\mathbf{S}\mathbf{w}$ precludes the trivial solution $\mathbf{w} = \mathbf{0}$. Moreover, it ensures the estimated graph is devoid of isolated vertices. The ℓ_2-norm regularization on the adjacency matrix $\mathbf{w}$ controls the graphs' edge sparsity pattern by penalizing larger edge weights (the sparsest graph is obtained for $\beta = 0$). The gradient of said g has the form

$$\nabla g(\mathbf{w}) = 4\beta\mathbf{w} - \alpha\mathbf{S}^\top\left(\frac{\mathbf{1}}{\mathbf{S}\mathbf{w}}\right), \tag{5}$$

where $\mathbf{1}/\mathbf{S}\mathbf{w}$ stands for element-wise division. Moreover, ∇g is a Lipschitz-continuous function with constant $\eta = \left(4\beta + \frac{2\alpha(N-1)}{d_{\min}^2}\right)$; see [21] for the proof that is omitted here due to lack of space.

For constant step size $\mu < \frac{2}{\eta}$, the PG iterations to solve the batch graph learning problem (4) are given by (henceforth $k = 0, 1, 2, \ldots$ denote iterations)

$$\mathbf{w}_{k+1} = \mathbf{prox}_{\mu h}\left(\mathbf{w}_k - \mu \nabla g(\mathbf{w}_k)\right), \tag{6}$$

where the proximal operator of h in (4) is

$$\mathbf{prox}_{\mu h}(\mathbf{w}) = \max\left(\mathbf{0}, \mathbf{w} - 2\mu \mathbf{z}\right). \tag{7}$$

The non-negative soft-thresholding operator in (7) sets to zero all edge weights in $\mathbf{w}$ that fall below the data-dependent thresholds in vector $2\mu\mathbf{z}$ (the max operator is applied entry-wise). Inspection of (6) shows that graph estimate refinements are generated via the composition of a gradient-descent step and a proximal operator.

All in all, (6) scales well to large graphs with thousands of nodes and it is competitive with the state-of-the-art primal-dual solver in [8]. In terms of convergence, as $k \to \infty$ the sequence of iterates (6) provably approaches a minimizer of the composite cost F in (4); see e.g., [20] for the technical details. Moreover, the worst-case convergence rate of PG algorithms is well documented (namely $\mathcal{O}(1/\epsilon)$ iteration complexity to return a ϵ-optimal solution measured in terms of F values), and can be boosted to $\mathcal{O}(1/\sqrt{\epsilon})$ via Nesterov-type acceleration techniques. Building on recent advances in time-varying convex optimization [15], our main contribution is to develop novel online PG algorithms for time-varying graphs in the previously unexplored streaming setting.

IV. Online Graph Learning

We switch gears to online estimation of $\mathbf{W}$ (or even tracking $\mathbf{W}_t$ in a dynamic setting) from streaming data $\{\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{x}_{t+1}, \ldots\}$. To this end, a viable approach is to solve at each time instant $t = 1, 2, \ldots$, the composite, time-varying optimization problem [cf. (4)]

$$\mathbf{w}_t^\star \in \operatorname*{argmin}_{\mathbf{w}} F_t(\mathbf{w}) := \overbrace{\mathbb{I}\{\mathbf{w} \succeq \mathbf{0}\} + 2\mathbf{w}^\top \mathbf{z}_{1:t}}^{h_t(\mathbf{w})} \underbrace{-\alpha \mathbf{1}^\top \log(\mathbf{S}\mathbf{w}) + \beta 2\|\mathbf{w}\|^2}_{g(\mathbf{w})}. \tag{8}$$

In writing $\mathbf{z}_{1:t}$ we make explicit that the Euclidean-distance matrix is computed using all signals acquired by time t. As data come in, the edge-wise ℓ_1-norm weights will fluctuate explaining the time dependence of $F_t(\mathbf{w})$ through its non-smooth component h_t.

Online algorithm construction. A naive sequential estimation approach consists of solving (8) repeatedly using the batch PG algorithm in Section III. However (pseudo) real-time operation in delay-sensitive applications may not tolerate running multiple inner PG iterations per time interval, so that convergence to $\mathbf{w}_t^\star$ is attained for each t. For time-varying graphs it may not be even prudent to obtain $\mathbf{w}_t^\star$ with high precision (hence incurring high delay and unnecessary computational cost), since at time $t+1$ a new datum arrives and the solution $\mathbf{w}_{t+1}^\star$ may be substantially off the prior estimate. These reasons motivate devising an efficient online and recursive algorithm to solve the time-varying optimization problem (8).

Algorithm 1: Online graph learning via PG

Input parameters α, β, γ, stream $\mathbf{z}_1, \mathbf{z}_2, \ldots$, initial $\mathbf{w}_1, \bar{\mathbf{z}}_0$.

for $t = 1, 2, \ldots,$ **do**

- Compute $\nabla g(\mathbf{w}_t) = 4\beta \mathbf{w}_t - \alpha \mathbf{S}^\top \left(\frac{\mathbf{1}}{\mathbf{S}\mathbf{w}_t}\right)$.
- Update $\bar{\mathbf{z}}_t = (1-\gamma)\bar{\mathbf{z}}_{t-1} + \gamma \mathbf{z}_t$.
- Update $\mu_t = \left(4\beta + \frac{2\alpha(N-1)}{\min(\mathbf{S}\mathbf{w}_t)^2}\right)^{-1}$.
- Update $\mathbf{w}_{t+1} = \max\left(\mathbf{0}, \mathbf{w}_t - \mu_t \nabla g(\mathbf{w}_t) - 2\mu_t \bar{\mathbf{z}}_t\right)$.

end

Our approach entails two steps per time instant $t = 1, 2, \ldots$. First, we recursively update the upper-triangular entries $\mathbf{z}_{1:t} = \mathbf{z}_{1:t-1} + \mathbf{z}_t$ of the Euclidean-distance matrix via an exponential moving average (EMA), namely

$$\bar{\mathbf{z}}_t = (1-\gamma)\bar{\mathbf{z}}_{t-1} + \gamma \mathbf{z}_t. \tag{9}$$

The constant $\gamma \in (0, 1)$ is a discount factor, which down-weighs past data to facilitate tracking dynamic graphs in non-stationary environments. The larger the constant γ, the faster EMA discounts past observations. Second, we run a single iteration of the batch graph learning algorithm developed in Section III to update $\mathbf{w}_{t+1}$, namely

$$\mathbf{w}_{t+1} = \mathbf{prox}_{\mu_t h_t}\left(\mathbf{w}_t - \mu_t \nabla g(\mathbf{w}_t)\right), \tag{10}$$

The resulting iterations are tabulated as Algorithm 1.

The computational complexity is dominated by the gradient evaluation in (5), incurring a cost of $\mathcal{O}(N^2)$ per instant t due to scaling and additions of vectors of length $N(N-1)/2$. For sparse graphs, the iterates $\mathbf{w}_t$ tend to become (and remain) quite sparse at early stages of the algorithm by virtue of the soft-thresholding operations (a sparse initialization $\mathbf{w}_1$ is preferable). It is thus possible to reduce the complexity further if Algorithm 1 is implemented carefully using sparse vector operations. Unlike recent approaches that learn dynamic graphs from the observation of smooth signals [12]–[14], Algorithm 1's memory storage requirement and computational cost per data sample $\mathbf{x}_t$ does not grow with t.

Convergence analysis. Here we establish that Algorithm 1 can closely track the sequence of minimizers $\mathbf{w}_t^\star$ for large enough t; see also the simulations in Section V. Noting that g is 4β-strongly convex and its gradient η-Lipschitz continuous, we can derive bounds for the tracking error $\|\mathbf{w}_t - \mathbf{w}_t^\star\|_F$. To this end, let us define $v_t := \|\mathbf{w}_{t+1}^\star - \mathbf{w}_t^\star\|_F$ to quantify the temporal variability of the optimal solution of (8). We have the following (non-asymptotic) performance guarantee for Algorithm 1, adapted from [15, Theorem 1].

Theorem 1 *For all $t \geq 1$, the sequence of iterates $\mathbf{w}_t$ generated by Algorithm 1 satisfies:*

$$\|\mathbf{w}_t - \mathbf{w}_t^\star\|_F \leq \tilde{L}_{t-1}\left(\|\mathbf{w}_0 - \mathbf{w}_0^\star\|_F + \sum_{\tau=0}^{t-1} \frac{v_\tau}{\tilde{L}_\tau}\right), \tag{11}$$

where $L_t = \max\{|1-4\mu_t\beta|, |1-\mu_t\eta_t|\}$, $\tilde{L}_t = \prod_{\tau=1}^{t} L_\tau$. *Moreover, for the sequence of objective values we can write* $F_t(\mathbf{w}_t) - F_t(\mathbf{w}_t^\star) \leq \frac{\eta_t}{2}\|\mathbf{w}_t - \mathbf{w}_t^\star\|_F$*; see [22, Theorem 10.29].*

To gain further insights let us define $\hat{L}_t := \max_{\tau=1,\ldots,t} L_\tau$, $\hat{v}_t := \max_{\tau=1,\ldots,t} v_\tau$. The sum of the geometric series in the right-hand side of (11) can be simplified to

$$\|\mathbf{w}_t - \mathbf{w}_t^\star\|_F \leq \left(\hat{L}_{t-1}\right)^t \|\mathbf{w}_0 - \mathbf{w}_0^\star\|_F + \frac{\hat{v}_t}{1-\hat{L}_{t-1}}.$$

Accordingly, $\hat{L}_t = (\eta_t - 4\beta)/\eta_t < 1$ since $\mu_t = \eta_t^{-1}$. Therefore, $(\hat{L}_{t-1})^t \to 0$ and Algorithm 1 converges to the vicinity of the optimal solution with a misadjustment $\hat{v}_t/(1-\hat{L}_{t-1})$. It follows that the tracking error increases with $\hat{v}_t$ (rapidly-varying graphs are more challenging) and also if the problem is badly conditioned (i.e., $\beta \to 0$ in which case $\hat{L}_t \to 1$).

V. Numerical Results

Here we test Algorithm 1 on synthetic and real-world financial market data. Throughout, we perform a grid search to determine the best regularization parameters α, β.

Synthetic data. To assess the performance of the proposed online graph learning algorithm, we test it on simulated streaming data. We generate a piecewise-constant sequence of two random Erdős-Rényi graphs (edge formation probability $p = 0.15$) with $N = 50$ nodes. The initial graph switches after $t = 4000$ time samples. The final graph is obtained from the initial one by redrawing 40 percent of its edges. For each time instant $t = 1, \ldots, 8000$, we simulate i.i.d. smooth signals with respect to the time-varying underlying graph. The signals are drawn from a Gaussian distribution $\mathbf{x}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{L}_t^\dagger + \sigma_e^2 \mathbf{I}_N\right)$, where σ_e represents the noise level; see e.g., [9]. Results are averaged over 10 independent Monte Carlo trials. The simulation is repeated for graphs with $N = 100$ nodes as well. As a baseline, we compute the time-varying solution F_t in (8) by running the batch PG algorithm until convergence.

Fig. 1 (top) shows that after around 1000 time samples (iterations) the objective value of the online Algorithm 1 reaches the optimal value of its time-varying batch counterpart. The observed oscillation depends on the step size μ_t in Algorithm 1. As expected, increasing the step size leads to faster convergence with larger oscillations. Conversely, reducing the step size leads to smoother and slower convergence. The other noteworthy observation is that the objective value markedly increases when the graph changes (after 4000 time samples), but the online algorithm can effectively track the dynamic graph after a sufficient number of samples have been acquired. A similar trend can be observed for the F-measure of the detected edges (defined as the harmonic mean between edge precision and recall); see Fig. 1 (bottom).

Financial market. Here, we perform a study involving real financial data. Since there is no ground-truth dynamic network, we endeavor to indirectly validate the proposed method by commenting on the intuitive structure observed from the learned sequence of graphs. We consider the daily stock prices for ten large American companies, including e.g., Microsoft

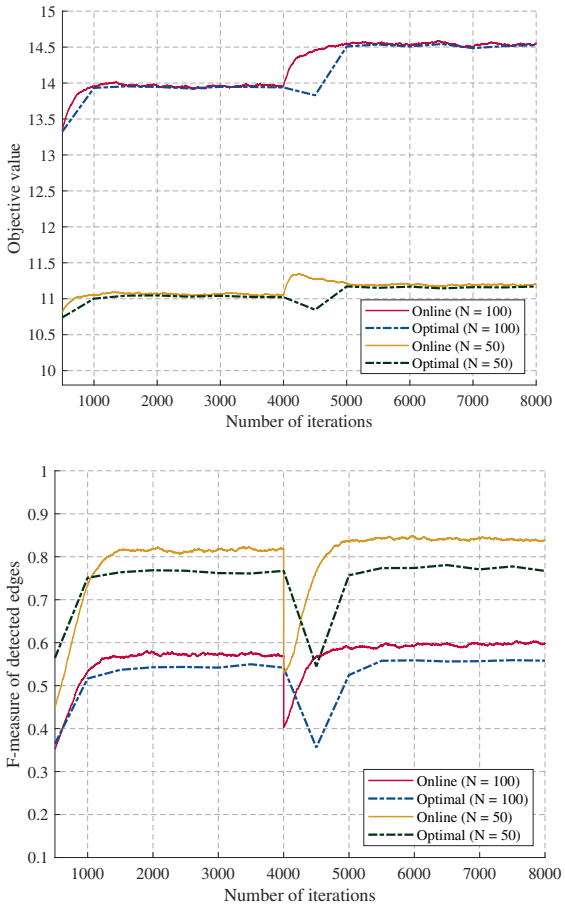

Fig. 1. Mean of objective value (8) (top), and F-measure of detected edges (bottom) as a function of acquired time samples (iteration). These results indicate that Algorithm 1 can effectively track its offline counterpart. Minor F-measure gaps are due to thresholding of edge weights.

(MSFT), Apple (AAPL) and Amazon (AMZN). We collect their daily stock prices from Yahoo! Finance over the time period from May 1st, 2019 to August 1st, 2020, which overlaps with the very recent COVID-19 pandemic that led to significant market instabilities. Under normal circumstances, we would expect limited variations in the network describing the pairwise relationships between the chosen stock prices, since these large companies are well-established in the market [23]. However, events like COVID-19 can cause abrupt changes in said network. We run Algorithm 1 to estimate daily graphs in order to monitor the sudden changes in the stock market. Following studies like [13], [23], we quantify the variation of the network via relative temporal deviation $\|\mathbf{W}_t - \mathbf{W}_{t-1}\|_F / \|\mathbf{W}_{t-1}\|_F$. We also learn networks from signals given by the relative temporal variation of stock prices.

In Fig. 2 (red) we plot the S&P500 log-price as an indicator of the market's condition. The relative temporal variation of the learned graphs is illustrated in Fig. 2 (blue), which are obtained by setting the regularization parameters as $\alpha = 0.316$, and $\beta = 0.05$. Fig. 2 (red) shows that the COVID-19 impacts on the markets started in February 2020 and it got to its worse situation during March 2020. The following sudden changes

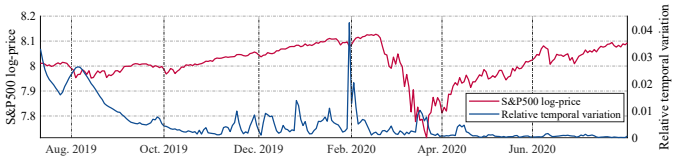

Fig. 2. The S&P500 log-price per day (red). The daily relative temporal variation of the learned graphs (blue). The temporal variation indicates some sudden changes which can be due to e.g., COVID-induced market tensions.

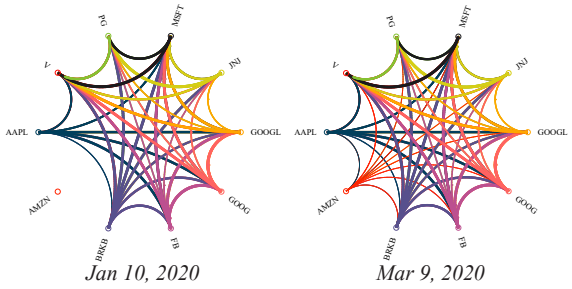

Fig. 3. The estimated network of the market over two different days: January 10th, 2020, and March 9th, 2020. Graph connectivity increases at the time of crisis when all the stock prices usually drop (right). While in stable times (left), the graph tends to be more loosely connected.

are apparent by inspection of the spikes in Fig. 2 (blue): (i) Sep'19, (ii) Nov'19, (iii) Dec'19, (iv) Jan'20, (v) Mar'20, and (vi) Apr'20. These abrupt changes are consistent with major events occurring during these time periods. In Sep'19 there was a congressional hearing regarding the impeachment inquiry of President Trump. The changes in Nov'19 can be explained by the optimism surrounding the U.S.-China trade negotiations. The U.S. House of Representatives' voted to impeach President Trump and that caused another sudden change in Dec'19. The big spike at the end of Jan'20 is probably the result of the World Health Organization (WHO) declaring a global health emergency due to the COVID-19 pandemic. In the middle of Mar'20, President Trump declared a national emergency due to COVID-19. The epidemic appears to continue (adversely) affecting the market well into Apr'20, when the daily case counts reached record values of 30000.

We also learned networks from graph signals representing the relative temporal variation (single-day discrete gradients) of the stock prices, instead of the raw daily values considered so far. We set $\alpha = 3.162$, and $\beta = 0.088$. Two instances of the learned graphs (Jan. 10, 2020, and Mar. 9, 2020) are depicted in Fig. 3. The stock prices dropped sharply during the economic crisis in Mar'20; see Fig. 2 (red). This has an equalization effect on the stock price gradients, which suddenly become negative for all nodes (companies). Therefore, the Mar. 9 graph becomes more connected as a consequence of the imposed smoothness prior. However, during relatively normal operation of the market (e.g, during early Jan'20) the temporal stock-price gradients will be less-tightly coupled (possibly fluctuating up or down depending on some other latent effects). Accordingly, the learned graph tends to be relatively more disconnected as seen in Fig. 3 (left).

VI. Conclusion

In this paper, we developed a novel online algorithm to learn graphs from streaming smooth signals. We exploited recent advances in time-varying convex optimization to derive efficient iterations that are guaranteed to track the (slowly) time-varying optimal solution of the batch estimator. The tracking ability of the proposed method is corroborated using synthetic and real-world financial market data.

References

[1] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[2] E. D. Kolaczyk, *Statistical Analysis of Network Data: Methods and Models*. New York, NY: Springer-Verlag, 2009.

[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, 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, 2019.

[5] 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, 2018.

[6] S. Segarra, A. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 467–483, Aug. 2017.

[7] R. Shafipour and G. Mateos, "Online topology inference from streaming stationary graph signals with partial connectivity information," *Algorithms*, vol. 13, no. 9, pp. 1–19, Sep. 2020.

[8] V. Kalofolias, "How to learn a graph from smooth signals," in *Artif. Intel. and Stat. (AISTATS)*, 2016, pp. 920–929.

[9] 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, 2016.

[10] V. Kalofolias and N. Perraudin, "Large scale graph learning from smooth signals," in *Int. Conf. Learning Representations (ICLR)*, 2019.

[11] S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero, "Learning sparse graphs under smoothness prior," in *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*, 2017, pp. 6508–6512.

[12] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*, 2017, pp. 2826–2830.

[13] J. V. d. M. Cardoso and D. P. Palomar, "Learning undirected graphs in financial markets," *arXiv preprint arXiv:2005.09958*, 2020.

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

[15] L. Madden, S. Becker, and E. Dall'Anese, "Online sparse subspace clustering," in *IEEE Data Sci. Wrksp.*, 2019, pp. 248–252.

[16] S. Vlaski, H. P. Maretić, R. Nassif, P. Frossard, and A. H. Sayed, "Online graph learning from sequential data," in *IEEE Data Sci. Wrksp.*, 2018.

[17] A. G. Marques, S. Segarra, and G. Mateos, "Signal processing on directed graphs," *arXiv preprint arXiv:2008.00586 [eess.SP]*, 2020.

[18] D. Zhou and B. Schölkopf, "A regularization framework for learning from graph data," in *Int. Conf. Mach. Learning (ICML)*, 2004.

[19] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. New York: Springer, 2009.

[20] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, p. 127–239, 2014.

[21] S. S. Saboksayr, G. Mateos, and M. Cetin, "Online discriminative graph learning from multi-class smooth signals," *Signal Processing*, vol. 186, p. 108101, 2021.

[22] A. Beck, *First-order Methods in Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2018.

[23] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, "Network inference via the time-varying Graphical Lasso," in *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 205–213.