LEARNING CAUSAL NETWORKS TOPOLOGY FROM STREAMING GRAPH SIGNALS

Mircea Moscu [1], *Roula Nassif* [2], *Fei Hua* [3], *Cédric Richard* [1]

[1] Laboratoire Lagrange, Université Côte d'Azur, OCA, CNRS, Nice, France
[2] Ecole Polytechnique Fédérale de Lausanne, Switzerland
[3] School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, China
Email: mircea.moscu@oca.eu; roula.nassif@epfl.ch; fei.hua@oca.eu; cedric.richard@unice.fr

ABSTRACT

Modern data analysis and processing tasks usually involve large sets of data structured by a graph. Typical examples include brain activity supported by neurons, data shared by users of social media, and traffic on transportation or energy networks. There are often settings where the graph is not readily available, and has to be estimated from data. This paper focuses on estimating a network structure capturing the dependencies among streaming graph signals in the form of a possibly directed, weighted adjacency matrix. Several works proposed centralized offline solutions to address this problem, without paying much attention to the distributed nature of networks. We start from a centralized setting and show how, by introducing a simple yet powerful data model, we can infer a graph structure from streaming data with a distributed online learning algorithm. Our algorithm is tested experimentally to illustrate its usefulness, and successfully compared to a centralized offline solution of the literature.

Index Terms— Network topology, graph signal processing, distributed learning, online learning

1. INTRODUCTION

In the last past years, data has become a raw resource that needs to be collected and refined before becoming useful information. Data are abundant and diverse, taking different forms, stemming from different sources: e-commerce, sporting events, entertainment media, and social interactions, to name a few. Structured data, where each component is linked in some way to others, are ubiquitous and generally evolve over time, making it difficult to process and analyze. Since seminal works such as [1], graph signal processing (GSP) has attracted great attention due to the potential applications it offers. Typical examples include brain activity imaging, social media analysis, and transportation or energy networks monitoring.

Most graph signal processing algorithms introduced in the last five years assume prior knowledge of the graph structure. However, there are often settings where the graph is not readily available, and has to be inferred from data by capturing the underlying relationship between the characteristics of the observations at each node. This paper focuses on estimating a network structure capturing the dependencies among streaming graph signals in the form of a possibly directed, weighted adjacency matrix.

Definitions: A graph $\mathcal{G}$ consists of a set $\mathcal{N}$ of N nodes, and a set $\mathcal{E}$ of edges such that if nodes m and n are linked, then $(m,n) \in \mathcal{E}$. For undirected graphs, these node pairs are unordered. Notation $\mathcal{N}_n$ stands for the set of indices of nodes in the neighbourhood of node n, i.e., $\mathcal{N}_n = \{m\colon (m,n) \in \mathcal{E}\}$.

At the node level, we collect a signal $\boldsymbol{x} \triangleq [x_1, \ldots, x_N]^\top$, assumed to be real-valued, where x_n is the sample of the signal $\boldsymbol{x}$ at node n. We endow the graph $\mathcal{G}$ with a shift operator [1], defined as an $N \times N$ matrix $\boldsymbol{S}$. Entries s_{nm} are non-zero if $(m,n) \in \mathcal{E}$. This matrix encodes the underlying graph connectivity. Valid choices for this operator are the adjacency matrix (weighted or not) or Laplacian matrix (and its variations) [2]. Operation $\boldsymbol{S}\boldsymbol{x}$ is called a graph shift and can be performed locally at each node n by aggregating samples in its neighborhood, i.e., $\sum_{m\in\mathcal{N}_n} s_{nm}x_m$. Also, $\boldsymbol{S}^k\boldsymbol{x}$ represents a shift of order k that aggregates samples from k-hop neighbors.

Prior works: For topology identification, several works have been put forward. A very early proposition is in [3], where a covariance estimation based method of inferring links is introduced. On the same line, in [4] the graphical Lasso is employed in order to estimate the inverse covariance matrix from data. In [5], the authors advocate that connectivity can be recovered from spectral templates, under the assumptions that the graph signal $\boldsymbol{x}$ is stationary and generated through a diffusion process. They estimate the shift operator $\boldsymbol{S}$ under a set of constraints that yields a matrix with desirable properties, such as zeros on the diagonal, sparsity and symmetry. The authors in [6] propose an adaptive algorithm for learning the topology from streaming graph signals driven by a diffusion process.

Under a graph signal smoothness assumption, the so-called pairwise distances matrix $\mathbf{Z}$ with entries defined as $z_{mn} \triangleq \|x_m - x_n\|^2$, is introduced in [7] to estimate a weighted adjacency matrix $\mathbf{W}$. The problem is then solved by assigning smaller weights to far away nodes, while reducing the number of less connected nodes and exceedingly large weights. Other solutions include the use of dictionaries. The authors in [8] devise a method for learning a dictionary that is able to efficiently represent the signals as linear combinations of atoms. Structural equation models are used in [9] to track slowly time-varying networks, with application to contagion propagation. Kernels have seen widespread use in topology inference problems. One of these works is [10] where kernels, chosen to best fit the data, model nonlinear relationships between nodes based on measurements at successive time instants. The authors present an auto-regressive framework that allows to track graph connectivity over time, proving useful in providing insights on brain connectivity. The multi-kernel approach in [11] uses partial correlations to encode graph topology and ℓ_p-norm regression to enhance performance. A review of the state of the art methods for graph topology inference is given in [12].

Encoding the graph topology with the shift matrix $\boldsymbol{S}$ is ubiquitous in graph signal models. This operator describes the interactions between entities and, by extension, it can be considered as a tool for representing relationships between data. Unlike existing meth-

ods, this paper focuses on identifying the topology of a graph from streaming graph signals in a distributed and online manner.

Notations: Normal font letters denote scalars, and boldface lowercase and uppercase letters stand for column vectors and matrices, respectively. Uppercase calligraphic letters denote sets. We denote by $\text{supp}\{\boldsymbol{A}\}$ the support of $\boldsymbol{A}$, and by $|\mathcal{N}|$ the cardinality of $\mathcal{N}$. $\lambda_{\max}(\cdot)$ stands for the largest eigenvalue of its matrix argument.

2. CENTRALIZED PROBLEM FORMULATION

2.1. Shift-invariant graph filtering

In this work, we focus on graph-based filtering framework. A graph filter takes a signal on graph $\boldsymbol{x}(i)$ as input, and outputs a signal $\boldsymbol{y}(i)$ given by $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x}$ indexed by the same graph [13]. Different forms have been considered for $\boldsymbol{H}$ in the literature. For example, the K-th order linear shift invariant graph filter is defined as [14, 15]:

$$\boldsymbol{y}(i) = \sum_{k=0}^{K-1} h_k \boldsymbol{S}^k \boldsymbol{x}(i), \quad i \geq 0, \tag{1}$$

with $\boldsymbol{S}$ a shift matrix and $\{h_k\}_{k=0}^{K-1}$ the filter coefficients. Observe that the previous model assumes the instantaneous diffusion of information, which may appear as a limitation of this model. A dynamical model was introduced to overcome this restriction [16, 17]:

$$\boldsymbol{y}(i) = \sum_{k=0}^{K-1} h_k \boldsymbol{S}^k \boldsymbol{x}(i-k), \qquad i \geq K-1. \tag{2}$$

Assuming that the shift matrix $\boldsymbol{S}$ is known, the authors in [16, 18] show how diffusion adaptation strategies can be applied to estimate the filter coefficients $\{h_k\}_{k=0}^{K-1}$ from streaming data $\{\boldsymbol{x}(i), \boldsymbol{y}(i)\}$.

2.2. Network topology inference

One possible tool for causal network topology inference is the multivariate autoregressive model defined as:

$$\boldsymbol{y}(i) = \sum_{k=0}^{K-1} \boldsymbol{S}^k \boldsymbol{x}(i-k) + \boldsymbol{v}(i), \quad i \geq K-1 \tag{3}$$

where $\boldsymbol{S}^k \triangleq \{s_{nm,k}\}$ in the above power series contains autoregressive coefficients that describe the influence of node m on node n at a distance of k hops, and $\boldsymbol{v}(i)$ is innovation noise. This model is helpful to asses Granger causality, where x_m is said to Granger-cause x_ℓ if knowledge of the former improves the prediction of the latter [19].

Consider a connected network with N nodes. We assume that each node ℓ knows the set of its neighbors $\mathcal{N}_\ell$ with which it communicates. We however assume that the support of $\boldsymbol{S}$ is unknown. The problem is to estimate $\boldsymbol{S}$ from streaming data $\{\boldsymbol{x}(i), \boldsymbol{y}(i)\}$. We assume that signal $\boldsymbol{x}(i)$ is zero-mean wide-sense stationary, i.e., correlation sequence $\boldsymbol{R}_x(k) = \mathbb{E}\{\boldsymbol{x}(i)\boldsymbol{x}^\top(i-k)\}$ is a function of the time lag k only. The noise $\boldsymbol{v}(i) = [v_1(i), \ldots, v_N(i)]^\top$ is assumed zero-mean, i.i.d., with covariance $\boldsymbol{R}_v = \text{diag}\{\sigma_{v,n}\}_{n=1}^N$. Under these assumptions, estimating matrix $\boldsymbol{S}$ in (3) can be performed by solving the following mean-square-error problem:

$$\begin{aligned} \boldsymbol{S}^* = \underset{\boldsymbol{S}}{\text{argmin}}\ \mathbb{E}\Big\|\boldsymbol{y}(i) - \sum_{k=0}^{K-1} \boldsymbol{S}^k \boldsymbol{x}(i-k)\Big\|^2 + \eta\, \Phi(\boldsymbol{S}) \\ \text{subject to } s_{nm} = 0 \ \text{ if } \ m \notin \mathcal{N}_n, \quad n = 1, \ldots, N \end{aligned} \tag{4}$$

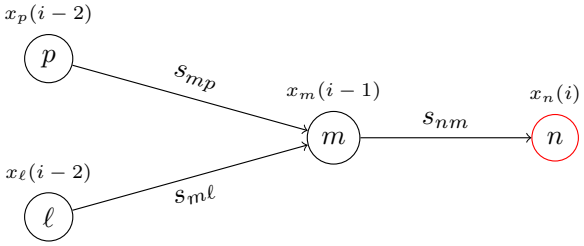


Fig. 1. Data paths toward node n. Links are depicted as directed edges in order to illustrate the flow of weighted data. In order to estimate its own s_{nm}, node n receives from its neighbour m the $(K-1)$-element vector $[x_m(i-1), s_{mp}x_p(i-2)+s_{m\ell}x_\ell(i-2)]^\top$.

with $\eta > 0$. The objective function in (4) includes a regularization term $\Phi(\boldsymbol{S})$ to account for some prior knowledge of $\boldsymbol{S}$ such as symmetry or sparsity. The constraints aim at forcing to zero the entries s_{nm} of $\boldsymbol{S}$ corresponding to node pairs $(n, m) \notin \mathcal{E}$.

Formulation (4) is non-convex, due to the matrix polynomial. This leads any resolution algorithm to possibly converge toward a local minimum rather than a global one. Reference [17] considers a similar problem in a centralized setting where the data across the network are collected and processed by a fusion center. In the next section, we shall show how the entries of $\boldsymbol{S}$ can be estimated in a distributed manner where nodes perform local computations and exchange information only with their neighbors.

3. DISTRIBUTED SOLUTION

3.1. Problem reformulation

The following strategy allows each node to locally estimate its own non-zero entries in $\boldsymbol{S}$. According to (3), the output $y_n(i)$ at each node n is given by:

$$y_n(i) = \sum_{k=0}^{K-1} \Big[\boldsymbol{S}^k \boldsymbol{x}(i-k)\Big]_n + v_n(i) \tag{5}$$

This can be rewritten as:

$$\begin{aligned} y_n(i) = x_n(i) + \boldsymbol{s}_n^\top [\boldsymbol{x}(i-1)]_{m\in\mathcal{N}_n} + \ldots + \\ \boldsymbol{s}_n^\top [\boldsymbol{S}^{K-2}\boldsymbol{x}(i-K+1)]_{m\in\mathcal{N}_n} + v_n(i), \end{aligned} \tag{6}$$

with

$$\boldsymbol{s}_n = \text{col}\{s_{nm} : m \in \mathcal{N}_n\} \tag{7}$$

the $|\mathcal{N}_n| \times 1$ vector aggregating all non-zero entries of the n-th row of $\boldsymbol{S}$. By subtracting $x_n(i)$ from $y_n(i)$, (6) can be expressed as:

$$\bar{y}_n(i) \triangleq y_n(i) - x_n(i) = \boldsymbol{z}_n^\top(i)\boldsymbol{s}_n + v_n(i), \tag{8}$$

where $\boldsymbol{z}_n(i)$ is a $|\mathcal{N}_n| \times 1$ column vector defined as:

$$\boldsymbol{z}_n(i) = \text{col}\left\{\sum_{k=0}^{K-2} \Big[\boldsymbol{S}^k \boldsymbol{x}(i-k-1)\Big]_m : m \in \mathcal{N}_n\right\}. \tag{9}$$

Reformulating (6) in the form (8) has the following rationale. Consider node n. At time instant i, this node weights the incoming data from its neighbors with the corresponding entries of the n-th

row of $\boldsymbol{S}$. The same reasoning holds for any neighboring node m of n, as illustrated in Fig. 1, which weights its own incoming data with entries of the m-th row of $\boldsymbol{S}$. This means that two-hop data sent by node ℓ at time instant $i-2$, passing through node m at time instant $i-1$, and received by node n at time instant i, are successively weighted by $s_{m\ell}$ and s_{nm}. Therefore, when estimating $\boldsymbol{S}$, node n can simply focus on its own weights stored in the n-th row of $\boldsymbol{S}$ provided that every other node in the network does the same with its own weights.

Reformulation (7)–(9) comes along with several benefits compared to the centralized solution in [17]. The main one concerns computational efficiency since only one-hop regressors $\boldsymbol{z}_n(i)$ are considered at each node n. These one-hop transfers also translate into lower communication costs.

3.2. Algorithm

We reformulate problem (4) by introducing the following aggregate cost function:

$$J(\boldsymbol{S}) = \sum_{n=1}^{N} J_n(\boldsymbol{s}_n) \tag{10}$$

where $J_n(\boldsymbol{s}_n)$ denotes the mean-square-error cost at node n, namely,

$$J_n(\boldsymbol{s}_n) \triangleq \mathbb{E}\|\bar{y}_n(i) - \boldsymbol{z}_n^{\top}(i)\boldsymbol{s}_n\|^2 + \eta_n \Phi(\boldsymbol{s}_n). \tag{11}$$

This form allows each node n to estimate its known entries $\boldsymbol{s}_n$ of $\boldsymbol{S}$, and to possibly account for some prior knowledge of $\boldsymbol{S}$ via $\Phi(\boldsymbol{s}_n)$.

For illustration purpose, we shall impose a symmetry constraint on $\boldsymbol{S}$. Following the strategy in [20], we address this problem by considering the following local cost function:

$$J_n(\boldsymbol{s}_n) \triangleq \mathbb{E}\|\bar{y}_n(i) - \boldsymbol{z}_n^{\top}(i)\boldsymbol{s}_n\|^2 + \eta_n \sum_{m\in\mathcal{N}_n} (s_{nm} - s_{mn})^2 \tag{12}$$

where parameter $\eta_n > 0$ controls the relative importance of respecting the symmetry constraint on $\boldsymbol{S}$ [21]. To minimize (12), we propose an incremental solution based on gradient descent, namely,

$$\boldsymbol{s}_n(i+1) = \boldsymbol{s}_n(i) + \mu_n[\boldsymbol{r}_{z_n y} - \boldsymbol{R}_{z_n}\boldsymbol{s}_n(i) - \eta_n \boldsymbol{\delta}(i)] \tag{13}$$

where $\boldsymbol{\delta}(i) = \boldsymbol{s}_n(i) - \tilde{\boldsymbol{s}}_n(i)$ with $\tilde{\boldsymbol{s}}_n = \text{col}\{s_{mn} : m \in \mathcal{N}_n\}$, μ_n a sufficiently small positive step-size, and:

$$\boldsymbol{R}_{z_n} \triangleq \mathbb{E}\{\boldsymbol{z}_n(i)\boldsymbol{z}_n^{\top}(i)\} \text{ and } \boldsymbol{r}_{z_n y} \triangleq \mathbb{E}\{\boldsymbol{z}_n(i)\bar{y}_n(i)\}. \tag{14}$$

Step-sizes μ_n in (13) must satisfy $0 < \mu_n < 2/\lambda_{\max}(\boldsymbol{R}_{z_n} + \eta_n \boldsymbol{I})$ in order to guarantee stability in the mean under certain independence conditions on the data [22]. However, since second-order moments are rarely available beforehand, a stochastic gradient descent strategy has to be devised. It consists of choosing instantaneous approximations such as:

$$\boldsymbol{R}_{z_n} \approx \boldsymbol{z}_n(i)\boldsymbol{z}_n^{\top}(i) \text{ and } \boldsymbol{r}_{z_n y} \approx \boldsymbol{z}_n(i)\bar{y}_n(i). \tag{15}$$

We used the adapt-then-penalize approach introduced in [23] to implement the algorithm (13) with (15). Setting $\epsilon_n(i) \triangleq \bar{y}_n(i) - \boldsymbol{z}_n^{\top}(i)\boldsymbol{s}_n(i)$, and introducing the intermediate estimate $\boldsymbol{\phi}_n(i)$, this strategy translates (13) into:

$$\boldsymbol{\phi}_n(i+1) = \boldsymbol{s}_n(i) + \mu_n \epsilon_n(i)\boldsymbol{z}_n(i), \tag{16a}$$

$$\boldsymbol{s}_n(i+1) = \boldsymbol{\phi}_n(i+1) - \mu_n\eta_n[\boldsymbol{\phi}_n(i+1) - \widetilde{\boldsymbol{\phi}}_n(i+1)] \tag{16b}$$

with $\widetilde{\boldsymbol{\phi}}_n(i+1) \triangleq \text{col}\{[\boldsymbol{\phi}_m(i+1)]_n : m \in \mathcal{N}_n\}$ the column vector that aggregates all partial estimates related to node n in partial estimates of its neighboring nodes. The algorithm is synthesized hereafter.

Algorithm 1: Local estimation of graph topology

For every node n:
Inputs: Parameters μ_n and η_n
Initialization: Randomly set all entries of $\boldsymbol{s}_n(0)$
Algorithm: At each time instant $i \geq 1$
Get weighted data $\left[\boldsymbol{S}^{k-1}\boldsymbol{x}(i-k)\right]_{m\in\mathcal{N}_n}$ from neighbors
Compute the regressor $\boldsymbol{z}_n(i)$ with (9)
Update the local estimate $\boldsymbol{s}_n$ with (16)

4. EXPERIMENTS

Multiple synthetic experiments were conducted with the goal of estimating $\boldsymbol{s}_n$ at each node n using the proposed algorithm. The resulting estimates were then aggregated into an estimate of $\boldsymbol{S}$. Next, this estimated matrix $\boldsymbol{S}$ was used in the context of spectral clustering for illustration purpose only. This particular application was chosen for its simplicity to illustrate how estimates obtained with our algorithm can provide useful insight into the topology of the graph.

Setting: An undirected community graph was generated using GSPBOX [24], with $N = 32$ nodes forming two communities. The adjacency matrix is shown in Fig. 2(a). The graph shift operator $\boldsymbol{S}$ was chosen to be $\boldsymbol{W}/[1.1 \cdot \lambda_{\max}(\boldsymbol{W})]$, the normalized weighted adjacency matrix. Weights w_{mn} were set to $\exp(-\gamma\,\|\boldsymbol{c}_m - \boldsymbol{c}_n\|^2)$, where $\boldsymbol{c}_n$ are the coordinates of the 2D embedding for node n. In order to illustrate the adaptation abilities of the method, we changed the shift operator during the experiment. Parameter γ was set to 0.1 during the first part of the experiment, and then changed to 0.6 for the second part. We considered an i.i.d. zero-mean Gaussian signal $\boldsymbol{x}(i)$ with covariance matrix $\boldsymbol{R}_x$ chosen to be the solution of the Lyapunov equation $\boldsymbol{S}\boldsymbol{R}_x\boldsymbol{S}^{\top} - \boldsymbol{R}_x + \boldsymbol{I} = \boldsymbol{0}$. Noise $\boldsymbol{v}(i)$ was also zero-mean Gaussian with covariance matrix $\boldsymbol{R}_v = \text{diag}\{\sigma_{v,n}^2\}_{n=1}^N$. Variances $\sigma_{v,n}^2$ were generated from the uniform distribution $\mathcal{U}(0.1, 0.15)$. We set the filter order to $K = 3$. Output data $\boldsymbol{y}(i)$ were generated with model (3). We used a constant step-size μ for all nodes and parameters η_n were set to 300. For each experiment, estimates were averaged over 50 Monte-Carlo runs.

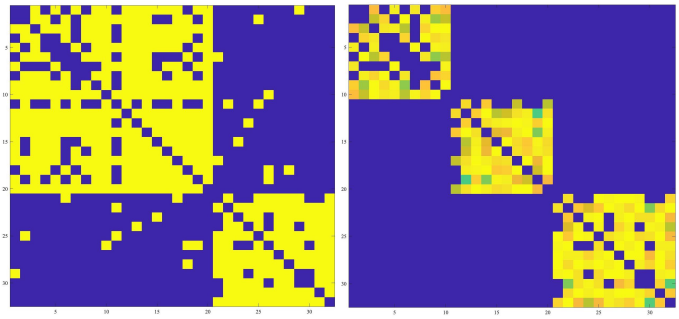
Experiment 1: Learning algorithm (16) was run to estimate $\boldsymbol{S}$. The estimated mean squared deviation (MSD) learning curve, defined as:

$$\text{MSD}(i) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}\|\boldsymbol{s}_n^* - \boldsymbol{s}_n(i)\|^2 \tag{17}$$

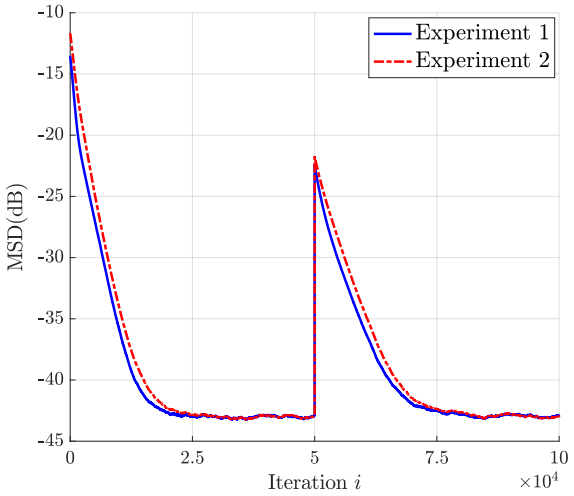
is depicted in Fig. 2(c). It shows that the algorithm converged monotonically to a reasonably low MSD, and succeeded in adapting to the change in $\boldsymbol{S}$ at time $i = 50000$.

Experiment 2: Data were generated as in Experiment 1, with a new shift matrix $\boldsymbol{S}'$ such that $\text{supp}\{\boldsymbol{S}'\} \subseteq \text{supp}\{\boldsymbol{S}\}$. This allowed us to consider a new setting where, even if a node m is linked to a node n, i.e., $(m, n) \in \mathcal{E}$, the output signal $y_n(i)$ at node n does not necessarily depend on the input signal $x_m(i-1)$ at node m via model (3). To design $\boldsymbol{S}'$, we selected a subset of nodes in one of the two communities of the initial community graph, and we divided their connection weights in $\boldsymbol{S}$ with all other nodes by 100. The resulting shift matrix $\boldsymbol{S}'$ is depicted in Fig. 2(b). In this way, we obtained two clusters according to the adjacency matrix $\boldsymbol{A}$, and three clusters according to the shift matrix $\boldsymbol{S}'$.

Learning algorithm (16) was run to estimate $\boldsymbol{S}'$. The learning curve represented in Fig. 2(c) shows that the algorithm converged to a reasonably low MSD, at a slower rate than in Experiment 1 possibly because of the larger number of clusters. To check this



(a) Adjacency matrix (b) Shift matrix $\boldsymbol{S}'$



(c) MSD

Fig. 2. (a) Adjacency matrix considered for the two experiments. (b) Shift matrix $\boldsymbol{S}'$ used in Experiment 2. (c) MSD learning curves.

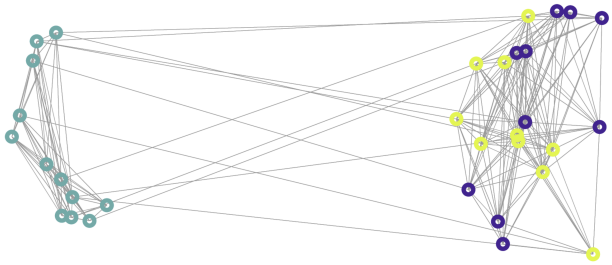


Fig. 3. Spectral clustering performed during Experiment 2. Two communities can be observed in the graph topology. At the graph signal level, three clusters are identified.

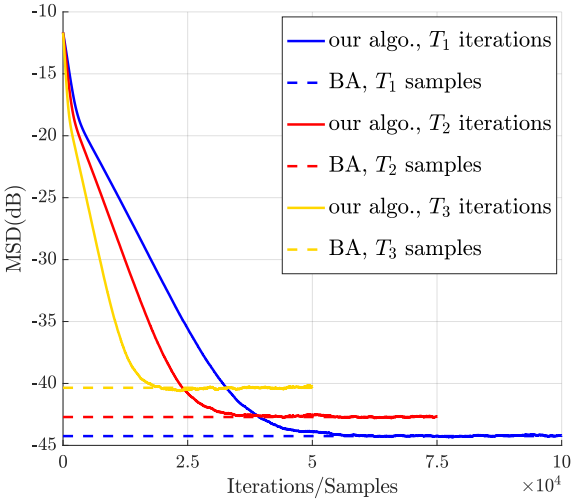


Fig. 4. Comparison of our algorithm with BA for 3 training set sizes: $T_1 = 10^5, T_2 = 7.5 \cdot 10^4$ and $T_3 = 5 \cdot 10^4$ samples.

assumption, we computed the eigen-decomposition of the estimated shift matrix to infer the number of clusters [25]. It was numerically found to be equal to 3. Finally, we performed a spectral clustering of the nodes with a k-means algorithm based on the first $k = 3$ eigenvectors. The result depicted in Fig. 3 is in accordance with the experimental setup.

Experiment 3: Comparisons were conducted with the centralized batch algorithm derived in [17], called benchmark algorithm (BA). We considered the same experimental setup as in Experiment 1, except that the number of nodes was set to $N = 20$. No regularization term $\Phi(\cdot)$ was used. Since it deals with a more complex polynomial model than our algorithm, we simplified the BA model by setting its extra coefficients to 0. As BA is a batch-mode algorithm which estimates model parameters from training data, we successively set the size of the training set to $T_1 = 10^5, T_2 = 7.5 \cdot 10^4$, and $T_3 = 5 \cdot 10^4$ samples. In each case, parameters of BA were set to achieve the best possible MSD. Next we set the step-sizes μ_n of our algorithm to achieve the same MSD at steady-state as BA. The results are presented in Fig. 4. We observe that our algorithm was able to achieve the same MSD with half of the training samples. From a computational point of view, note that our method needs to process every sample only once, whereas the BA processes the whole training set many more times, depending on the chosen solver.

5. CONCLUSION

In this paper, we proposed a distributed online strategy for topology identification based on graph signals. This framework allows to estimate a weighted adjacency matrix based on local one-hop computations. Since most of state-of-the-art topology inference algorithms work in a batch mode, this online approach represents a step forward. A second step forward is that our algorithm can adapt in an online way to changes in the graph shift operator.

For future work, multiple directions of research can be followed. A first one would be using other regularizers in the optimization problem, such as Lasso or group-Lasso in order to control the number of estimated connections. Another option would be to extend our method by considering a kernel-based framework in order to cope with nonlinear relationships between agents.

6. REFERENCES

[1] 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 Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[2] N. Biggs, *Algebraic Graph Theory*, Cambridge University Press, 1993.

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

[4] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical Lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–41, 2008.

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

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

[7] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. International Conference on Artificial Intelligence and Statistics*, 2016, pp. 920–929.

[8] D. Thanou, D. I. Shuman, and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Transactions on Signal Processing*, vol. 62, no. 15, 2014.

[9] B. Baingana, G. Mateos, and G. B. Giannakis, "Dynamic structural equation models for tracking topologies of social networks," in *Proc. IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, St. Martin, France, 2013, pp. 292–295.

[10] Y. Shen, B. Baingana, and G. B. Giannakis, "Topology inference of directed graphs using nonlinear structural vector autoregressive models," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 6513–6517.

[11] L. Zhang, G. Wang, and G. B. Giannakis, "Going beyond linear dependencies to unveil connectivity of meshed grids," in *Proc. IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Curaçao, Dutch Antilles, 2017, pp. 1–5.

[12] Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard, "Learning graphs from data: A signal representation perspective," *arXiv preprint arXiv:1806.00848*, 2018.

[13] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Graph filters," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, Canada, 2013, pp. 6163–6166.

[14] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. 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.

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

[16] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "Distributed diffusion adaptation over graph signals," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018, pp. 4129–4133.

[17] 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, 2017.

[18] F. Hua, R. Nassif, C. Richard, H. Wang, and A. H. Sayed, "A preconditioned graph diffusion LMS for adaptive graph signal processing," in *Proc. European Conference on Signal Processing (EUSIPCO)*, Rome, Italy, 2018, pp. 1–5.

[19] A. Bolstad, B. D. Van Veen, and R. Nowak, "Causal network inference via group sparse regularization," *IEEE Transactions on Signal Processing*, vol. 59, no. 6, pp. 2628–2641, 2011.

[20] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, "Diffusion LMS for multitask problems with local linear equality constraints," *IEEE Transactions on Signal Processing*, vol. 65, no. 19, pp. 4979 – 4993, 2017.

[21] Z. J. Towfic and A. H. Sayed, "Adaptive penalty-based distributed stochastic convex optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 15, pp. 3924–3938, 2014.

[22] A. H. Sayed, *Adaptive Filters*, John Wiley & Sons, 2008.

[23] C. Yu and A. H. Sayed, "Learning by networked agents under partial information," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, 2017, pp. 3874–3878.

[24] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014.

[25] N. Tremblay, G. Puy, P. Borgnat, R. Gribonval, and P. Vandergheynst, "Accelerated spectral clustering using graph filtering of random signals," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4094–4098.