Topology-Aware Joint Graph Filter and Edge Weight Identification for Network Processes

Natali, Albero; Coutino, Mario; Leus, Geert

DOI
10.1109/MLSP49062.2020.9231913

Publication date
2020

Document Version
Final published version

Published in
2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)

2020 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2020, ESPOO, FINLAND

TOPOLOGY-AWARE JOINT GRAPH FILTER AND EDGE WEIGHT IDENTIFICATION FOR NETWORK PROCESSES

Alberto Natali, Mario Coutino and Geert Leus

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, Delft, The Netherlands
E-mails: {a.natali; m.a.coutinominguez; g.j.t.leus}@tudelft.nl

ABSTRACT

Data defined over a network have been successfully modelled by means of graph filters. However, although in many scenarios the connectivity of the network is known, e.g., smart grids, social networks, etc., the lack of well-defined interaction weights hinders the ability to model the observed networked data using graph filters. Therefore, in this paper, we focus on the joint identification of coefficients and graph weights defining the graph filter that best models the observed input/output network data.

While these two problems have been mostly addressed separately, we here propose an iterative method that exploits the knowledge of the support of the graph for the joint identification of graph filter coefficients and edge weights. We further show that our iterative scheme guarantees a non-increasing cost at every iteration, ensuring a globally-convergent behavior. Numerical experiments confirm the applicability of our proposed approach.

Index Terms— Filtering over graphs, graph signal processing, graph filter identification, networked data modeling, topology identification

1. INTRODUCTION

The increasing amount of *networked* data, also conceptualized as graph signals within the graph signal processing (GSP) field [1] [2], has gained a lot of attention in the scientific community. Due to this, many signal processing tasks have been adapted towards their networked counterpart, as extensively detailed in [3].

In the graph setting, it is common to parameterize network processes through graph filters, due to their versatility and their natural distributed implementation [4] [5]. They play an important role within GSP, with applications ranging from reconstruction [6] [7] [8], denoising [9] and classification [10], to forecasting [11] [12] and (graph-)convolutional neural networks [13]. Notable recent advances in such structures are [14], which generalizes state-of-the-art graph filters to filters where every node weights the signal of its neighbors with different values, and [15], which extends the classical problem of blind system identification or blind de-convolution to the graph setting.

Given the structure of the graph, encoded by the so-called graph shift operator (GSO) [2], and assuming a process modelled by a graph filter, identifying an underlying network process from input/output networked data amounts to estimate the graph filter coefficients, thus alleviating the estimation workload [2] [16]. A key assumption in graph filtering is the *knowledge* of the GSO, which can be obtained from some other field of research or can be estimated from historical data. The latter relates to network topology inference or graph learning which, in recent years, has experienced an exponentially-increasing scientific interest, see, e.g., [17] [18] [19].

Related to the scenario we are going to consider, there are also works that model the observed signal as the output of an unknown graph filter over an unknown graph. In [20], a two-step GSO identification approach is taken, where first the GSO's eigenvectors are identified from the diffused (stationary) graph signals and then the GSO's eigenvalues are estimated based on some general properties of the GSO. In [21], the work of [20] is extended to non-stationary graph signals, entailing the solution of a system of quadratic matrix equations. Using the same approach, the problem of directed network topology identification is investigated in [22]. Note, though, that none of these above works focuses on estimating the related graph filter. More similar to our work, is the approach of [23], where not only the GSO but also the filter taps are learned. Although the context of [23] is different, in that work, a general linear filter operator is estimated from the data and then both the GSO and the filter taps are estimated from it.

All the previous approaches rely on a multi-step algorithm and only exploit some general properties of the GSO, e.g., sparsity. In addition, in many practical networks such as social and supply networks, the support of the graph is a priori known, that is, the connections between different entities of the network are already known, yet their importance might be unknown. And this information is not directly handled by the above algorithms.

Motivated by the above reasons, this work aims to jointly estimate the graph filter coefficients and the weights of the network topology. This joint approach leads to an optimization problem that is non-convex. We tackle the non-convexity of the problem by building on sequential convex programming (SCP), a local optimization tool for non-convex problems that leverages the convex optimization machinery. We show that an alternating minimization between the filter coefficients and the GSO guarantees that the objective function value at each iteration is non-increasing, obtaining a globally convergent method.

2. PRELIMINARIES

In this section, we introduce the GSP background material necessary for the rest of the paper, including the formal definition of graph signals and the core concepts of graph filtering and topology identification.

Graph Signal Processing We consider the case in which the data of interest live in a non-Euclidean domain, described by the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{S})$, where $\mathcal{V} = \{1, \dots, N\}$ is the set of nodes (or vertices), $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{S}$

978-1-7281-6662-9/20/$31.00

is a symmetric $N \times N$ matrix that represents the graph structure. The matrix $\mathbf{S}$ is called the graph shift operator (GSO) [2], whose entries $[\mathbf{S}]_{ij}$ for $i \neq j$ are different from zero only if nodes i and j are connected by an edge. Typical choices of the GSO include the (weighted) adjacency matrix $\mathbf{W}$ [2] and the graph Laplacian $\mathbf{L}$ [1].

This allows us to define a graph signal, denoted by the vector $\boldsymbol{x} \in \mathbb{R}^N$, as a mapping from the node set to the set of real vectors; that is, $\boldsymbol{x} : \mathcal{V} \to \mathbb{R}^N$. In this way, $x_i \in \mathbb{R}$ is a scalar that represents the signal value at node i. Because $\mathbf{S}$ reflects the local connectivity of $\mathcal{G}$, the operation $\mathbf{S}\boldsymbol{x}$ performs at each node a local computation enabling us to introduce the concept of filtering in the graph setting.

Graph Filters We can process a graph signal $\boldsymbol{x}$ by means of a so-called graph filter [2] as:

$$\boldsymbol{y} = \mathbf{H}(\boldsymbol{h}, \mathbf{S})\, \boldsymbol{x} = \sum_{k=0}^{K} h_k \mathbf{S}^k \boldsymbol{x}, \tag{1}$$

where K is the order of the filter, $\mathbf{H}(\boldsymbol{h}, \mathbf{S})$ is a polynomial matrix on $\mathbf{S}$ and $\boldsymbol{h} := [h_0, \ldots, h_K]^\top$ is the vector that contains the filter taps. Due the locality of $\mathbf{S}$, graph filters represent linear transformations that can be implemented in a distributed setting [20]. More formally, the output entry y_i of $\boldsymbol{y}$ at node i is a linear combination of $K+1$ terms: the first term is the signal value x_i of node i; the kth term ($k = 1, 2, \ldots, K$) combines signal values x_j from the k-hop neighbors of node i.

Topology Identification When the connections of the network cannot be directly observed or the network is just a conceptual model of pair-wise relationships among entities, a fundamental question is how to learn its structure from the graph signals. Formally, consider the matrix $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T] \in \mathbb{R}^{N \times T}$ that stacks column-wise T graph signals $\boldsymbol{x}_t$ residing over the network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{S})$. The goal is to infer the *latent* underlying network topology encoded in the GSO $\mathbf{S}$ under some optimality criterion.

This problem has been addressed in the past by means of statistical approaches, mostly based on correlation analysis and its connections to covariance selection and high-dimensional regression for learning Gaussian graphical models. Only more recently, GSP postulated the network topology inference problem under the assumption that the observed signals exhibit certain properties over the graph, such as smoothness, stationarity or band-limitedness. The reader interested in this topic is referred to [17] [18] [19].

Differently from the traditional topology identification setting, instead of estimating $\mathbf{S}$ from $\mathbf{X}$, we rely on model (1) and focus on a problem where given input and output data, the values of the nonzero entries of $\mathbf{S}$, i.e., the edge weights, and the filter taps $\boldsymbol{h}$ of a graph filter $\mathbf{H}(\boldsymbol{h}, \mathbf{S})$ have to be jointly identified. In Section 3, we rigorously formulate this problem and, in Section 4, we propose a way to efficiently tackle it.

3. JOINT GRAPH FILTER AND TOPOLOGY ESTIMATION

Suppose there is an unknown network process that can be accurately modelled by a graph filter $\mathbf{H}(\boldsymbol{h}, \mathbf{S})$ where, in response to an input $\boldsymbol{x}_t$, we observe a corresponding output $\boldsymbol{y}_t$. Such dynamics can be found for instance in social networks, where as a result of an advertisement campaign, we may expect to observe a response of the network's users; or in epidemics, where the nodes of the network are cities and we monitor the evolution of a spreading disease from one time instant to the next.

Let us assume that there are T input-output pairs available, and that we stack them column-wise in the matrices $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T]$ and $\mathbf{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T]$, respectively. Let the unknown filter $\mathbf{H}(\boldsymbol{h}, \mathbf{S})$ be of the form in (1). At this point, we are ready to formally state the problem we are going to address.

Problem Statement *Given the input-output data $\{\boldsymbol{x}_t, \boldsymbol{y}_t\}_{t=1}^T$ and the support, $\mathcal{A}$, of the graph $\mathcal{G}$, the goal is to identify the filter coefficients $\boldsymbol{h}$ and the GSO $\mathbf{S}$ embodied in the graph filter $\mathbf{H}(\boldsymbol{h}, \mathbf{S})$, that maps $\boldsymbol{x}_t$ into $\boldsymbol{y}_t$ as accurately as possible.*

The above problem can be mathematically defined with a least-squares formulation as:

$$\begin{array}{ll} \underset{\boldsymbol{h}, \mathbf{S}}{\operatorname{argmin}} & \|\mathbf{Y} - \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{X}\|_{\mathrm{F}}^2 \\ \text{s.t.} & \mathbf{S} \in \mathcal{S} \\ & \operatorname{supp}(\mathbf{S}) \subseteq \mathcal{A} \end{array} \tag{2}$$

where $\mathcal{S}$ represents the set of valid GSOs, $\mathcal{A}$ denotes the set with the support of $\mathcal{G}$, and $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius matrix norm. Note the (relaxed) constraint on the support: as the sparsity pattern of the GSO might have been overestimated, we leave it to the algorithm to optimize it, eventually shrinking to zero some unnecessary edges. That is, we constrain only the entries of the GSO to be zero in correspondence to the zeros of the support, leaving the other entries unconstrained (both zero and non-zero values are admitted).

From (2), we can deduce that the problem is not convex. Indeed, the objective function is made up of cross-products between the entries of $\mathbf{S}$ and the filter coefficients h_k, and by the power terms $\mathbf{S}^k$. The overall optimization problem is hence not convex and traditional tools of convex optimization cannot be used.

Although not directly handling the fixed-support case, the works referenced in Section 1 address the estimation problem using multi-step approaches to find $\mathbf{S}$ and/or $\boldsymbol{h}$. For instance, in [23] each realization is modeled through a graph filter-based vector auto-regressive (VAR) model, and this structure is leveraged to first recover the graph filters $\mathbf{H}_i(\boldsymbol{h}, \mathbf{S})$ representing the matrix filter taps of the VAR, and only then to recover the shift $\mathbf{S}$ and the coefficients $\boldsymbol{h}$ from them. Other approaches, such as [20] [21] are only interested in learning the shift $\mathbf{S}$, while others, such as [15], only in the filter coefficients $\boldsymbol{h}$.

Differently from the method in [23], in the following, we introduce a globally convergent SCP-based method to directly find both the filter taps $\boldsymbol{h}$ and the GSO $\mathbf{S}$. To the best of our knowledge, this is the first work that jointly learns the filter taps and the graph topology from observations.

4. ALTERNATING MINIMIZATION

To tackle the non-convexity of the problem and to bypass the limited flexibility of other methods, we resort to the alternating minimization (AM) approach, acting iteratively on $\boldsymbol{h}$ and $\mathbf{S}$. The general AM pseudo-code, adapted to our case, is reported in Algorithm 1. Notice that due to steps 3 and 4 in Algorithm 1, the cost is guaranteed to be a non-increasing function of the iteration number. In the following, we show how to perform step 3 and 4 of the proposed algorithm.

Given the estimate of the GSO $\mathbf{S}$ at the $(n-1)$th iteration, i.e., $\mathbf{S}^{(n-1)}$, the estimation problem at the nth iteration for the filter taps vector $\boldsymbol{h}$, i.e., $\boldsymbol{h}^{(n)}$, reads as:

$$\boldsymbol{h}^{(n)} = \underset{\boldsymbol{h}}{\operatorname{argmin}} \|\mathbf{Y} - \sum_{k=0}^{K} h_k \big(\mathbf{S}^{(n-1)}\big)^k \mathbf{X}\|_{\mathrm{F}}^2. \tag{3}$$

Problem (3) is convex and boils down to the traditional linear least squares (LLS) problem.

Algorithm 1 Joint GF & GSO Identification

Require: Feasible $\mathbf{S}^{(0)}$, $\varepsilon > 0$, $\mathcal{A}$, $\mathcal{S}$
1: $n \leftarrow 1$
2: **while** not converged **do**
3: $\quad \boldsymbol{h}^{(n)} \leftarrow \text{argmin}_{\mathbf{h}}\, f\left(\boldsymbol{h}, \mathbf{S}^{(n-1)}\right)$ [See Eq. (3)]
4: $\quad \mathbf{S}^{(n)} \leftarrow \text{argmin}_{\mathbf{S}}\, f\left(\boldsymbol{h}^{(n)}, \mathbf{S}\right)$ (**SCP**) [See Alg. 2]
5: $\quad$ Check convergence $(\boldsymbol{h}^{(n)}, \mathbf{S}^{(n)}, \varepsilon)$
6: $\quad n \leftarrow n + 1$
7: **end while**
8: **return** $\mathbf{S}^{(n)}, \boldsymbol{h}^{(n)}$

The solution of (3) is then used in the next step, i.e., step 4, to minimize the function with respect to the constrained GSO $\mathbf{S}$; that is

$$\begin{aligned} \mathbf{S}^{(n)} \quad = \quad & \underset{\mathbf{S}}{\text{argmin}} \quad \{f(\mathbf{S}) := \|\mathbf{Y} - \textstyle\sum_{k=0}^{K} h_k^{(n)} \mathbf{S}^k \mathbf{X}\|_{\text{F}}^2\} \\ & \text{s.t.} \quad \mathbf{S} \in \mathcal{S} \\ & \quad\;\; \text{supp}\,(\mathbf{S}) \subseteq \mathcal{A} \end{aligned} \tag{4}$$

As problem (4) is not convex, we employ SCP [24], a heuristic and local optimization method for non-convex problems that leverages convex optimization, where the non-convex portion of the problem is modeled by convex functions that are (at least locally) accurate.

Given the non-convex function $f(\mathbf{S})$, the idea in SCP is to maintain a solution estimate $\mathbf{S}^{[l]}$ and a respective convex *trust region* $\mathcal{T}^{[l]} \subseteq \mathbb{R}^{N \times N}$ over which we "trust" our solution to reside[1]. Then, using a convex approximation $\widehat{f}$ of f, around $\mathbf{S}^{[l]}$, the next solution estimate, $\mathbf{S}^{[l+1]}$, is computed using the optimizer of $\widehat{f}$ in $\mathcal{T}^{[l]}$. Typical trust regions include ℓ_2-norm balls or bounded regions.

For our case, we define as trust region the box:

$$\mathcal{T}^{[l]} = \begin{cases} \mathbf{S} \in \mathcal{S}, \\ \text{supp}\,(\mathbf{S}) \subseteq \mathcal{A} \\ |[\mathbf{S}]_{ij} - [\mathbf{S}^{[l]}]_{ij}| \leq \rho_{ij}(l), \text{ if } (i,j) \in \mathcal{E} \neq 0, \; \forall i,j \in \mathcal{V}, \end{cases} \tag{5}$$

where $\rho_{ij} : \mathbb{Z}_+ \to \mathbb{R}_{++}$ is a mapping from the iteration number to the breadth of the search for the (i,j)th entry.

For the convex approximation of the function, we linearize the function $f(\mathbf{S})$ around the previous estimate $\mathbf{S}^{[l]}$ using its first-order Taylor approximation[2]:

$$\hat{f}^{[l]}(\mathbf{S}) := f(\mathbf{S}^{[l]}) + \text{tr}\left[\nabla_{\mathbf{S}} f(\mathbf{S}^{[l]})^{\top} (\mathbf{S} - \mathbf{S}^{[l]})\right]. \tag{6}$$

We then find a feasible intermediate iterate by solving the problem:

$$\hat{\mathbf{S}} = \underset{\mathbf{S} \in \mathcal{T}^{[l]}}{\arg\min}\; \hat{f}^{[l]}(\mathbf{S}). \tag{7}$$

Due to the non-convexity of the cost function $f(\mathbf{S})$, its value at the (feasible) point $\hat{\mathbf{S}}$ is not guaranteed to be lower than the one at $\mathbf{S}^{[l]}$. Hence, to find the "best" feasible solution $\mathbf{S}$ at the $(l+1)$th iteration, we first resort to a line search to find the optimal scaling step size parameter α_l toward the feasible descent direction $\Delta_l := \hat{\mathbf{S}} - \mathbf{S}^{[l]}$; that is,

$$\alpha_l^* = \arg\min\nolimits_{\alpha_l \in [0,1]} f(\mathbf{S}^{[l]} + \alpha_l \Delta_l). \tag{8}$$

[1]We use the superscript with square brackets to indicate the SCP iterations.

[2]The computation of $\nabla_{\mathbf{S}} f(\mathbf{S}^{[l]})$ is reported in the Appendix.

Then, we compute our next solution estimate $\mathbf{S}^{[l+1]}$ through

$$\mathbf{S}^{[l+1]} = \mathbf{S}^{[l]} + \alpha_l^* \Delta_l, \tag{9}$$

which is feasible for the original problem as long as the set $\mathcal{S}$ is convex, i.e., the update in (9) is a convex combination of feasible points. The specialized SCP procedure for our problem is summarized in Algorithm 2. Note that steps 7-9 guarantee, at each iteration, the feasibility of the iterate and a non-increasing cost function value, leading to the global convergence of Algorithm 1.

Algorithm 2 SCP

Require: $\mathbf{S}^{(n)}$, $\boldsymbol{h}^{(n)}$, $\{\rho_{ij}\}_{(i,j)\in\mathcal{E}}$, $\varepsilon > 0$
1: $l \leftarrow 1$
2: $\mathbf{S}^{[0]} \leftarrow \mathbf{S}^{(n)}$
3: **while** not converged **do**
4: $\quad$ Compute $\{\rho_{ij}(l-1)\}_{(i,j)\in\mathcal{E}}$
5: $\quad$ Construct $\hat{f}^{[l-1]}(\mathbf{S})$ as in (6)
6: $\quad$ Define $\mathcal{T}^{[l-1]}$ as in (5)
7: $\quad \hat{\mathbf{S}} \leftarrow \underset{\mathbf{S}\in\mathcal{T}^{[l-1]}}{\arg\min}\; \hat{f}^{[l-1]}(\mathbf{S})$
8: $\quad \alpha_{l-1}^* \leftarrow \arg\min_{\alpha_{l-1}\in[0,1]} f(\mathbf{S}^{[l-1]} + \alpha_{l-1}(\hat{\mathbf{S}} - \mathbf{S}^{[l-1]}))$
9: $\quad \mathbf{S}^{[l]} \leftarrow \mathbf{S}^{[l-1]} + \alpha_{l-1}^*(\hat{\mathbf{S}} - \mathbf{S}^{[l-1]}))$
10: $\quad$ Check convergence $(\boldsymbol{h}^{(n)}, \mathbf{S}^{[l]}, \varepsilon)$
11: $\quad l \leftarrow l + 1$
12: **end while**
13: **return** $\mathbf{S}^{[l]}$

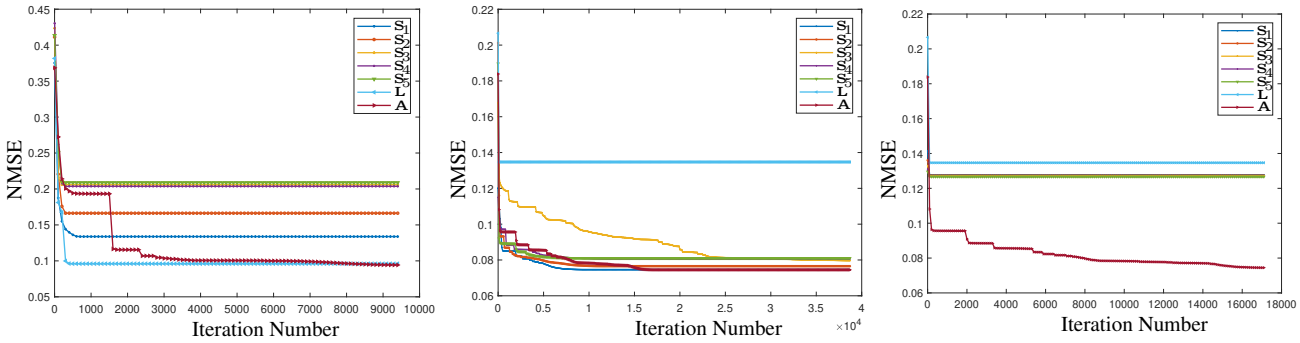
Due to the non-convexity of the cost function, the global optimality of the solution is not guaranteed, thus the results are dependent on the initial starting point(s) as they might lead to different local minima. Despite that in these cases multi-start is recommended, we have found in our numerical experiments that both the unweighted adjacency matrix, $\mathbf{A}$, and the respective combinatorial Laplacian matrix, $\mathbf{L}$, are good initial iterates, i.e., $\mathbf{S}^{(0)}$, for the proposed approach; they are straightforward choices and can be computed using the support of the graph. To validate this claim, in our experiments, we generate initial GSO iterates $\mathbf{S}_i^{(0)}$, through a method reported in the Appendix, and show their performance in the next section, along with those of $\mathbf{A}$ and $\mathbf{L}$.

5. NUMERICAL RESULTS

In this section, we show some numerical results obtained for identifying different graph filters and GSOs $\mathbf{S}$. In these experiments, we consider cases where the GSOs to identify are the weighted adjacency matrix and the Laplacian.

To evaluate the correctness of our method, we first generate a random graph composed of $N = 30$ nodes with the GSP Toolbox [25] and construct from the graph the respective GSO $\mathbf{S}$ involved in the graph filter that generates the output data. We then generate $T = 500$ input graph signals $\{\boldsymbol{x}_t\}_{t=1}^T$ drawn from a standard normal distribution. By fixing the order of the graph filter to $K = 5$, we generate graph filter taps $\boldsymbol{h}$ following a Gaussian distribution with zero mean and $\sigma = 3$. Finally, the output graph signals $\{\boldsymbol{y}_t\}_{t=1}^T$ are generated following (1).

In our experiments, we analyze two main aspects of the proposed method: *i)* the convergence of the algorithm, regardless the initial starting point; and *ii)* the similarity in terms of edge weights between the groundtruth GSO $\mathbf{S}$ and the identified one $\widehat{\mathbf{S}}$. To provide



(a) $S_g = L$, $S_h = L$ (b) $S_g = W$, $S_h = W$ (c) $S_g = W$, $S_h = L$

Fig. 1: NMSE for different settings of the true GSO type S_g and the hypothesis GSO type S_h. The legend in each plot contains the considered GSOs for initializing the algorithm.

a fair comparison, we assume we do not know in advance the type of GSO that generate the network process, i.e., $\mathcal{S}$ is not completely known a priori. For this, we provide a guess of GSO type as input to Algorithm 1, and hope that a proper guess leads to a good fitting. In the sequel, we denote with S_g the type of GSO used to *generate* the data, and with S_h the type of GSO *hypothesized*. Both types of GSOs can assume the values W and L, indicating respectively the (weighted) adjacency matrix and the Laplacian matrix[3].

As performance metric for the error evaluation, we consider the normalized MSE (NMSE), defined as

$$\text{NMSE} = \frac{\sum_{t=1}^{T} \|\widehat{\mathbf{y}}_t - \mathbf{y}_t\|_2^2}{\sum_{t=1}^{T} \|\mathbf{y}_t\|_2^2} \tag{10}$$

where $\widehat{\mathbf{y}}_t$ is the predicted graph signal relative to the input $\mathbf{x}_t$.

Figure 1a shows the NMSE as function of the "cumulative" iteration number[4], for $S_g = S_h = L$. Regardless of the starting point, we observe the non-increasing behavior of the NMSE, corroborating the global convergence of the algorithm. For this particular (S_g, S_h) combination, $\mathbf{L}$ and $\mathbf{A}$ are the best performing starting points in terms of final NMSE, with $\mathbf{L}$ reaching convergence in just a few iterations. The sharp steps downwards, especially noticeable in the case of $\mathbf{A}$ are due to the update of the graph filter coefficients $\boldsymbol{h}$. In this case, the other initial points are not better that the straightforward initial guesses. Similar observations can be made from Fig. 1b. A case of GSO mismatch is shown in Fig. 1c, where the data are generated using the weighted adjacency matrix, but the algorithm is running based on the Laplacian hypothesis. As expected, the $\mathbf{A}$ matrix is the best starting point. Comparing Fig. 1b and Fig. 1c, where the curves starting at $\mathbf{A}$ and $\mathbf{L}$ achieve the same NSMSE, we note how in case of matched hypotheses, the GSOs generated through the generation procedure yield a lower error with respect to the mismatched counterpart.

As a quantitative measure of similarity between the groundtruth and the inferred weights, we report their Spearman correlation coefficient r_s, which is a non parametric measure of rank correlation. In particular, it answers the following question: *do edges with higher weight in the groundtruth GSO tend to have a higher weight in the inferred one?* A perfect Spearman correlation of $+1$ or -1 occurs when each of the variables is a perfect monotone function of the other. In our setting, $r_s = 0.74$ thus confirming a strong positive correlation of the two vectors. Moreover, as depicted in the Q-Q plot of Fig. 2, the quantiles of the two vectors lie almost entirely on the straight line, allowing us to state that the weights of the two GSOs come approximately from the same distribution. For a qualitative and visual assessment of the method, in Fig. 3a and Fig. 3b we show, respectively, the graphs and the weighted sparsity pattern of the groundtruth and the learned Laplacian matrix (for $S_g = S_h = L$). We observe how, up to a scaling factor, the algorithm is able to give a larger weight to those edges that are also "important" in the original graph. All these considerations make us optimistic in the continuation of the development and the study of the proposed approach, driving us toward its application in more complex real-world scenarios.

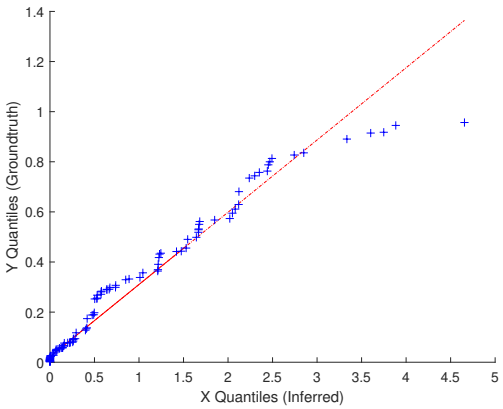


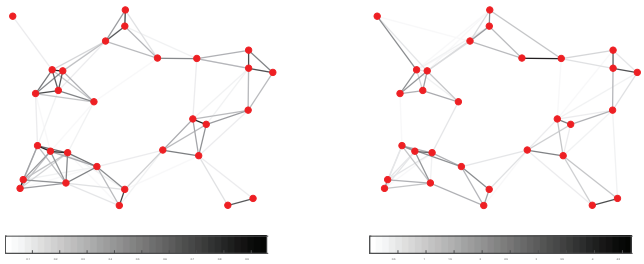
Fig. 2: Q-Q plot of the weights of the groundtruth GSO and the weights of the inferred Laplacian for the case $S_g = L$, $S_h = L$

6. CONCLUSION

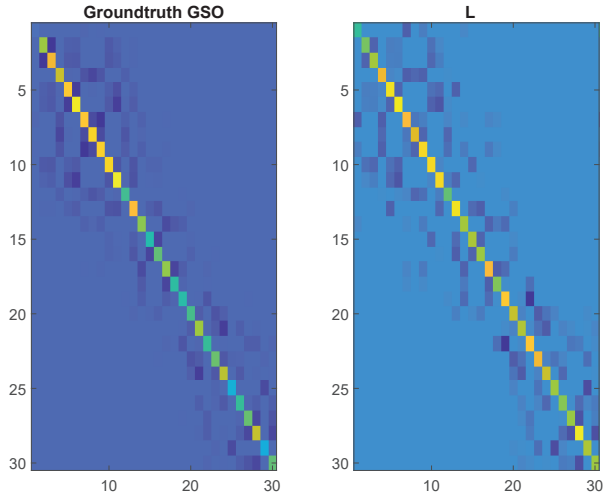
In this work, we formulated and studied the problem of jointly estimating the filter coefficients and the graph shift operator (GSO) defining a graph filter that models the dynamics of signals defined over a network. In particular, motivated by practical scenarios, we exploited the a priori knowledge of the sparsity pattern of the network. We proposed an alternating-minimization approach, whose non-convex subproblem is handled through sequential convex programming methods. As shown in the numerical results, the proposed method is globally convergent and is able to identify the type of GSO used to generate the data. Quantitative statistical measures and qual-

[3]Note how we don't use here the bold notation, because both S_g and S_h are (textual) parameters of the algorithm, in contrast to the considered GSOs starting points that are effectively matrices.

[4]We count all the iterations of the algorithm up to its convergence. We sum in a cumulative manner the outer and the inner iterations of Algorithm 1.



(a) Left: groundtruth graph. Right: inferred graph. The initial condition for the inferred graph was L. The darker the edge in the graph, the higher its value.



(b) Left: groundtruth Laplacian. Right: inferred Laplacian

Fig. 3: (a) Graphs and (b) Laplacian matrix heatmaps for the case $S_g = S_h$ =L

itative graphics demonstrated the efficacy of the algorithm to assign higher values to those weights that are prominent in the real graph.

7. APPENDIX

To compute the derivative $\nabla_{\mathbf{S}} f(\mathbf{S})$, let us first expand the function $f(\mathbf{S})$[5]:

$$\begin{aligned} f(\mathbf{S}) = & \\ & = \text{tr}\left[(\mathbf{Y} - \mathbf{H}(\boldsymbol{h},\mathbf{S})\mathbf{X})(\mathbf{Y} - \mathbf{H}(\boldsymbol{h},\mathbf{S})\mathbf{X})^\top\right] \\ & = \text{tr}\left(\mathbf{Y}\mathbf{Y}^\top\right) - 2\text{tr}\left(\mathbf{H}\mathbf{X}\mathbf{Y}^\top\right) + \text{tr}\left[\mathbf{H}^\top\mathbf{H}\mathbf{X}\mathbf{X}^\top\right] \\ & = \text{tr}\left[\mathbf{Y}\mathbf{Y}^\top\right] - 2\sum_{k=0}^{K} h_k \text{tr}\left[\mathbf{S}^k\mathbf{X}\mathbf{Y}^\top\right] \\ & \quad + \sum_{k_1}^{K}\sum_{k_2}^{K} h_{k_1}h_{k_2}\text{tr}\left(\mathbf{S}^{k_1+k_2}\mathbf{X}\mathbf{X}^\top\right) \end{aligned}$$

[5]We set $\boldsymbol{h}^{(n)}$ to $\boldsymbol{h}$, and $\mathbf{H}(\boldsymbol{h},\mathbf{S})$ to $\mathbf{H}$, for the rest of the proof.

Then

$$\begin{aligned} \nabla_{\mathbf{S}} f(\mathbf{S}) = & \\ & = -2\sum_{k=0}^{K} h_k \nabla_{\mathbf{S}}\text{tr}\left[\mathbf{S}^k\mathbf{X}\mathbf{Y}^\top\right] \\ & \quad + \sum_{k_1}^{K}\sum_{k_2}^{K} h_{k_1}h_{k_2}\nabla_{\mathbf{S}}\text{tr}\left(\mathbf{S}^{k_1+k_2}\mathbf{X}\mathbf{X}^\top\right) \end{aligned}$$

Because $\mathbf{S}$ is symmetric, we have to take into account its structure for the computation of the derivative of $f(\mathbf{S})$. Indeed, due to the matrix symmetry, the overall gradient can be decomposed in:

$$\nabla_{\mathbf{S}} f(\mathbf{S}) = \left[\frac{\partial f(\mathbf{S})}{\partial \mathbf{S}}\right] + \left[\frac{\partial f(\mathbf{S})}{\partial \mathbf{S}}\right]^\top - \text{diag}\left[\frac{\partial f(\mathbf{S})}{\partial \mathbf{S}}\right].$$

Finally, because $\frac{\partial}{\partial \mathbf{S}}\text{Tr}\left(\mathbf{S}^k\right) = k\left(\mathbf{S}^{k-1}\right)^\top$ and $\frac{\partial}{\partial \mathbf{S}}\text{Tr}\left(\mathbf{B}\mathbf{S}^k\right) = \sum_{r=0}^{k-1}\left(\mathbf{S}^r\mathbf{B}\mathbf{S}^{k-r-1}\right)^\top$, we have that the component $[\partial f(\mathbf{S})/\partial \mathbf{S}]$ of the gradient is :

$$\begin{aligned} \frac{\partial f(\mathbf{S})}{\partial \mathbf{S}} = & \\ & = -2\sum_{k=0}^{K} h_k \nabla_{\mathbf{S}}\text{tr}\left[\mathbf{S}^k\mathbf{X}\mathbf{Y}^\top\right] \\ & \quad + \sum_{k_1}^{K}\sum_{k_2}^{K} h_{k_1}h_{k_2}\nabla_{\mathbf{S}}\text{tr}\left(\mathbf{S}^{k_1+k_2}\mathbf{X}\mathbf{X}^\top\right) \\ & = -2\sum_{k=1}^{K} h_k\left[\sum_{r=0}^{k-1}(\mathbf{S}^r\mathbf{X}\mathbf{Y}^\top\mathbf{S}^{k-r-1})^\top\right] \\ & \quad + \sum_{k_1}^{K}\sum_{k_2}^{K} h_{k_1}h_{k_2}\sum_{r=0}^{k_1+k_2-1}(\mathbf{S}^r\mathbf{X}\mathbf{X}^\top\mathbf{S}^{k_1+k_2-r-1})^\top \end{aligned}$$

7.1. GSO Candidate Generation

Let the model be $\boldsymbol{y} = \mathbf{H}(\boldsymbol{h},\mathbf{S})\boldsymbol{x}$ for some order K of the filter. Then, a $K = 1$ approximation for the overall problem (2) is given by

$$\boldsymbol{y} \approx (\hat{h}_0^{(1)}\mathbf{I} + \hat{\mathbf{S}}_1)\boldsymbol{x}, \tag{11}$$

where $\hat{h}_0^{(1)}$ is the constant filter tap estimate related to the first order approximation, and $\hat{\mathbf{S}}_1 \in \mathcal{S}$ is the respective estimate for the GSO. We assume $\hat{h}_1^{(1)} = 1$ to avoid the scalar ambiguity that would otherwise arise in the term $\hat{h}_1\hat{\mathbf{S}}_1$. This also decouples the filter parameters from the GSO, both of which can be estimated by LLS. This way, a first GSO candidate $\mathbf{S}_1^{(0)}$ for the algorithm is found. Next, we consider a second order approximation of the model

$$\boldsymbol{y} \approx (\hat{h}_0^{(2)}\mathbf{I} + \hat{\mathbf{S}}_2 + \hat{h}_2^{(2)}\mathbf{S}_1^{(0)2})\boldsymbol{x}, \tag{12}$$

where the variables are now $\hat{h}_0^{(2)}, \hat{h}_2^{(2)}$ and $\hat{\mathbf{S}}_2$, and we still assume the first filter tap $\hat{h}_1^{(2)}$ is equal to one. This again leads to a LLS problem which generates a second GSO candidate $\mathbf{S}_2^{(0)}$. We iterate this procedure by increasing the order of the filter at each step, and maintaining the term that is linear in the GSO variable $\mathbf{S}$. At the end, we have K initial GSO candidates $\mathbf{S}_1^{(0)}, \mathbf{S}_2^{(0)}, \ldots, \mathbf{S}_K^{(0)}$, which can be given as input to Algorithm 1.

8. REFERENCES

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

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

[3] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[4] David I Shuman, Pierre Vandergheynst, Daniel Kressner, and Pascal Frossard, "Distributed signal processing via chebyshev polynomial approximation," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 4, pp. 736–751, 2018.

[5] Santiago Segarra, Antonio G Marques, and Alejandro Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 4117–4131, 2017.

[6] Sunil K Narang, Akshay Gadde, and Antonio Ortega, "Signal processing techniques for interpolation in graph structured data," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 5445–5449.

[7] Benjamin Girault, Paulo Gonçalves, Eric Fleury, and Arashpreet Singh Mor, "Semi-supervised learning for graph to signal mapping: A graph signal wiener filter interpretation," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 1115–1119.

[8] Elvin Isufi, Paolo Di Lorenzo, Paolo Banelli, and Geert Leus, "Distributed wiener-based reconstruction of graph signals," *2018 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 21–25, 2018.

[9] Masaki Onuki, Shunsuke Ono, Masao Yamagishi, and Yuichi Tanaka, "Graph signal denoising via trilateral filter on graph spectral domain," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 137–148, 2016.

[10] J. Ma, W. Huang, S. Segarra, and A. Ribeiro, "Diffusion filtering of graph signals and its use in recommendation systems," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 4563–4567.

[11] Elvin Isufi, Andreas Loukas, Nathanael Perraudin, and Geert Leus, "Forecasting time series with varma recursions on graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 18, pp. 4870–4885, 2019.

[12] A. Natali, E. Isufi, and G. Leus, "Forecasting multi-dimensional processes over graphs," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 5575–5579.

[13] Fernando Gama, Antonio G Marques, Geert Leus, and Alejandro Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 1034–1049, 2018.

[14] Mario Coutino, Elvin Isufi, and Geert Leus, "Advances in distributed graph filtering," *IEEE Transactions on Signal Processing*, vol. 67, no. 9, pp. 2320–2333, 2019.

[15] Santiago Segarra, Gonzalo Mateos, Antonio G Marques, and Alejandro Ribeiro, "Blind identification of graph filters," *IEEE Transactions on Signal Processing*, vol. 65, no. 5, pp. 1146–1159, 2016.

[16] Jiani Liu, Elvin Isufi, and Geert Leus, "Filter design for autoregressive moving average graph filters," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 1, pp. 47–60, 2018.

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

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

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

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

[21] Rasoul Shafipour, Santiago Segarra, Antonio G Marques, and Gonzalo Mateos, "Identifying the topology of undirected networks from diffused non-stationary graph signals," *arXiv preprint arXiv:1801.03862*, 2018.

[22] R. Shafipour, S. Segarra, A. G. Marques, and G. Mateos, "Directed network topology inference via graph filter identification," in *2018 IEEE Data Science Workshop (DSW)*, June 2018, pp. 210–214.

[23] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077–2092, April 2017.

[24] Stephen Boyd, "Sequential convex programming," *Lecture Notes, Stanford University*, 2008.

[25] Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst, and David K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014.