Learning Heat Diffusion Graphs

Dorina Thanou, Xiaowen Dong, Daniel Kressner, and Pascal Frossard

***Abstract*—Information analysis of data often boils down to properly identifying their hidden structure. In many cases, the data structure can be described by a graph representation that supports signals in the dataset. In some applications, this graph may be partly determined by design constraints or predetermined sensing arrangements. In general though, the data structure is not readily available nor easily defined. In this paper, we propose to represent structured data as a sparse combination of localized functions that live on a graph. This model is more appropriate to represent local data arrangements than the classical global smoothness prior. We focus on the problem of inferring the connectivity that best explains the data samples at different vertices of a graph that is *a priori* unknown. We concentrate on the case where the observed data are actually the sum of heat diffusion processes, which is a widely used model for data on networks or other irregular structures. We cast a new graph learning problem and solve it with an efficient nonconvex optimization algorithm. Experiments on both synthetic and real world data finally illustrate the benefits of the proposed graph learning framework and confirm that the data structure can be efficiently learned from data observations only. We believe that our algorithm will help solving key questions in diverse application domains such as social and biological network analysis where it is crucial to unveil proper data structure for understanding and inference.**

***Index Terms*—Graph signal processing, heat diffusion, Laplacian matrix learning, representation theory, sparse prior.**

I. Introduction

DATA analysis and processing tasks typically involve large sets of structured data, where the structure carries critical information about the nature of these data. One can find numerous examples of such datasets in a wide diversity of application domains, such as transportation networks, social or computer networks, brain analysis or even digital imaging and vision. In such datasets, there is a clear interplay between the structure (e.g., the transportation network) and the data (e.g., traffic measurements, or signals, captured by sensors in different parts of the network) that should be exploited for data analysis. Graphs are commonly used to describe the structure of such data as they provide a flexible tool for representing and eventually manipulating information that resides on topologically complicated domains. In particular, the structure of the data is captured by the topology of the graph and the data or signals are modeled as the result of graph processes evolving on that topology. Once an appropriate graph is constructed, inference and analysis tasks can be carried out with a careful consideration of the data structure using, for example, spectral theory [1] or graph signal processing [2] concepts. While recent research has put a focus on the development of effective methods for processing data on graphs and networks, relatively little attention has been given to the definition of graphs when they are not readily available in the datasets. This problem remains critical and may actually represent the major obstacle towards effective processing of structured data.

In this work, we first propose a generic graph signal model where the data consists of (sparse) combinations of overlapping local patterns that reside on the graph. These patterns may describe localized events or specific processes appearing at different vertices of the graph, such as traffic bottlenecks in transportation networks or rumor sources in social networks. This model is more appropriate for capturing specific or local properties of structured data than the traditional global smoothness model [3], [4][1]. More specifically, we view the data measurements as observations at different time instants of a few processes that start at different nodes of an unknown graph and diffuse with time. Such data can be represented as the combination of graph heat kernels or, more generally, of localized graph kernels. Particularly the heat diffusion model can be widely applied in real world scenarios to understand the distribution of heat (sources) [5]. One example is the propagation of a heat wave in geographical spaces. Another example is the movement of people in buildings or vehicles in cities, which are represented on a geographical graph. Finally, a shift of people's interest towards certain subjects on social media platforms such as Twitter could also be understood via a heat diffusion model [6].

We then cast a new graph learning problem that aims at estimating a graph that best explains the data measurements under the heat diffusion model. Specifically, we represent our graph signals as a linear combination of a few (sparse) components from a graph dictionary consisting of heat diffusion kernels. The graph learning problem is then formulated as a regularized inverse problem where both the graph and the sparse coefficients are unknown. We propose a new algorithm to solve the resulting nonconvex optimization problem, which, under mild

Manuscript received November 4, 2016; accepted June 15, 2017. Date of publication July 24, 2017; date of current version August 14, 2017. The guest editor coordinating the review of this manuscript and approving it for publication was Guest editor Prof. Michael Rabbat. *(Corresponding author: Dorina Thanou.)*

D. Thanou is with the Swiss Data Science Center, EPFL/ETHZ, Lausanne 1015, Switzerland (e-mail: dorina.thanou@epfl.ch).

X. Dong is with the MIT Media Lab, Cambridge, MA 02139 USA (e-mail: xdong@mit.edu).

D. Kressner is with the Chair of Numerical Algorithms and High-Performance Computing, École Polytechnique Fédérale de Lausanne, Lausanne 1015, Switzerland (e-mail: daniel.kressner@epfl.ch).

P. Frossard is with the Signal Processing Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne 1015, Switzerland (e-mail: pascal.frossard@epfl.ch).

Digital Object Identifier 10.1109/TSIPN.2017.2731164

[1]In the extreme case, when the diffusion parameter tends to infinity, this model can capture globally smooth signals.

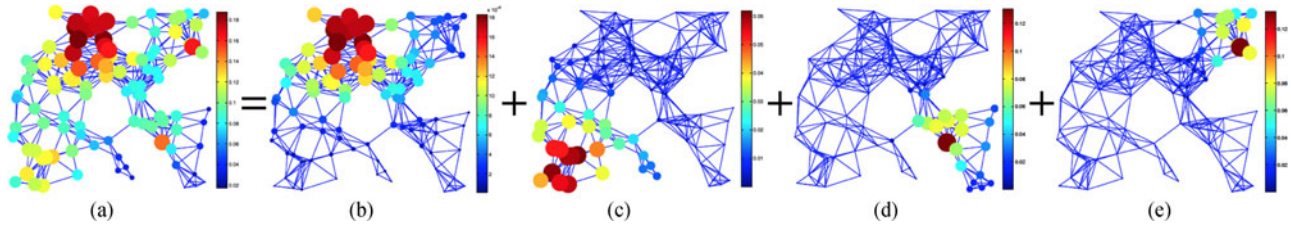


Fig. 1. Decomposition of a graph signal (a) in four localized simple components (b), (c), (d), (e). Each component is a heat diffusion process ($e^{-\tau \mathcal{L}}$) at time τ that has started from different network nodes (δ_n). The size and the color of each ball indicate the value of the signal at each vertex of the graph.

assumptions [7], guarantees that the iterates converge to a critical point. We finally provide a few illustrative experiments on synthetic data, as well as on two real world datasets that capture (i) the diffusion of tracers in atmospheric systems and (ii) the mobility patterns of Uber trips in New York City. The graph recovered from the first dataset correctly captures the trajectory of the chemical tracer, while the graphs learned from the Uber data reveal mobility patterns at different time intervals of the day across the city. The results confirm that the proposed algorithm is effective at inferring meaningful graph topologies in both synthetic and real world settings. Our framework is one of the first attempts to learn graphs carrying the structure of data that are not necessarily globally smooth but instead obey a more generic sparse model. We believe that this framework will prove particularly useful in the analysis of social and biological networks, for example, where the data structure is not immediately given by the application settings or design constraints.

The structure of the paper is as follows. We first highlight some related work on the learning of graph topologies in Section II. In Section III, we introduce our signal model and the structure of the diffusion dictionary. The graph learning algorithm is presented in Section IV. Finally, in Section V, we evaluate the performance of our algorithm for both synthetic and real world graph signals.

II. Related Work

A number of approaches have recently been proposed to learn the structure of data. Intense research efforts have been dedicated to methods for estimating covariance matrices (see, e.g., [8]), which carry information about the data structure. Richer structures can be estimated by learning data graphs instead of the mere covariance matrix. For example, the work in [3] learns a valid graph topology (the adjacency matrix) with an optimization problem that is very similar to sparse inverse covariance estimation, but it instead involves a regularized full-rank Laplacian matrix. Then, the authors in [4] relax the assumption of a full-rank matrix and propose to learn a valid graph Laplacian by imposing smoothness of observations on the graph. Thus, instead of focusing on pairwise-correlations between random variables, they explore the link between the signal model and the graph topology to learn a graph that provides a globally smooth representation of the corresponding graph signals. This framework has been extended further to yield a more scalable algorithm for learning a valid graph topology [9]. The authors in [10] propose an algorithm to estimate a generalized Laplacian matrix instead of the classical combinatorial or normalized Laplacian. Finally, manifold learning certainly represents another important class of works that aims at estimating the data structure and bears some similarity with the graph learning algorithms in some specific settings [11]. However, all the above works assume that the data evolve smoothly on the underlying structure, which is not necessarily the ideal model for all datasets.

The idea of recovering graph topologies for different graph signal models is relatively new and has not yet received a lot of attention. An autoregressive model that is based on graph filter dynamics is used in [12] to discover unknown relations among the vertices of a set of time series. The authors in [13] model the observations as being measured after a few steps of diffusing signals that are initially mutually independent and have independent entries. The diffusion process is modeled by powers of the normalized Laplacian matrix. They propose an algorithm for characterizing and then computing a set of admissible diffusion matrices, which relies on a good estimation of the covariance matrix from the independent signal observations. The exact point on the polytope is chosen using some specific criteria that are based either on the simplicity of the solution or the desired sparsity of the recovered graph. The problem of estimating a topology from signal observations that lead to particular graph shift operators is studied in [14]. The authors propose to learn a sparse graph matrix that can explain signals from graph diffusion processes, under the assumption that eigenvectors of the shift operators, i.e., the graph templates, are estimated from the covariance of the graph signals. The graph learning problem then becomes equivalent to learning the eigenvalues of the shift matrix, under the constraints that the shift operator is sparse. We will discuss the differences with this scheme in the experimental section. Contrary to the existing works, we learn a graph diffusion process without making any assumption on the eigenvectors of the graph process but instead make an explicit assumption on the diffusion process and the sparse signal model.

III. Sparse Representation of Graph Signals

A. Signal Representation

We consider a weighted and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$, where $\mathcal{V}$ and $\mathcal{E}$ represent the vertex (node) and edge sets of the graph, respectively. The $N \times N$ matrix W contains the edge weights, with $W_{ij} = W_{ji}$ denoting the positive weight

of an edge connecting vertices i and j, and $W_{ij} = 0$ if there is no edge. Without loss of generality, we assume that the graph is connected. The graph Laplacian operator is defined as $L = D - W$, where D is the diagonal degree matrix with the ith diagonal element equal to the sum of the weights of all edges incident to vertex i [1]. Being a real symmetric matrix, the graph Laplacian has an orthonormal basis of eigenvectors. We let $\chi = [\chi_0, \chi_1, \ldots, \chi_{N-1}]$ denote the eigenvector matrix of L. The diagonal matrix Λ contains the corresponding eigenvalues $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{N-1}$ on its diagonal.

A graph signal is a function $x : \mathcal{V} \to \mathbb{R}$ such that $x(v)$ is the value of the function at the vertex $v \in \mathcal{V}$. We consider the factor analysis model from [15] as our graph signal model, which is a generic linear statistical model that aims at explaining observations of a given dimension from a set of unobserved latent variables. Specifically, we consider

$$x = \mathcal{D}h + u_x + \epsilon, \tag{1}$$

where $x \in \mathbb{R}^N$ is the observed graph signal, $h \in \mathbb{R}^K$ is the latent variable that controls x, and $\mathcal{D} \in \mathbb{R}^{N \times K}$ is a representation matrix that linearly relates the two variables, with $K \geq N$. The parameter $u_x \in \mathbb{R}^N$ is the mean of x, which we set to zero for simplicity, and ϵ is a multivariate Gaussian noise with zero mean and covariance $\sigma_\epsilon^2 I_N$.

To represent signals residing on graphs, and especially to identify and exploit structure in the data, we need to take the intrinsic geometric structure of the underlying graph into account. This structure is usually incorporated in the columns of the representation matrix, i.e., atoms of a dictionary [16], [17]. These atoms carry spectral and spatial characteristics of the graph. Specifically, one can consider spectral graph dictionaries defined by filtering the eigenvalues of the graph Laplacian in the following way:

$$\mathcal{D} = [\widehat{g}_1(L)\, \widehat{g}_2(L) \ldots \widehat{g}_S(L)], \tag{2}$$

where $\{\widehat{g}_s(\cdot)\}_{s=1,\ldots,S}$ are graph filter functions defined on a domain containing the spectrum of the graph Laplacian. Each of these filters captures different spectral characteristics of the graph signals.

For efficient signal representation, the latent variables h should be sparse such that they reveal the core components of the graph signals [18]. In particular, one can impose a Laplace (sparse) prior on the latent variable h like

$$p(h) = \prod_i \alpha \exp(-\alpha |h(i)|), \tag{3}$$

where α is constant, and a Gaussian prior on the noise ϵ. Then the conditional probability of x given h can be written as

$$p(x|h) \sim \mathcal{N}(\mathcal{D}h, \sigma_\epsilon^2 I_N).$$

Given the observation x and the Laplace prior distribution of h in (3), we can compute a maximum a posteriori (MAP) estimate of the sparse set of components. Specifically, by applying Bayes' rule and assuming without loss of generality that $u_x = 0$, the MAP estimate of the latent variable h is [19]:

$$\begin{aligned} h_{\text{MAP}}(x) &:= \arg\max_h p(h|x) = \arg\max_h p(x|h)p(h) \\ &= \arg\min_h \left(-\log p_E(x - \mathcal{D}h) - \log p_H(h)\right) \\ &= \arg\min_h \|x - \mathcal{D}h\|_2^2 + \alpha \|h\|_1, \end{aligned} \tag{4}$$

where $\|\cdot\|_1$ denotes the ℓ^1-norm.

Sparsity-based inverse problems have been widely used in the literature to perform classical signal processing tasks on the observations x, such as denoising and inpainting. Sparsity however largely depends on the design of the dictionary, which itself depends on the graph. In the following, we discuss the choice of the representation matrix and the latent variables in our heat diffusion signal model.

B. Diffusion Signals on Graphs

In this paper, we focus on graph signals generated from heat diffusion processes, which are useful in identifying processes evolving nearby a starting seed node. In particular, the graph Laplacian matrix is used to model the diffusion of the heat throughout a graph or, more generally, a geometric manifold. The flow of the diffusion is governed by the following differential equation with initial conditions:

$$\frac{\partial x}{\partial \tau} - Lx = 0,\ x(v, 0) = x_0(v) \tag{5}$$

where $x(v, \tau)$ describes the heat at node v at time τ, beginning from an initial distribution of heat given by $x_0(v)$ at time zero. The solution of the differential equation is given by

$$x(v, \tau) = e^{-\tau L} x_0(v). \tag{6}$$

Going back to our graph signal model, the graph heat diffusion operator is defined as [20]

$$\widehat{g}(L) := e^{-\tau L} = \chi e^{-\tau \Lambda} \chi^T.$$

Intuitively, different powers τ of the heat diffusion operator correspond to different rates of heat flow over the graph. From the differential equation point of view, they correspond to different durations of the dynamics. If such operators are used to define a dictionary in (2), our graph signal model of (1) becomes

$$x = \mathcal{D}h + \epsilon = [e^{-\tau_1 L}\, e^{-\tau_2 L}\ \cdots\ e^{-\tau_S L}\,]h + \epsilon,$$

which is a linear combination of different heat diffusion processes evolving on the graph. For each diffusion operator $e^{-\tau_s L}$, the signal component $e^{-\tau_s L} h_s$ can also be interpreted as the result of filtering an initial graph signal h_s with an exponential, low-pass filter $e^{-\tau_s L}$ on the graph spectral domain. The obtained signal x is the sum of each of these simple components $x = \sum_{s=1}^{S} e^{-\tau_s L} h_s$. Notice that the parameter τ in our model carries a notion of scale. In particular, when τ is small, the ith column of $\mathcal{D}$, i.e., the atom $\mathcal{D}(i, :)$ centered at node i of the graph is mainly localized in a small neighborhood of i. As τ becomes larger, $\mathcal{D}(i, :)$ reflects information about the graph at a larger scale around i. Thus, our signal model can be seen as an additive model of diffusion processes that started at different

time instances. Finally, the sparsity assumption of (3) on the latent variables h implies that we expect the diffusion process to start from only a few nodes of the graph, at specific time instances, and spread over the entire graph over time.

IV. Learning Graph Topologies Under Sparse Signal Prior

In many applications, the graph is not necessarily known, and thus the MAP estimate of the latent variables in (4) cannot be solved directly. In the following, we show how the sparse representation model of the previous section can be exploited to infer the underlying graph topology, under the assumption that the signals are generated by a set of heat diffusion processes. First, we formulate the graph learning problem, and then we propose an efficient algorithm to solve it.

A. Problem Formulation

Given a set of M signal observations $X = [x_1, x_2, \ldots, x_M] \in \mathbb{R}^{N \times M}$, resulting from heat diffusion processes evolving on an unknown weighted graph $\mathcal{G}$, our objective is twofold: (i) infer the graph of N nodes by learning the graph Laplacian L, and (ii) learn, for each signal, the latent variable that reveals the sources of the observed processes, i.e., $H = [h_1, h_2, \ldots, h_M]$ and the diffusion parameters $\tau = [\tau_1, \tau_2, \ldots, \tau_S]$. As the graph Laplacian L captures the sparsity pattern of the graph, learning L is equivalent[2] to learning the graph $\mathcal{G}$. This results in the following joint optimization problem for H, L, and τ:

$$\begin{aligned} \underset{L,H,\tau}{\text{minimize}} \quad & \|X - \mathcal{D}H\|_F^2 + \alpha \sum_{m=1}^{M} \|h_m\|_1 + \beta \|L\|_F^2 \\ \text{subject to} \quad & \mathcal{D} = [e^{-\tau_1 L}\, e^{-\tau_2 L} \ldots e^{-\tau_S L}] \\ & \text{tr}(L) = N, \\ & L_{ij} = L_{ji} \leq 0,\ i \neq j, \\ & L \cdot \mathbf{1} = \mathbf{0}, \\ & \tau \geq 0, \end{aligned} \tag{7}$$

where h_m corresponds to the mth column of the matrix H. According to (4), the objective can be interpreted as the negative log-likelihood of the latent variables (columns of H) conditioned on the graph Laplacian L. The positive scalars α and β are regularization parameters, while $\mathbf{1}$ and $\mathbf{0}$ denote the vectors of all ones and zeros, respectively. In addition, $\text{tr}(\cdot)$ and $\|\cdot\|_F$ denote the trace and Frobenius norm of a matrix, respectively. The trace constraint acts as a normalization factor that fixes the volume of the graph and the remaining constraints guarantee that the learned L is a valid Laplacian matrix that is positive semidefinite. Note that the trace constraint, together with the other constraints, also fixes the ℓ^1-norm of L, while the Frobenius norm is added as a penalty term in the objective function to control the distribution of the off-diagonal entries in L, that is, the edge weights of the learned graph. For a small α, a big β implies a small Frobenius norm, which, together with the trace constraint on the graph Laplacian, lead to a Laplacian matrix with many non-zero entries that are similar to each others. The influence of the parameters α, β will be analyzed in the experimental section. When H is fixed, the optimization problem bears similarity to the linear combination of ℓ^1 and ℓ^2 penalties in an elastic net regularization [21], in the sense that the sparsity term is imposed by the trace constraint. When L, τ are fixed, problem (7) becomes equivalent to a MAP estimator, as discussed in the previous subsection.

Note that our problem formulation depends on the number of blocks S, i.e., the number of scales of the diffusion processes. The choice of S depends on the training signals, in particular, on the number of scales that one can detect in the training data. As we expect the diffusion processes to be localized, we typically choose a small value for S, say, 1 to 3. One of the blocks would correspond to a very small scale (i.e., highly localized atoms), and the other blocks would capture larger scale, but still somewhat localized patterns.

The optimization problem (7) is nonconvex with respect to L, H, τ simultaneously. In particular, the data fidelity term $\|X - \mathcal{D}H\|_F^2$ is smooth but nonconvex as it contains the product of the three matrix variables (e.g., $e^{-\tau L} H$). As such, the problem may have many local minima and solving it is hard. One could apply alternating minimization, where at each step of the alternation we update one variable by fixing the rest. This, however, does not provide convergence guarantees to a local minimum and, moreover, solving the problem with respect to L is difficult due to the matrix exponential, which makes the problem nonconvex even when τ, H are fixed. In the next section, we propose an effective algorithm to solve the graph learning problem, which is not affected by this difficulty.

B. Graph Learning Algorithm

In order to solve (7), we apply a proximal alternating linearized minimization algorithm (PALM) [7], which can be interpreted as alternating the steps of a proximal forward-backward scheme [22]. PALM is a general algorithm for solving a broad class of nonconvex and nonsmooth minimization problems, which, under mild assumptions [7], guarantees that the iterates converge to a critical point. Moreover, it does not require convexity of the optimization problem with respect to each variable separately. The basis of the algorithm is alternating minimization between the three variables (L, H, τ), but in each step we linearize the nonconvex fitting term $\|X - \mathcal{D}H\|_F^2$ with a first order function at the solution obtained from the previous iteration. In turn, each step becomes the proximal regularization of the nonconvex function, which can be solved efficiently. More specifically, the algorithm consists of three main steps: (i) update of H, (ii) update of L, (iii) update of τ, and inside each of these steps we compute the gradient and estimate the Lipschitz constant with respect to each of the variables. Algorithm 1 contains a summary of the basic steps of PALM adapted to our graph learning problem.

[2]Since our graph does not contain self-loops, the weight matrix W of the graph can be simply computed as $W = -L$, and then setting the diagonal entries to zero.

Algorithm 1: Learning heat kernel graphs (**LearnHeat**).

1: **Input:** Signal set X, number of iterations iter
2: **Output:** Sparse signal representations H, graph Laplacian L, diffusion parameter τ
3: **Initialization:** $L = L^0$, $\mathcal{D}^0 = [e^{-\tau_1 L}\ e^{-\tau_2 L} \dots e^{-\tau_S L}]$, $\tau = \tau^0$
4: **for** $t = 1, 2, \dots,$ iter **do:**
5: Choose $c_t = \gamma_1 C_1(L^t, \tau^t)$
6: Update H^{t+1} by solving opt. problem (9)
7: Choose $d_t = \gamma_2 C_2(H^{t+1}, \tau^t)$
8: (a) Update L^{t+1} by solving opt. problem (11)
9: (b) Update $\mathcal{D}^{t+1} = [e^{-\tau_1^t L^{t+1}} \dots e^{-\tau_S^t L^{t+1}}]$
10: Choose $e_t = \gamma_3 C_3(L^{t+1}, H^{t+1})$
11: (a) Update τ^{t+1} by solving opt. problem (12)
12: (b) Update $\mathcal{D}^{t+1} = [e^{-\tau_1^{t+1} L^{t+1}} \dots e^{-\tau_S^{t+1} L^{t+1}}]$
13: **end for**
14: $L = L^{\text{iter}}, H = H^{\text{iter}}, \tau = \tau^{\text{iter}}$.

In the following, we explain in detail each of the steps of Algorithm 1. We make use of the following definitions:

$$Z(L, H, \tau) = \|X - \mathcal{D}H\|_F^2, \quad f(H) = \alpha \sum_{m=1}^{M} \|h_m\|_1,$$

$$g(L) = \delta(L|\mathcal{C}) + \beta\|L\|_F^2,$$

where δ is an indicator function for the convex set $\mathcal{C} = \{\text{tr}(L) = N, L_{ij} = L_{ji} \leq 0, i \neq j, L \cdot \mathbf{1} = \mathbf{0}\}$, defined as

$$\delta(L|\mathcal{C}) = \begin{cases} 1, & \text{if } L \in \mathcal{C} \\ +\infty, & \text{otherwise.} \end{cases}$$

1) Update of H (Algorithm 1: Lines 5-6): For iteration $t+1$ of the sparse coding update step, we apply the first step of the PALM algorithm that requires the proximal regularization of the nonconvex function $Z(L, H, \tau)$, linearized at (L^t, H^t, τ^t):

$$H^{t+1} = \text{prox}_{c_t}^f \left(H^t - \frac{1}{c_t} \nabla_H Z(L^t, H^t, \tau^t) \right)$$

where L^t, H^t, τ^t are the updates obtained at iteration t and c_t is a positive constant. $\text{prox}_{c_t}^f$ is the proximal operator [23] of the convex function $f(H)$ with parameter c_t, given by

$$\text{prox}_{c_t}^f(G) = \text{sign}(G)\max(|G| - \alpha/c_t, 0), \tag{8}$$

with all operations understood elementwise and $G = H^t - \frac{1}{c_t}\nabla_H Z(L^t, H^t, \tau^t)$. We note that the soft thresholding operator is applied on each column of the matrix H. From the definition of the proximal operator, this step is equivalent to solving the following optimization problem:

$$H^{t+1} = \underset{H}{\text{argmin}} \langle H - H^t, \nabla_H Z(L^t, H^t, \tau^t) \rangle + \frac{c_t}{2}\|H - H^t\|_F^2 + f(H), \tag{9}$$

where we recall that the definition of the inner product between two matrices is $\langle H - H^t, \nabla Z_H(L^t, H^t, \tau^t) \rangle = \text{tr}\big(\nabla Z_H(L^t, H^t, \tau^t)^T (H - H^t)\big)$. The required gradient of $Z(L, H, \tau) = \|X - \mathcal{D}H\|_F^2$ with respect to H is computed in Appendix A-A. The parameter c_t is defined such that $c_t = \gamma_1 C_1(L^t, \tau^t)$, with $\gamma_1 > 1$ and the Lipschitz constant $C_1(L^t, \tau^t)$ of $\nabla_H Z(L^t, H, \tau^t)$ with respect to H, as derived in Appendix B-A.

2) Update of L (Algorithm 1: Lines 7-9): Similarly, the graph update step, that is the second step of PALM, is performed by

$$L^{t+1} = \text{prox}_{d_t}^g \left(L^t - \frac{1}{d_t} \nabla_L Z(L^t, H^{t+1}, \tau^t) \right), \tag{10}$$

with $d_t = \gamma_2 C_2(H^{t+1}, \tau^t)$ for some $\gamma_2 > 1$ and the estimate $C_2(H^{t+1}, \tau^t)$ of the Lipschitz constant of $\nabla_L Z(L^t, H^{t+1}, \tau^t)$ described in Appendix B-B. Given that $g(L) = \delta(L|\mathcal{C}) + \beta\|L\|_F^2$ comprises a quadratic term constrained in a convex polytope, the proximal minimization step (10) is a quadratic program (QP) that can be written as:

$$\begin{aligned} \underset{L}{\text{minimize}} \quad & \langle L - L^t, \nabla_L Z(L^t, H^{t+1}, \tau^t) \rangle \\ & + \frac{d_t}{2}\|L - L^t\|_F^2 + \beta\|L\|_F^2 \\ \text{subject to} \quad & \text{tr}(L) = N, \\ & L_{ij} = L_{ji} \leq 0,\ i \neq j, \\ & L \cdot \mathbf{1} = \mathbf{0}. \end{aligned} \tag{11}$$

This requires the gradient of $Z(L, H, \tau) = \|X - \mathcal{D}H\|_F^2$ with respect to L, the derivation of which can be found in Appendix A-B. Given this gradient, the optimization problem (11) can be solved using operator splitting methods, such as the alternating direction method of multipliers (ADMM) [24]. In this paper, we solve the problem by using the algorithm proposed in [25], which converts the problem to a convex cone optimization problem, and utilizes ADMM to solve the homogenous self-dual embedding. Compared to other methods, this approach finds both primal and dual solutions of the problem, is free of parameters, and scales to large problem sizes.

3) Update of τ (Algorithm 1: Lines 10-12): Finally, we can update the diffusion parameters $\tau = [\tau_1, \tau_2, \dots, \tau_S]$ following the same reasoning as above. The corresponding proximal splitting step is

$$\tau^{t+1} = \text{prox}_{e_t}^{\delta_\tau} \left(\tau^t - \frac{1}{e_t} \nabla_\tau Z(L^{t+1}, H^{t+1}, \tau^t) \right)$$

where $e_t = \gamma_3 C_3(H^{t+1}, L^{t+1})$, with $\gamma_3 > 1$ and the Lipschitz constant $C_3(H^{t+1}, L^{t+1})$ computed in Appendix B. $\delta_\tau(\tau)$ is an indicator function defined as follows:

$$\delta_\tau(\tau) = \begin{cases} 1, & \text{if } \tau \geq 0 \\ +\infty, & \text{otherwise.} \end{cases}$$

The optimization problem can be written as

$$\begin{aligned} \underset{\tau}{\text{minimize}} \quad & \langle \tau - \tau^t, \nabla_\tau Z(L^{t+1}, H^{t+1}, \tau^t) \rangle + \frac{e_t}{2}\|\tau - \tau^t\|_F^2 \\ \text{subject to} \quad & \tau \geq 0. \end{aligned} \tag{12}$$

This problem has a closed form solution given by

$$\tau^{t+1} = \max\left(-\frac{\nabla_\tau Z(L^{t+1}, H^{t+1}, \tau^t) - e_t \tau^t}{e_t}, 0\right), \quad (13)$$

with the gradient computed in Appendix A-C. Finally, we note that if we have an a priori estimate of the diffusion parameters τ (e.g., from the training phase) then we solve our optimization problem with respect to L, H by following the first two steps of our algorithm.

C. Discussion on the Computational Complexity

In the following, we discuss the computational complexity of our graph learning algorithm. Dealing with the heat diffusion processes $e^{-\tau_s L}$ represents one of the main computational bottlenecks. Both, the computation of the matrix exponential via a spectral decomposition or via the scaling and squaring method [26] as well as the computation of its gradient described in Appendix A-B require $\mathcal{O}(N^3)$ operations. Thus, this part of the algorithm can be expected to become time consuming for very large graphs. One way to reduce this cost is to approximate the heat diffusion kernel with a polynomial of degree K, reducing the complexity of applying a heat diffusion process to $\mathcal{O}(|\mathcal{E}|K)$, where $|\mathcal{E}|$ is the number of edges of the graph Laplacian. Since we generally consider heat diffusion processes that remain well localized, the degree K will typically be small. This approximation of the heat diffusion process is particularly efficient when the graph Laplacian is sparse. Also, it can be expected that the complexity of the gradient computation greatly reduces when using a polynomial approximation of the kernel; see [27] for some recent work in this direction. A detailed investigation of this aspect is part of our future work.

We further note that the computational complexity of the sparse coding step (lines 5-6 of the Algorithm) is dominated by the cost of computing the Lipschitz constant (see Appendix A-A), which requires the computation of the product $\mathcal{D}^T\mathcal{D}$ and is of order $\mathcal{O}(S^2N^3)$. Again, this cost greatly reduces when using a polynomial approximation of the kernel. In particular, the term $\mathcal{D}\mathcal{D}^T X$ can be computed in a fast way by exploiting the fact that $\mathcal{D}\mathcal{D}^T X = \sum_{s=1}^{S} \widehat{g}_s^2(L)X$, which leads to a polynomial of degree $2K$. Thus, the computational cost can be reduced to the one of the iterative sparse matrix-vector multiplication [17]. The update of the sparse codes in (9) requires $\mathcal{O}(N^2S)$ operations. Finally, the update of the graph Laplacian (Algorithm 1: lines 7-9) consists of three steps: the recursive approximation of the Lipschitz constant (Appendix B-B), the computation of the gradient discussed above and the solution of the optimization problem (11). The solution of (11) involves three main steps [25], among which the most expensive one is solving a linear system. For large scale systems, this can be done efficiently by applying a conjugate gradient method. Finally, the computation of the Lipschitz constant in the update of τ (see Appendix B-C) requires the computation of the spectral norm of L, which can be estimated in $\mathcal{O}(|\mathcal{E}|)$ operations by a few steps of the power or Lanczos method [28].

V. Experiments

In this section, we evaluate the performance of the proposed algorithm in both synthetic and real world experiments. We solve the optimization problem of (11) using ADMM, which is implemented with the splitting conic solver [25], a numerical optimization package for solving large-scale convex cone problems.[3] As a termination criteria, we stop the algorithm when a maximum number of iterations (set to 1000 in our experiments) is reached or the absolute difference in the value of the objective function at two consecutive iterations is smaller than 10^{-4}. Whenever a groundtruth graph is available, in order to evaluate quantitatively the performance of our learning algorithm in recovering the edges of the groundtruth graph, we report the *Precision*, *Recall*, *F-measure* and *Normalized Mutual Information (NMI)* [29] scores, as well as the difference in terms of the Frobenius norm of the edge weights. Specifically, the *Precision* evaluates the percentage of correct edges in the learned graph, that is, the edges that are present in the groundtruth graph. The *Recall* evaluates the percentage of the edges in the groundtruth graph that are present in the learned graph. The *F-measure* thus takes into account both *Precision* and *Recall* to measure the overall accuracy of the obtained edge set, and it is defined as

$$\text{F-measure} = 2\,\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Finally, the *NMI* measures the mutual dependence between the obtained edge set and that of the groundtruth graph from an information theoretic viewpoint.

A. Results on Synthetic Data

1) Simulation Settings: We first test the performance of the learning algorithm by comparing the learned graph to the one from the groundtruth in synthetic datasets. We evaluate the performance of the algorithm on random graphs of $N = 20$ vertices, generated from three different models: the radial basis function (RBF) random graph, the Barabási-Albert model (BA) [30], and the Erdős-Rényi model (ER) [31]. In the case of the RBF graph, we generate the coordinates of the vertices uniformly at random in the unit square, and we set the edge weights based on a thresholded Gaussian kernel function so that $W(i,j) = e^{-\frac{[\text{dist}(i,j)]^2}{2\sigma^2}}$ if the distance between vertices i and j is less than or equal to κ, and zero otherwise. We further set $\sigma = 0.5$ and $\kappa = 0.75$ in our experiments. In the ER graph, an edge is included with probability 0.2 independently of the other edges. Finally, in the BA graph, we add vertices one after the others and connect them to existing vertices following a preferential attachment mechanism. Given the adjacency matrix of each type of graph, we finally compute the graph Laplacian and we normalize in such a way that its trace is equal to N.

With the above model-based graphs, we then construct synthetic graph signals as follows. We use the graph Laplacian to generate an oracle dictionary of the form $\mathcal{D} = [e^{-\tau_1 L}\, e^{-\tau_2 L}]$, with $\tau_1 = 2.5, \tau_2 = 4$, for the RBF and the ER graph and $\tau_1 = 1, \tau_2 = 4$ for the BA model. These values are chosen in

[3]The conic solver can be found at https://github.com/cvxgrp/scs

such a way that our dictionaries contain two patterns that are sufficiently distinct from each other. In particular, the one corresponding to a small τ captures a very localized pattern while the one corresponding to a large τ captures a diffusion that has already spread in the local neighborhood of the vertex. We then generate 100 graph signals by linearly combining three random atoms from the dictionary with random coefficients drawn from a Gaussian distribution with zero mean and unit variance. The sparse vector of random coefficients represents the initial heat on the graph. We finally hide the graph structure, and apply Algorithm 1 with different sets of parameters α, β. In particular, we choose different powers of 10 ranging from 1 to -6, with a stepsize of -0.5 for α, and from 0 to -2, with a stepsize of -1 for β. For each pair of parameters, we estimate the graph only from the signal observations. The initialization of the graph Laplacian is done with a random valid Laplacian matrix. We compare our algorithm (LearnHeat) with the following three methods: (i) the algorithm proposed in [14], which is based on learning the graph Laplacian from diffusion filters and does not have any assumption on the global smoothness of the signals, (ii) the algorithm in [4] which recovers a graph Laplacian by assuming global smoothness of the signals on the graph, and (iii) the algorithm in [13] which infers a normalized graph Laplacian that belongs to the polytope of admissible solutions for diffusion matrices. We solve the algorithm in [14] for different values of the parameter $\epsilon = [0 : 0.02 : 2]$, where ϵ controls the imperfection of the spectral templates estimated from the covariance matrix, and provides a constraint on how close the optimized matrix should be to these templates. We threshold the learned Laplacian matrices by ignoring entries whose absolute values are smaller than 10^{-4}. For the algorithm of [13], we select the graph on the polytope under the simplicity criterion that is defined in the corresponding paper. This criterion encourages the recovery of eigenvalues that lead to a diffusion matrix with empty diagonal.

2) Graph Learning Performance: We first compare visually the learned graph Laplacian for an RBF graph model with the corresponding groundtruth one. The results illustrated in Fig. 2 are the ones obtained for the pair of α and β that leads to the best quantitative results (see below), and the best ϵ for [14]. First, we consider the noiseless case of clean training signals (rows 1-2). We observe that both the graph Laplacians learned with the proposed algorithm and the diffusion filters of [14] are visually consistent with the groundtruth Laplacian, reaching an *F-measure* score [29] of 0.9784 and 0.9927 respectively. On the other hand, the performance of [4] that is based on a smooth signal model is worse in terms of *F-measure* score (0.9173). This is quite expected as, when the diffusion parameter τ is relatively small, the signals generated by heat diffusion processes consist mainly of localized, piecewise smooth components that can be better captured with the other two algorithms. A globally smooth signal model can help recovering parts of the graph, but is not accurate enough to reveal the true topology. However, as τ increases the diffusion tends to a steady state that is a smooth signal on the graph. In that case, the behavior of our algorithm is expected to be close to the one in [4]. Finally, the algorithm of [13] generates results that are less consistent with the groundtruth graph. However, we should note that the algorithm is designed for estimating the normalized graph Laplacian, which means that the signal model is not consistent with out training signals. In addition, since the objective of the algorithm is to find a matrix that belongs to the admissible set of the diffusion matrices, due to this assumption the algorithm is more constrained, and in general more training signals are needed to give a good recovery performance. On the other hand, the other algorithms, including the proposed one, do not necessarily guarantee that the matrix belongs to this set.

In the second set of experiments, we test the sensitivity of the three algorithms to noise on the training signals. In particular, we add some white noise with zero mean and variance 0.02 to our training signals, leading to a signal to noise ratio of approximately 13 dB. In the second row of Fig. 2 , we observe that LearnHeat is quite resilient to the noise, reaching an *F-measure* score of 0.9552 and an error of 0.2642 in terms of the Frobenius difference of the edge weights compared to the groundtruth ones. The performance of [14] seems to deteriorate significantly due to the noise, achieving an *F-measure* score of 0.8451 and error weight of 0.3546. This is quite expected as the algorithm is based on the estimation of the eigenvectors of the Laplacian, which depends on the covariance of the noisy training set. The performance of [4] deteriorates too but less significantly, as this algorithm contains a term in the optimization problem that performs denoising of the training signals. Finally, the algorithm of [13] seems to be able to recover only the strongest edges of the topology.

In order to evaluate quantitatively the performance of our learning algorithm in recovering the edges of the groundtruth graph, we report the *Precision*, *Recall*, *F-measure* and *Normalized Mutual Information (NMI)* [29] scores, as well as the difference in terms of the Frobenius norm of the edge weights, averaged over ten random instances of three graph models with their corresponding 100 signal observations. For computing the *NMI*, we first compute a 2-cluster partition of all the vertex pairs using the learned graph, based on whether or not there exists an edge between the two vertices. We then compare this partition with the 2-class partition obtained in the same way using the groundtruth graph. Since more training signals are needed for [13], we skip these results in this comparison. The LearnHeat results shown in Tables I, II are the ones corresponding to the best combinations of α and β in terms of *F-measure* for noiseless and noisy training signals respectively, while the results of [14] are the ones obtained for the constant ϵ that gives the best *F-measure*. These results confirm that our algorithm is able to learn graph topologies that are very similar to the groundtruth ones and its performance is quite robust to noise. The algorithm of [14] seems to perform very well in the noiseless case. However, its performance deteriorates significantly in some of the noisy cases (i.e., RBF graph). The reason for that is that this algorithm makes no assumptions on the underlying graph process, apart from the fact that the process is a filter of an unknown graph matrix. On the other hand, our algorithm uses the knowledge of the heat diffusion process to limit the effect of the noise in the data. As expected, the worst performance is observed in [4]. Since our training signals consist of localized heat diffusion patterns, the performance of [4] is significantly

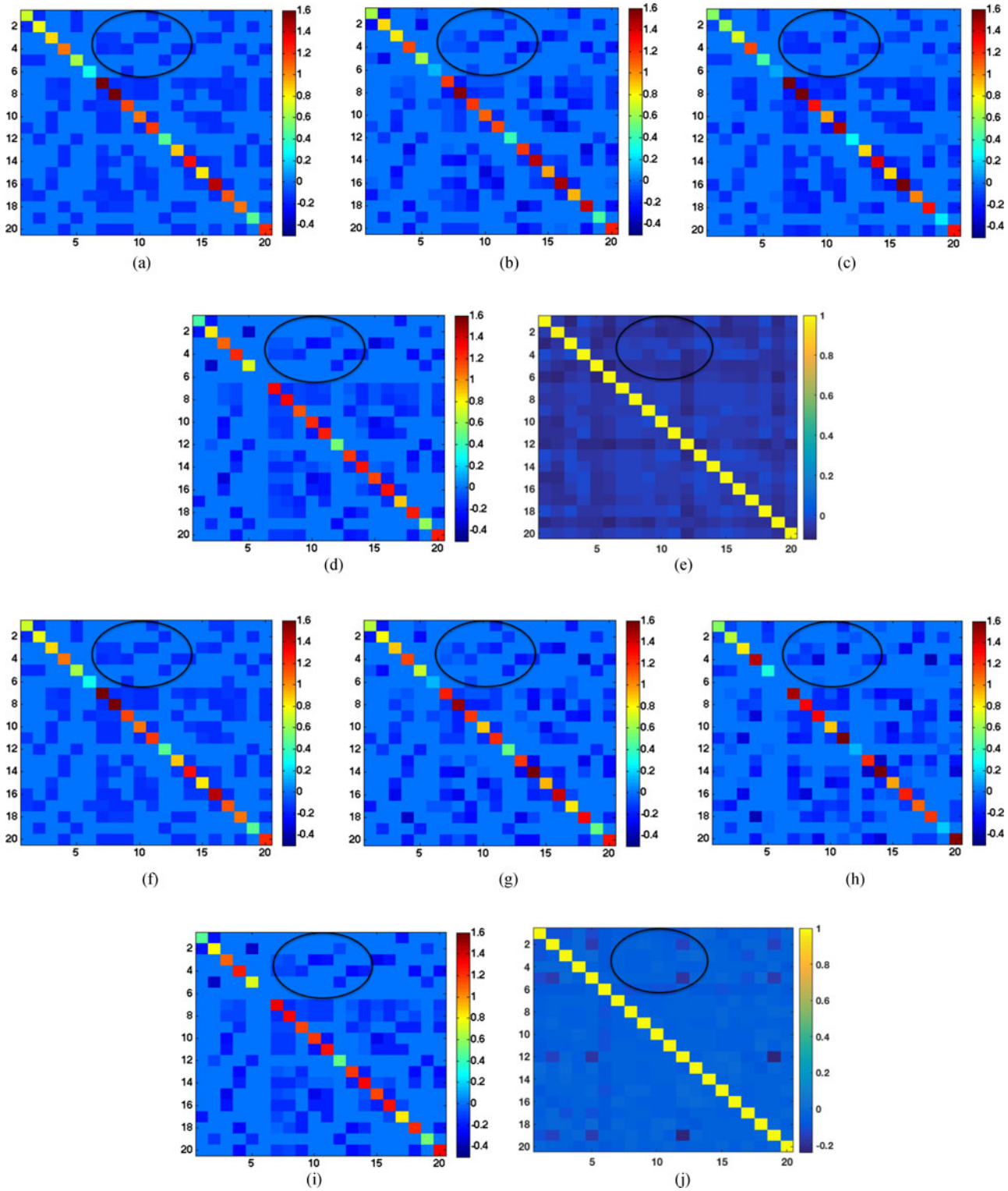


Fig. 2. The learned graph Laplacian matrices for a Gaussian RBF graph. The color indicates the values of the entries of the graph Laplacians. The first two rows illustrate the groundtruth Laplacian and the Laplacians recovered with LearnHeat, the algorithm of [14], the smooth signal model of [4], and the diffusion learning model of [13] when the training signals are clean. The other two rows illustrate the same results obtained from noisy training signals.

penalized since it is rather designed for signals that have global smoothness properties.

3) Algorithm Analysis: To understand the effect of the number of the training signals in the learning performance, we run a set of experiments on some clean training signals. In Fig. 3, we illustrate the best *F-measure* score achieved for a training set of size $M = [2, 20, 200, 2000, 20000]$ for the algorithms proposed in [14], [4] and [13]. For a fairer comparison, we compute the number of edges in the groundtruth topology and for each of the learned graph, we keep only the highest in

TABLE I
GRAPH LEARNING PERFORMANCE FOR CLEAN DATA

Graph model	F-measure	Precision	Recall	NMI	ℓ^2 Weight Error
Gaussian RBF (LearnHeat)	0.9779	0.9646	**0.9920**	0.8977	0.2887
Gaussian RBF [14]	**0.9911**	**0.9905**	0.9919	**0.9550**	**0.2081**
Gaussian RBF [4]	0.8760	0.8662	0.8966	0.5944	0.4287
ER (LearnHeat)	**0.9303**	**0.8786**	**0.9908**	**0.7886**	**0.3795**
ER [14]	0.8799	0.8525	0.9157	0.65831	0.3968
ER [4]	0.7397	0.6987	0.8114	0.4032	0.5284
BA (LearnHeat)	**0.9147**	**0.8644**	**0.9757**	**0.7538**	0.4009
BA [14]	0.8477	0.7806	0.9351	0.6009	**0.3469**
BA [4]	0.6969	0.6043	0.8459	0.3587	0.5880

TABLE II
GRAPH LEARNING PERFORMANCE FOR NOISY DATA

Graph model	F-measure	Precision	Recall	NMI	ℓ^2 Weight Error
Gaussian RBF (LearnHeat)	**0.9429**	**0.9518**	**0.9355**	**0.7784**	**0.3095**
Gaussian RBF [14]	0.8339	0.8184	0.8567	0.5056	0.3641
Gaussian RBF [4]	0.8959	0.7738	0.9284	0.5461	0.4572
ER (LearnHeat)	**0.8217**	0.7502	**0.9183**	**0.5413**	**0.3698**
ER [14]	0.8195	**0.7662**	0.8905	0.5331	0.3809
ER [4]	0.6984	0.5963	0.8690	0.3426	0.5172
BA (LearnHeat)	0.8155	0.7503	0.8986	0.5258	0.4036
BA [14]	**0.8254**	**0.7613**	**0.9068**	**0.5451**	**0.3980**
BA [4]	0.7405	0.6800	0.8230	0.3980	0.5899

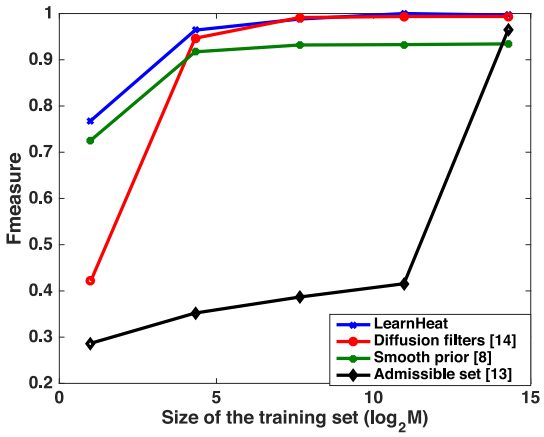
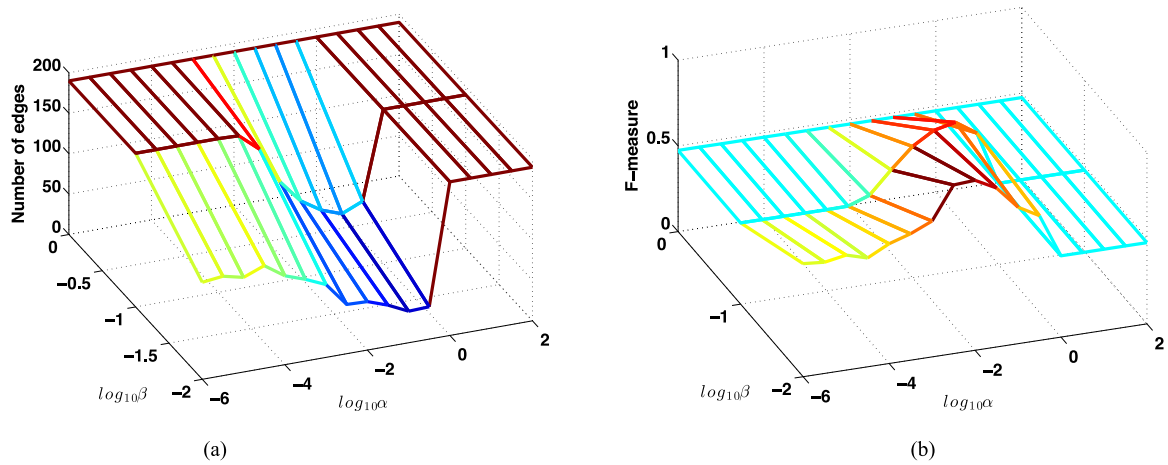


Fig. 3. Dependence of the *F-measure* on the size of the training set (in a logarithmic scale) for the four different algorithms i.e., LearnHeat, Diffusion Filters [14], smooth prior [4], and the topology inferred with admissibility constraints [13].

magnitude entities such that the total number corresponds to the number of edges of the groundtruth graph. We then compute the *F-measure* score based on the thresholded learned matrix. We observe that the performance of all four algorithms under study depends on the training set. However, for a very small size of the training set, our algorithm seems to outperform the others. In that regime, the recovery performance from the diffusion filters [14] depends on the estimation of the spectral templates, which is highly dependent on the number of the training samples. Although this approach is quite interesting and works very well when the training set is large and the estimation of the covariance matrix is accurate, it might face some limitations when the training set is limited and noisy. Similar behavior is shown also by the algorithm proposed in [13]. In contrary, our algorithm learns a graph diffusion process without making any assumption on the eigenvectors of the graph process: it rather sets an explicit assumption on the (heat) diffusion process and the signal model. Moreover, our sparsity assumption imposes additional structure to the problem, leading to high recovery performance even when the training set is limited.

We now study the effect of the parameters α and β in the objective function of (7). We illustrate in Fig. 4 the number of edges of the learned graph and the *F-measure* score under different combinations of these parameters, for a random instance of the Gaussian RBF graph. The obtained results indicate that for very large or very small values of α, the influence of β is limited and the performance of the algorithm in terms of both the number of edges and the *F-measure* is mainly determined by the sparsity control parameter α. For the remaining values of α, the value of β is linked to the sparsity of the learned graph; an optimal β would lead to a graph that has a similar level of sparsity to the groundtruth graph, hence maximizes the learning performance. In particular, for a fixed α, the number of learned edges decreases as β decreases. A big β implies a small Frobenius norm, which leads to a Laplacian matrix with many non-zero entries that are similar to each other. Thus, the correct value of β is determined by the true sparsity of the underlying graph. Then, in order to understand the effect of the parameter α, we need to distinguish the following three cases. When α is relatively small, due to a constraint on the trace (effectively the ℓ^1-norm) of the Laplacian matrix, the algorithm does not allow



(a) (b)

Fig. 4. (a) The number of edges in the learned graph, and (b) the *F-measure* score, under different combinations of the parameters α and β for an instance of the Gaussian RBF graph.

to learn an empty graph. In fact, the penalty on ℓ^1-norm on H becomes negligible due to a small α, and the graph is being optimized under the constraint of a fixed ℓ^1-norm and a penalty on the ℓ^2-norm. This would lead to a fitting term that is almost negligible. Thus, the solution of the optimization problem is mainly determined by the second and the third term of the objective function of (11) and the Laplacian constraints. In this case, a β sufficiently large would promote small and similar off-diagonal entries in the Laplacian matrix. As we increase α, we observe in Fig. 4 that the number of edges decreases, and the learned graph becomes similar to the groundtruth one as indicated by the *F-measure* score. In particular, there exists a range of values for the parameter α where the learned graph reaches a number of edges that is similar to the one of the true graph, and the *F-measure* reaches its peak. Alternatively, when the value of α is relatively big, the solution of the sparse coding step tends to give a matrix H that is sparser. In that case, the algorithm tries to express the signals in the dense matrix X as a heat diffusion process starting from some sparse initial heat sources H. We recall that the heat kernel can be written as the Taylor expansion of the exponential function $e^{-\tau L} = \sum_{k=0}^{\infty}(-\tau)^k \frac{L^k}{k!}$. Moreover, the kth power of the Laplacian is localized in the k-hop neighborhood of a node n, i.e., $(L^k)_{n,m} = 0$ if nodes n and m are not connected with a path of at least k-hops on the graph [32]. Thus, the initial heat h, corresponding to an observation x, diffuses all over the graph only if there exists a finite path connecting the sources indicated in h with the other nodes of the graph. As a result, in order to approximate a dense observation x, the graph that we learn should be more connected. In the extreme case when H is a zero matrix, the objective function penalizes only the Frobenius norm of L. The latter explains the tendency of the algorithm to favor complete graphs with similar entries when α is large.

4) Source Localization: In a final set of experiments, we illustrate the performance of the learned diffusion dictionary in terms of source localization. For the sake of simplicity, we focus on the case of only one dictionary block. In particular, we use the different instances of the learned topologies with our scheme for an RBF Gaussian graph model. We then use the learned graphs to solve a sparse inverse problem, similar to (4), to recover the sources from a set of some other signal observations. For each value of the parameter $\tau = [10^{-1} : 10^{0.5} : 10^{1.5}]$, we generate one diffusion dictionary per topology instance. Each of these dictionaries are used to generate a set of 1000 testing signals that are each a linear combination of 3 atoms of the corresponding dictionary, generated in the same way as the training signals. The location of these atoms defines the initial sources of the process. We aim at recovering the initial sources by solving an iterative soft thresholding algorithm [33] with the diffusion dictionary on a learned graph.

In Fig. 5, we show the source recovery performance for different values of the parameter τ. In particular, in Fig. 5(a), we illustrate the average *F-measure* score between the groundtruth sparse codes of the testing signals and the recovered ones as a function of τ. We observe that the *F-measure* is high when τ is low. The latter is intuitive as a small τ implies that the diffusion process is quite localized around the initial sources, leading to an easier recovery. As τ increases the performance reduces significantly, as the diffusion process tends towards a smooth signal on the graph. Thus, recovering the sources becomes more difficult. We notice that the recovery performance is measured in terms of the activation of the sources, i.e., the non-zero position of the learned sparse codes, and not the actual value. In order to understand better the source localization performance with a sparse prior, we keep only the $s = 3$ highest ones in terms of magnitude values of the sparse codes, where s is the number of the initial sources. We then illustrate in Fig. 5(b) the average number of recovered sources for different values of the parameter τ, for the sparsity parameter α of (4) that gives the best source recovery results. These results are consistent with the previous ones and they confirm that when τ is low, the location of the sparse codes with higher magnitude refers to the initial sources.

5) Discussion on the Performance of the Algorithm: We note that the algorithm proposed in this paper can be seen as a sparse dictionary learning algorithm with a dictionary that has some

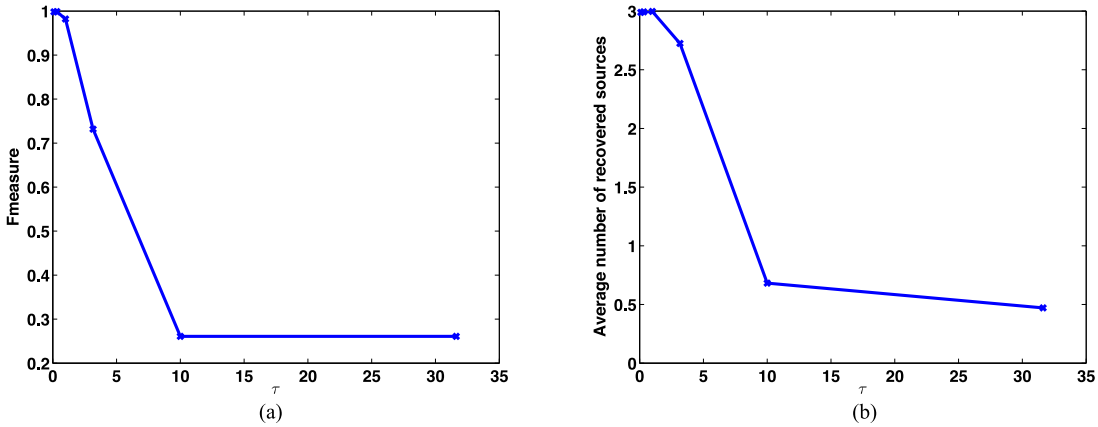


Fig. 5. Source recovery performance measured with respect to (a) the *F-measure* score between the recovered and the groundtruth sparse codes and (b) the location of the highest in magnitude sparse codes coefficients for different values of the diffusion parameter τ, and three initial sources.

predefined structure, defined by the heat diffusion kernel. In the previous simulation results, we have studied the effect of the number of the training signals and the sparsity of the signals on the underlying dictionary. Here, we discuss another important parameter, that is the coherence of the dictionary.

Given the heat diffusion structure, the coherence of the atoms depends on the connectivity of the graph and the diffusion parameter τ. These two parameters define the support of the dictionary atoms, i.e., their spread on the graph. In particular, for the same value of τ, the denser the graph, the faster the heat is expected to diffuse on the network. This however leads to a dictionary with coherent atoms, which might penalize the performance of the sparse coding step. On the other hand, when the graph is sparser, and τ not very large, the coherence of the atoms is significantly lower. The choice of the graph sparsity thus depends on the distribution of the data. If our signals consists of very localized patterns, the diffusion of the heat on a sparser graph would be able to capture these patterns, favoring sparser representations of the signals.

For a fixed graph connectivity, the values of the diffusion parameters τ and their relative difference within the different subdictionaries also play a significant role. When the atoms from each subdictionary are quite different from each other (the diffusion patterns are sufficiently distinct), we expect the *F-measure* score to be high. On the other hand, when the diffusion parameters of each subdictionary generate atoms that are similar to each other, both subdictionaries contain similar information, leading to a high coherence of the dictionary. As a result, the sparse coding step, which is one of the two steps of the graph learning algorithm, tends to fail, and the recovery performance of the graph deteriorates significantly.

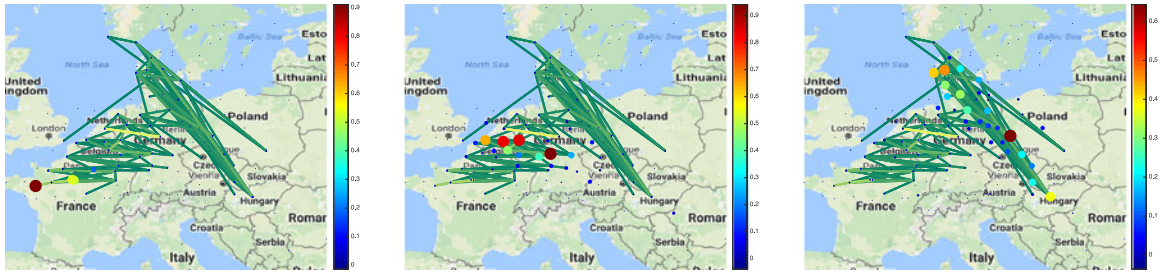
B. Graph Learning in Real-World Datasets

1) ETEX Dataset: We now illustrate the performance of our algorithm on real world datasets and in particular in the application of graph signal representation using learned graph topologies. We first consider data from the European Tracer Experiment (ETEX), which took place in 1994 [34].[4] The experiment consists in injecting a particular gas, namely the tracer, into the atmospheric system and then in observing the evolution of the tracer with a variety of sampling and analysis tools. Tracer techniques are widely applied for the determination of dispersion and dilution patterns of atmospheric pollutants. In particular, an easily identifiable tracer (perfluorocarbons) has been released in the atmosphere from the city of Rennes, in France. The concentration of the tracer has then been measured over a period of 72 consecutive hours, at 168 ground-level stations in Western and Eastern Europe. In our experiments, we consider the 168 sampling stations as nodes of the graph and the concentration measured in each of them as signals on the graph. The measurements obtained at different time instances within the 72-hour period form 30 observations, which are used to infer the diffusion topology that can explain well the diffusion of the tracer. For this experiment, we choose $S = 1$ as the observations consist of many zeros entries, which indicates that they can be approximated with a single diffusion process at small scale. Moreover, we fix the scale parameter to $\tau = 3$ and we initialize the Laplacian matrix, with a random graph Laplacian matrix.

In Fig. 6, we illustrate the most important edges of the graph learned with LearnHeat and some representative measurements of the concentration of the tracers, which are used as training signals in the learning. The estimated graph indicates the main directions towards which the tracer moved, which are consistent with the signal observations. These directions are influenced by many parameters such as the meteorological conditions and the direction of the wind. We observe that there exist some strong connections between stations in France and Germany, and those in Sweden and Hungary, which are consistent with the conclusions in the documentation of the dataset [34].

Finally, in Fig. 7, we study how well a diffusion dictionary based on the graph Laplacian can represent the signal observations with only a few atoms of the dictionary. We compute

[4]The dataset is publicly available in https://rem.jrc.ec.europa.eu/etex/ and has already been processed in [35].



(a) (b) (c)

Fig. 6. (a)-(c) The learned graph and different measurements over time of the concentration of the tracer (signal observations). The color code represents the concentration measured in each station. The color indicates the strength of the edge. The lighter the color, the stronger is the edge.

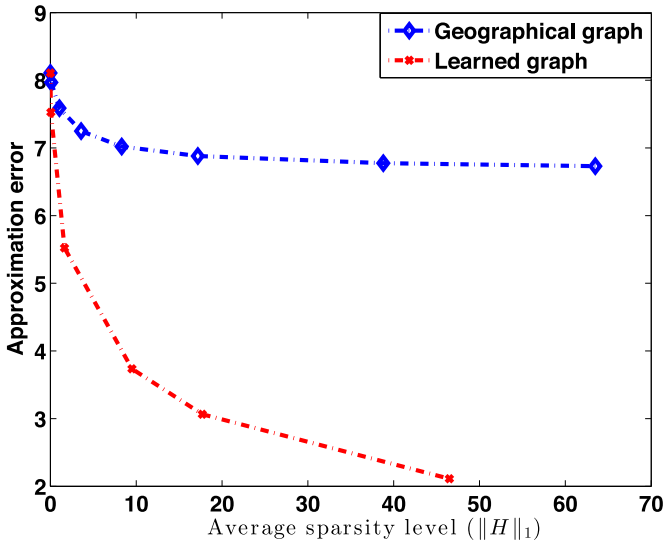


Fig. 7. Approximation performance of the daily signals for different sparsity levels on dictionaries generated from a geographical graph (blue) and the learned graph (red).

the sparse approximation using an iterative soft thresholding algorithm that promotes sparsity of the coefficients H. We compare our results with a diffusion dictionary designed based on a graph that is constructed using geographical distances between these stations. We observe that the approximation error $\|X - e^{-\tau L} H\|_F^2$ is significantly smaller in the case of the diffusion dictionary based on the learned graph for different sparsity levels. These results indicate that learning the topology can bring significant benefits for effective structured data representation.

2) Uber Dataset: In the final set of experiments, we use our graph learning algorithm to detect patterns from Uber rides in New York City. In particular, we use the Uber dataset[5] for the month of September 2014, which provides time and location for pickups. For the sake of simplicity, we divide the city into $N = 29$ taxi zones, as shown in Fig. 8(a), and each zone is a node of a graph. The hourly number of Uber pickups in each zone is a signal on the graph. Moreover, we divide the day into five time slots 1) 7 am - 10 am, 2) 10 am - 4 pm, 3) 4 pm - 7 pm, 4) 7 pm - 12 pm, 5) 12 pm - 7 am. For each of these slots, we define as training signals the number of pickups measured for each hour inside the corresponding time interval, all weekdays of the month.

For each of these five set of training signals, we learn a heat diffusion dictionary with $S = 2$ blocks, for different parameters of α and β. In order to choose the best parameter of α and β, we define as a criteria that the number of edges of the learned graph should be approximately $4N$. We expect that the graph learned for each time interval conveys some information about the traffic patterns in the city. In Fig. 8, we show the learned graphs for each of the time intervals. We can clearly see patterns that are indicative of the behavior of the people in the city. First, there is a clear tendency of people using Uber to go/come mostly to/from airports (JFK, La Guardia, Newark) in early morning [see Fig. 8(b)]. Moreover, the connections of the graph during the rush hours (7 am - 10 am and 4 pm - 7 pm) indicate the commuting of people from/to different neighborhood of the city to/from Manhattan. During the day (10 am - 4 pm), there is no clear pattern as the graph learned from the distribution of the Uber cars indicates that people tend to use Uber to go to random places in the city. Finally, from 7 pm to midnight [see Fig. 8(f)], most of the connections are concentrated across Manhattan, which probably indicates that most of the people use Uber to visit bars or restaurants that are concentrated around that area. These are of course just some observations that confirm the efficiency of our algorithm in learning meaningful graphs. A more detailed mining of the mobility patterns in New York City requires taking into consideration other factors such as the population of each region, a finer grid of the zone, the organization of the city in terms of public transport, which is out of the scope of this paper.

For the sake of completeness, we compare visually the graphs that we learn with the state-of-the-art methods [14], [13], [4], using the same Uber data, for the time interval between 7 pm to midnight. Since there is no groundtruth graph to compare with, for each method, we illustrate the graph that corresponds to parameters that give almost the same number of edges. The results are shown in Fig. 9. Although there is no clear way of validating these results, we can observe that the algorithm proposed in [13] seems to give similar patterns with Fig. 8(f),

[5]The dataset is publicly available in https://github.com/fivethirtyeight/uber-tlc-foil-response

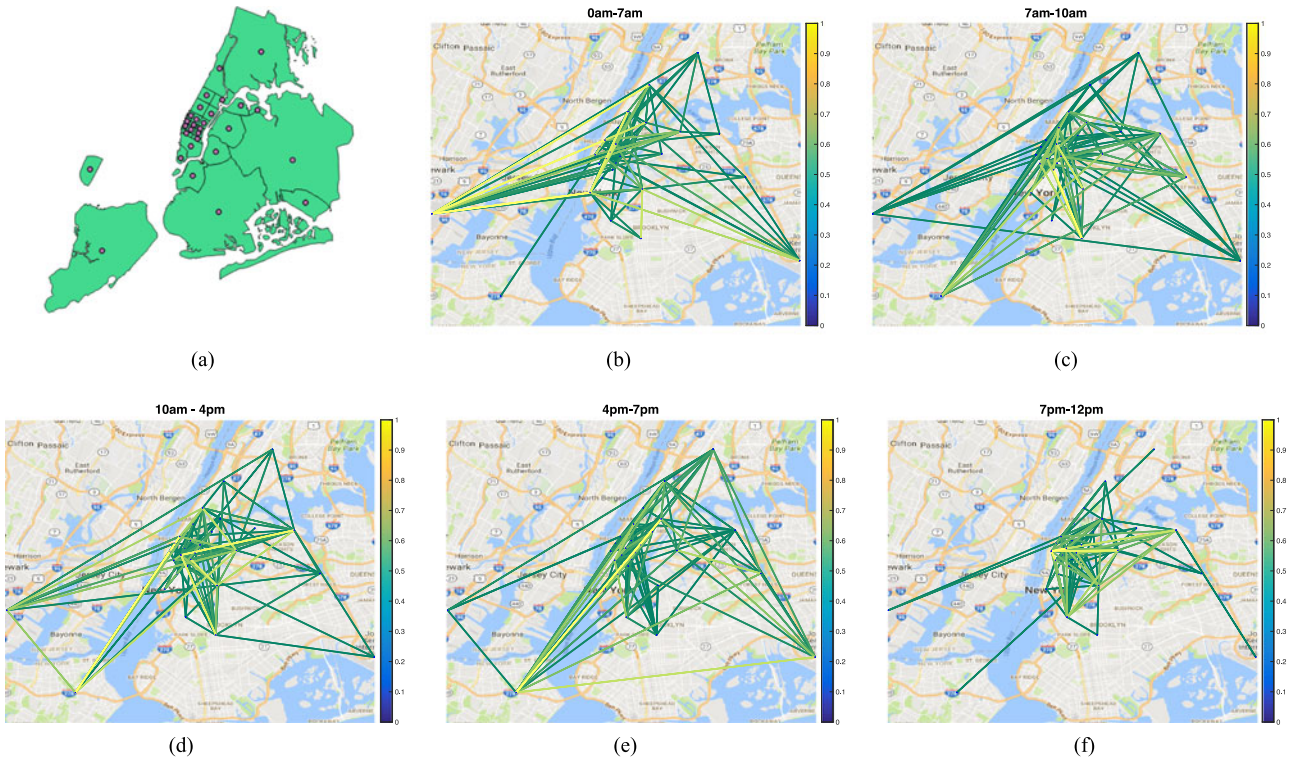


Fig. 8. (a) Boundaries of the taxi zones in New York City. Each zone is represented by a node on a the learned graph. The learned graphs over different time intervals: (b) 0.00 - 7.00 am, (c) 7.00 am - 10.00 am, (d) 10.00 am - 4.00 pm, (e) 4.00 pm - 7.00 pm, and (f) 7.00 pm - 12 pm.

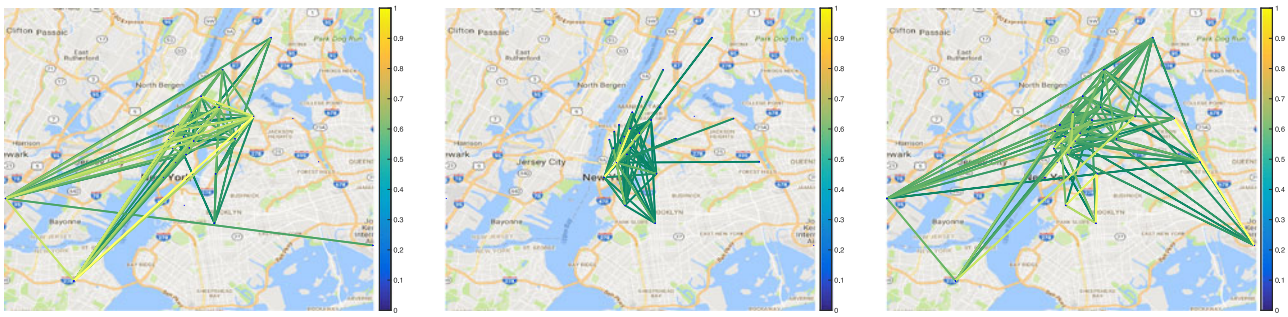


Fig. 9. The learned graphs over the time interval of 7.00 pm - 12 pm with different state-of-the-art methods: (a) [14], (b) [13], and (c) [4]. The color indicates the strength of the edge.

which is a big concentration of the edges inside the Manhatten area. On the other hand, in Fig. 9(c), we observe that the graph learned with a globally smooth model [4] contains connections that are spread across all the areas of the city.

To summarize, we should note that when the underlying training signals do not necessarily follow a heat diffusion model, as it might be the case in real-world data, there might be a gap between signal representation and learning graphs. Since our graph learning framework is posed as an optimization problem which promotes sparse signal representation, it is possible that a good graph for representation is not the actual one that reflects the diffusion but the one that minimizes the approximation error. The advantage of our algorithm though is that we manage to learn a meaningful graph topology, by only utilizing signals that are aggregated, without having access to the time-stamped data, as it is the case in our real-world experiments.

VI. Conclusion

In this paper, we have presented a framework for learning graph topologies (graph Laplacians) from signal observations under the assumption that the signals are generated from heat diffusion processes starting from a few nodes of the graph. Specifically, we have proposed an algorithm for learning graphs that enforces sparsity of the graph signals in a dictionary that is a concatenation of graph diffusion kernels at different scales. Experimental results on both synthetic and real world diffusion processes have confirmed the usefulness of the proposed

algorithm in recovering a meaningful graph topology and thus leading to better data understanding and inference. We believe that the proposed graph learning framework opens new perspectives in the field of data inference and information propagation in particular from the emerging graph signal processing viewpoint.

APPENDIX A
COMPUTATION OF THE GRADIENT

As noted in Section IV-B, Algorithm 1 requires the computation of the gradient of the fitting term with respect to each of the variables H, L, τ. In the following, we discuss the computation of each of the gradients separately.

A. Gradient with Respect to H

The gradient of $Z(L, H, \tau) = \|X - \mathcal{D}H\|_F^2$ with respect to a column h_j of H is independent of the other columns of H. Moreover, it depends only on the corresponding observation x_j and it can be written as

$$\nabla Z_{h_j}(L^t, H^t, \tau^t) = -2\mathcal{D}^T(x_j - \mathcal{D}h_j). \tag{14}$$

B. Gradient with Respect to L

The gradient of $\|X - \mathcal{D}H\|_F^2$ with respect to L is:

$$\begin{aligned}
&\nabla_L \|X - \mathcal{D}H\|_F^2 \\
&= \nabla_L \left(\mathrm{tr}\left(\left(X - \sum_{s=1}^{S} e^{-\tau_s L} H_s \right)^T \left(X - \sum_{s=1}^{S} e^{-\tau_s L} H_s \right) \right) \right) \\
&= \nabla_L \big(\mathrm{tr}(X^T X) - 2 \sum_{s=1}^{S} \mathrm{tr}(H_s X^T e^{-\tau_s L}) \\
&\quad + \sum_{s=1}^{S} \sum_{s'=1}^{S} \mathrm{tr}(H_{s'} H_s^T e^{-(\tau_s + \tau_{s'})L}) \big) \\
&= -2 \sum_{s=1}^{S} \nabla_L \mathrm{tr}(H_s X^T e^{-\tau_s L}) \\
&\quad + \sum_{s=1}^{S} \sum_{s'=1}^{S} \nabla_L \mathrm{tr}(H_{s'} H_s^T e^{-(\tau_s + \tau_{s'})L}).
\end{aligned} \tag{15}$$

In order to compute (15), we make use of the following proposition.

Proposition 1: Consider a general matrix $A \in \mathbb{R}^{N \times N}$ and a symmetric matrix $L \in \mathbb{R}^{N \times N}$, admitting a spectral decomposition $L = \chi \Lambda \chi^T$. Then

$$\nabla_L \mathrm{tr}(A e^L) = \chi \big((\chi^T A^T \chi) \circ B \big) \chi^T,$$

where $\circ$ denotes the Hadamard product and B is the $N \times N$ matrix defined by the entries

$$[B]_{ij} = \begin{cases} e^{\Lambda_{ii}} & \text{if } \Lambda_{ii} = \Lambda_{jj} \\ \frac{e^{\Lambda_{ii}} - e^{\Lambda_{jj}}}{\Lambda_{ii} - \Lambda_{jj}} & \text{otherwise.} \end{cases} \tag{16}$$

Proof: The desired gradient is uniquely defined by satisfying the relation

$$\mathrm{tr}(A e^{(L+\Delta)}) - \mathrm{tr}(A e^L) = \langle \nabla_L \mathrm{tr}(A e^L), \Delta \rangle + \mathcal{O}(\|\Delta\|^2) \tag{17}$$

for all sufficiently small perturbations Δ. Using the fact that the eigenvectors of L are orthonormal, i.e., $\chi^T \chi = I$, where I is the identity matrix, we can write the left hand-side of (17) as follows:

$$\begin{aligned}
&\mathrm{tr}(A e^{(L+\Delta)}) - \mathrm{tr}(A e^L) \\
&= \mathrm{tr}(\chi^T A \chi \chi^T e^{(L+\Delta)} \chi) - \mathrm{tr}(\chi^T A \chi \chi^T e^L \chi) \\
&= \mathrm{tr}(\chi^T A \chi e^{(\Lambda + \chi^T \Delta \chi)}) - \mathrm{tr}(\chi^T A \chi e^{\Lambda}).
\end{aligned} \tag{18}$$

The Frèchet derivative of the matrix exponential at a diagonal matrix Λ applied to a direction Δ is the $N \times N$ matrix $De^{\Lambda}(\Delta) = B \circ \Delta$ with B defined in (16); see [26]. Using the above developments and the linearity of the trace operator we obtain that

$$\begin{aligned}
\langle \nabla_\Lambda \mathrm{tr}(\chi^T A \chi e^{\Lambda}), \Delta \rangle &= \mathrm{tr}(\chi^T A \chi D e^{\Lambda}(\Delta)) \\
&= \mathrm{tr}(\chi^T A \chi (B \circ \Delta)) = \langle \chi^T A^T \chi \circ B, \Delta \rangle.
\end{aligned} \tag{19}$$

Finally, using again the orthonormality of the eigenvectors χ, we can write

$$\begin{aligned}
\langle \nabla_L \mathrm{tr}(A e^L), \Delta \rangle &= \langle \chi^T \nabla_L \mathrm{tr}(A e^L) \chi, \chi^T \Delta \chi \rangle \\
&\overset{(18)}{=} \langle \nabla_\Lambda \mathrm{tr}(\chi^T A \chi e^{-\Lambda}), \chi^T \Delta \chi \rangle \\
&= \langle \chi \nabla_\Lambda \mathrm{tr}(\chi^T A \chi e^{-\Lambda}) \chi^T, \Delta \rangle.
\end{aligned} \tag{20}$$

Combining (19), (20), we conclude that $\nabla_L \mathrm{tr}(A e^L) = \chi(\chi^T A^T \chi \circ B)\chi^T$. ■

Given the result of Proposition 1, the gradient $\nabla_L \mathrm{tr}(A e^{\nu L})$ for some $\nu \in \mathbb{R}$ can be found by applying the chain rule: $\nabla_L \mathrm{tr}(A e^{\nu L}) = \nu \nabla_{\nu L} \mathrm{tr}(A e^{\nu L})$.

C. Gradient With Respect to τ

The gradient of $\|X - \mathcal{D}H\|_F^2$ with respect to τ satisfies

$$\begin{aligned}
&\nabla_\tau \|X - \mathcal{D}H\|_F^2 \\
&= \nabla_\tau \left(\mathrm{tr}\left(\left(X - \sum_{s=1}^{S} e^{-\tau_s L} H_s \right)^T \left(X - \sum_{s=1}^{S} e^{-\tau_s L} H_s \right) \right) \right. \\
&\quad \left. + \sum_{s=1}^{S} \sum_{s'=1}^{S} \mathrm{tr}(H_{s'} H_s^T e^{-(\tau_s + \tau_{s'})L}) \right) \\
&= -2 \sum_{s=1}^{S} \nabla_\tau \mathrm{tr}(H_s X^T e^{-\tau_s L}) \\
&\quad + \sum_{s=1}^{S} \sum_{s'=1}^{S} \nabla_\tau \mathrm{tr}(H_{s'} H_s^T e^{-(\tau_s + \tau_{s'})L}).
\end{aligned} \tag{21}$$

By the Taylor expansion of the exponential, it follows for any $A \in \mathbb{R}^{N \times N}$ that

$$\nabla_{\tau_s} \mathrm{tr}(A e^{-\tau_s L}) = -\mathrm{tr}(AL e^{-\tau_s L}). \tag{22}$$

Combining (21), (22), we obtain that

$$\nabla_{\tau_s}\|X-\mathcal{D}H\|_F^2 = 2\mathrm{tr}(H_s X^T L e^{-\tau_s L}) - 2\sum_{s'=1}^{S}\mathrm{tr}(H_{s'}H_s^T L e^{-(\tau_s+\tau_{s'})L}).$$

Finally, the gradient with respect to the vector τ is given by a vector whose elements consist of the gradient with respect to each element of τ, i.e., $\nabla_\tau\|X-\mathcal{D}H\|_F^2 = \left\{\nabla_{\tau_s}\|X-\mathcal{D}H\|_F^2\right\}_{s=1}^S$.

APPENDIX B
COMPUTATION OF THE LIPSCHITZ CONSTANTS

A condition for ensuring convergence of PALM is that at each iteration of the algorithm the descent lemma is satisfied [7]. This, however, requires to determine a global Lipschitz constant or an approximation thereof such that the descent condition is satisfied. Next, we discuss the computation of the Lipschitz constants related to the update of each of the three variables L, H, τ in our graph learning algorithm. As we will see, it is feasible to compute these constants for the update of H and τ. On the other hand, the computation of the Lipschitz constant is more difficult for L because of the involved matrix exponential. In this case, we perform backtracking to approximate the Lipschitz constant.

A. Variable H

The function $\nabla_H Z(H,L,\tau)$ is globally Lipschitz with Lipschitz constant $C_1(L,\tau) = \|2\mathcal{D}^T\mathcal{D}\|_F$, as can be seen from

$$\begin{aligned}&\|\nabla_H Z(L,H_1,\tau) - \nabla_H Z(L,H_2,\tau)\|_F\\ &= \|-2\mathcal{D}^T(X-\mathcal{D}H_1) + 2\mathcal{D}^T(X-\mathcal{D}H_2)\|_F\\ &= \|2\mathcal{D}^T\mathcal{D}H_1 - 2\mathcal{D}^T\mathcal{D}H_2\|_F\\ &\le \|2\mathcal{D}^T\mathcal{D}\|_F\|H_1-H_2\|_F,\end{aligned}$$

B. Variable L

Due to the difficulty of computing the Lipschitz constant for an exponential matrix function, we estimate the associated constant $C_2(H,\tau)$ by performing backtracking line search as follows. One condition for convergence of PALM is that the descent lemma is satisfied at each iteration, i.e.,

$$\begin{aligned}&Z(L^{t+1},H^{t+1},\tau^t) \le Z(L^t,H^{t+1},\tau^t)\\ &+\langle L^{t+1}-L^t, \nabla_L Z(L^t,H^{t+1},\tau^t)\rangle + \frac{C_2(H,\tau)}{2}\|L^{t+1}-L^t\|_F^2.\end{aligned} \tag{23}$$

Moreover, the solution L^{t+1} of the optimization problem (11) indicates that for every $L\in\mathcal{C}$, the objective function evaluated at L will always be greater or equal to the one evaluated at L^{t+1}, i.e.,

$$\begin{aligned}&\langle L^{t+1}-L^t, \nabla_L Z(L^t,H^{t+1},\tau^t)\rangle + \frac{d_t}{2}\|L^{t+1}-L^t\|_F^2 + \beta\|L^{t+1}\|_F^2\\ &\le \langle L-L^t, \nabla_L Z(L^t,H^{t+1},\tau^t)\rangle + \frac{d_t}{2}\|L-L^t\|_F^2 + \beta\|L\|_F^2.\end{aligned}$$

By setting $L=L^t$ in the right-hand side of the inequality, we obtain that

$$\begin{aligned}&\langle L^{t+1}-L^t, \nabla_L Z(L^t,H^{t+1},\tau^t)\rangle + \frac{d_t}{2}\|L^{t+1}-L^t\|_F^2\\ &+\beta\|L^{t+1}\|_F^2 \le \beta\|L^t\|_F^2\end{aligned}$$

Combining with (23) and using the fact that $d_t \ge C_2(H,\tau)$, we obtain that

$$Z(L^{t+1},H^{t+1},\tau^t)+\beta\|L^{t+1}\|_F^2 \le Z(L^t,H^{t+1},\tau^t)+\beta\|L^t\|_F^2. \tag{24}$$

This result guarantees the decrease of the objective function after each update of the Laplacian matrix over the iterations. The backtracking is shown in Algorithm 2.

Algorithm 2: Backtracking Algorithm for Estimating $C_2(H,\tau)$ at Iteration $t+1$.

1: **Input**: $\eta = 1.1$, initial guess for $C_2(H,\tau)$, $k=1$
2: **Output:** Estimate of the Lipschitz constant $C_2(H,\tau)$
3: **while** (23) is False **do:**
4: Update: $C_2(H,\tau) = \eta^k C_2(H,\tau)$, $d_t = \gamma_2 C_2(H,\tau)$
5: $k = k+1$
6: Update L^{t+1} by solving (11)

C. Variable τ

Since the objective function is convex and twice differentiable with respect to τ, we estimate the Lipschitz $C_3(L,H)$ by computing the Hessian $\nabla_\tau^2\|X-\mathcal{D}H\|_F^2$. Using (21), the entries of this $S\times S$ matrix are given by

$$\begin{aligned}\nabla_\tau^2 Z_{ss} =& -2\,\mathrm{tr}(H_s X^T L^2 e^{-\tau_s L}) + 4\,\mathrm{tr}(H_s H_s^T L^2 e^{-\tau_s L})\\ &+2\sum_{s'\neq s=1}^{S}\mathrm{tr}(H_{s'}H_s^T L^2 e^{-(\tau_s+\tau_{s'})L}),\end{aligned} \tag{25}$$

$$\nabla_\tau^2 Z_{ss'} = 2\,\mathrm{tr}(H_{s'}H_s^T L^2 e^{-(\tau_s+\tau_{s'})L}),\ \text{if } s\neq s'.$$

Given that the Hessian is a positive semidefinite matrix, its 2-norm is its largest eigenvalue and any upper bound on this eigenvalue gives a global Lipschitz constant. We will use the fact that the largest absolute row sum of the matrix represents such an upper bound. For this purpose, we first estimate

$$\begin{aligned}|\nabla_\tau^2 Z_{ss}| \le& \left(2\|H_s\|_F\|X^T\|_F + 4\|H_s\|_F\|H_s^T\|_F\right)\|L^2\|_2^2\\ &+2\sum_{s'\neq s=1}^{S}\|H_{s'}\|_F\|H_s^T\|_F\|L^2\|_2^2,\\ |\nabla_\tau^2 Z_{ss'}| \le& \|H_{s'}\|_F\|H_s^T\|_F\|L^2\|_2^2,\end{aligned}$$

where we have used the fact that $\|L^2 e^{-\tau_s L}\|_2 \leq \|L^2\|_2$, for every $\tau_s \geq 0$, due to the positive semidefiniteness of L. An upper bound on the largest eigenvalue, which in turn gives the Lipschitz constant, is thus given by

$$C_3(L,H) = \max_{s'} \|L\|_2^2 \left(2\|H_{s'}\|_F \|X\|_F + 4 \sum_{s=1}^{S} \|H_{s'}\|_F \|H_s\|_F \right).$$

ACKNOWLEDGMENT

The authors would like to thank S. Segarra, A. Ribeiro, and B. Pasdeloup for sharing their MATLAB code of the algorithms in [14] and [13] used in the experiments. Additionally, we would like to thank G. Stathopoulos for discussions on the solution of the optimization problem, and the anonymous reviewers for their constructive comments on earlier versions of this paper.

REFERENCES

[1] F. R. K. Chung, *Spectral Graph Theory*. Providence, RI, USA: Amer. Math. Soc., 1997.
[2] D. I Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.
[3] B. Lake and J. Tenenbaum, "Discovering structure by learning sparse graph," in *Proc. 33rd Annu. Cogn. Sci. Conf.*, 2010, pp. 778–783.
[4] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, Dec. 2016.
[5] F. Chung, "The heat kernel as the pagerank of a graph," *Natl. Acad. Sci.*, vol. 104, no. 50, pp. 19 735–19 740, 2007.
[6] H. Ma, H. Yang, M. R. Lyu, and I. King, "Mining social networks using heat diffusion processes for marketing candidates selection," in *Proc. 17th ACM Conf. Inf. Knowl. Manage.*, 2008, pp. 233–242.
[7] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Math. Program.*, vol. 146, nos. 1/2, pp. 459–494, Aug. 2014.
[8] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, Jul. 2008.
[9] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. 19th Int. Conf. Artif. Intell. Statist.*, 2016, pp. 920–929.
[10] E. Pavez and A. Ortega, "Generalized Laplacian precision matrix estimation for graph signal processing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 6350–6354.
[11] M. Belkin and P. Niyogi, "Towards a theoretical foundation for Laplacian-based manifold methods," in *Proc. 15th Annu. Conf. Comput. Learn. Theory*, 2005, pp. 486–500.
[12] J. Mei and J. M. F. Moura, "Signal processing on graphs: Estimating the structure of a graph," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Brisbane, Australia, Apr. 2015, pp. 5495–5499.
[13] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat, "Characterization and inference of weighted graph topologies from observations of diffused signals," 2016, arXiv:1605.02569.
[14] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," 2016, arXiv:1608.03008v1.
[15] D. J. Bartholomew, M. Knott, and I. Moustaki, *Latent Variable Models and Factor Analysis: A Unified Approach*, 3rd ed. Hoboken, NJ, USA: Wiley, Jul. 2011.
[16] X. Zhang, X. Dong, and P. Frossard, "Learning of structured graph dictionaries," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Kyoto, Japan, 2012, pp. 3373–3376.
[17] D. Thanou, D. I Shuman, and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3849–3862, Aug. 2014.
[18] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proc. IEEE*, vol. 98, no. 6, pp. 1045–1057, Apr. 2010.
[19] R. Gribonval, "Should penalized least squares regression be interpreted as maximum a posteriori estimation?" *IEEE Trans. Signal Process.*, vol. 59, no. 5, pp. 2405–2410, May. 2011.
[20] A. Smola and R. Kondor, "Kernels and regularization on graphs," in *Proc. 16th Annu. Conf. Comput. Learn. Theory*, 2003, pp. 144–158.
[21] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Stat. Soc. Ser. B*, vol. 67, no. Pt2, pp. 301–320, 2005.
[22] H. Bauschke and P. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Berlin, Germany: Springer, 2011.
[23] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014.
[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
[25] B. O. Brendan, E. Chu, N. P. Neal, and S. Boyd, "Operator splitting for conic optimization via homogeneous self-dual embedding," *J. Optim. Theory Appl.*, vol. 169, no. 3, pp. 1042–1068, 2016.
[26] N. J. Higham, *Functions of Matrices: Theory and Computation*. Philadelphia, PA, USA: SIAM, 2008.
[27] P. Kandolf and S. D. Relton, "A block Krylov method to compute the action of the Frechet derivative of a matrix function on a vector with applications to condition number estimation," *SIAM J. Sci. Comput.*, May 2017.
[28] G. H. Golub and C. F. Van Loan, *Matrix Computations* (Johns Hopkins Studies in the Mathematical Sciences), 4th ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 2013.
[29] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval.* Cambridge, U.K.: Cambridge Univ. Press, 2008.
[30] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, Oct. 1999.
[31] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hungarian Acad. Sci.*, vol. 5, pp. 17–61, 1960.
[32] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2011.
[33] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Mar. 2009.
[34] K. Nodop, R. Connolly, and F. Girardi, "The field campaigns of the European tracer experiment (ETEX): Overview and results," *Atmos. Environ.*, vol. 32, no. 24, pp. 4095–4108, 1998.
[35] R. Pena, X. Bresson, and P. Vandergheynst, "Source localization on graphs via l1 recovery and spectral graph theory," in *Proc. IEEE 12th Image, Video, Multidimensional Signal Process. Workshop*, 2016, pp. 1–5.

Authors' photographs and biographies not available at the time of publication.