Recovery of Time-Varying Graph Signals via Distributed Algorithms on Regularized Problems

Junzheng Jiang *Member, IEEE*, David B. Tay, Qiyu Sun and Shan Ouyang

***Abstract*—The recovery of missing samples from available noisy measurements is a fundamental problem in signal processing. In the graph setting, the aim is to recover missing samples on a subset of vertices using samples from other vertices. This process is also sometimes known as graph signal inpainting, reconstruction, forecasting or inference. Many of the existing algorithms do not scale well with the size of the graph and/or they cannot be implemented efficiently in a distributed manner. In this paper, we develop efficient distributed algorithms for the recovery of time-varying graph signals. The *a priori* assumptions are that the signal is smooth with respect to the graph topology and correlative across time. These assumptions can be incorporated in an optimization formulation of the algorithm via Tikhonov regularization terms. Although this approach is widely used in many graph signal processing problems, our formulation is tailored to yield algorithms that can be efficiently implemented in a distributed manner. Two different distributed algorithms, arising from two different formulations, are proposed to solve the optimization problems. The first involves the ℓ_2-norm, and a distributed least squared recovery algorithm (DLSRA) is proposed that leverages the graph topology and sparsity of the corresponding Hessian matrix. The proposed DLSRA is different from other Newton-like methods as there is no need to update the approximate inverse of the Hessian during the iterations. The second involves the ℓ_1-norm and the philosophy of the alternating direction method of multipliers (ADMM) is utilized to develop the algorithm. To achieve a totally distributed algorithm, the proposed inexact Newton method is incorporated into the conventional ADMM to give a distributed ADMM recovery algorithm (DAMRA). The two distributed algorithms require only data exchanges between vertices in localized neighbourhood subgraphs. This is in contrast to centralized methods where global data exchanges are required. Experiments on a variety of synthetic and real-world datasets demonstrate that the proposed algorithms are superior to the existing methods in terms of the computational complexity and convergence rate.**

Keywords: Graph signal recovery, Distributed algorithm, Least squares method, Alternative direction method of multipliers (ADMM).

I. Introduction

Graph signal processing is an emerging discipline which finds applications in a diverse range of fields, e.g. sensor networks, image processing, power grids and big data [1]-[7]. Graph signal processing (GSP) deals with signals residing on irregular discrete domains, that are modelled by graphs, and generalizes techniques from classical signal processing for regular domains. In recent years, several traditional tools in the regular domain have been extended to the graph domain, including the graph Fourier transform [3], graph filter [8], [9], graph wavelet filter bank [10]-[13], graph signal recovery [14], etc. Nevertheless, there is still a variety of problems that remain unsolved.

A graph signal is defined by two entities: (i) the graph topology which can be specified mathematically via a graph matrix, such as the adjacency or Laplacian; and (ii) the signal values that are indexed by the graph vertices, which can be represented as vectors. For example, the temperature measurement at the nodes of a sensor network can be modeled as a graph signal. Two fundamental, but complementary problems in GSP is to (i) estimate the topology, and/or (ii) estimate the signal, from partial or incomplete information from one or both entities. Recent works on topology estimation can be found in [15]-[19]. In this paper we are concerned with the second problem of signal recovery, which is also known as graph signal inpainting, reconstruction, forecasting or inference. Such a problem arises in many real-world scenarios, for instance in wireless sensor networks. In sensor networks, the signals are inevitably corrupted by some random disturbances and sensor malfunction occurs due to the complexity associated with the sheer number of sensors deployed and environmental factors. Therefore graph signal recovery is an important practical problem.

There has been much work devoted to signal recovery in the regular domain, such as for images [20], [21] and videos [22], [23]. These classical techniques cannot however deal with problems on irregular domains directly. By exploiting the graph topology via the graph matrix (adjacency or Laplacian), extensions to the graph domain are found in [14], [24]-[27] for time-static signals and in [16], [28]-[31] for time-varying signals. A common theme in these works is the notion of signal smoothness w.r.t. graph topology and signal correlations along the temporal axis which are then exploited in formulating the recovery algorithm. More detailed comparisons of these works are found in subsection I-B.

It is desirable for any graph signal processing algorithm to scale well with the number of nodes in the graph, so that computational efficiency will still be achieved with large graphs. Generally speaking, graph signal processing can be implemented in a centralized manner or a distributed manner. The former requires the entire data to be available in a facility for data processing, whereas the latter requires only data within a localized subgraph, before processing in distributed facilities.

Jiang and Ouyang are with the School of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China; David B. Tay is with the School of Information Technology, Deakin University, Waurn Ponds, Victoria 3216, Australia; and Sun is with the Department of Mathematics, University of Central Florida, Orlando, Florida 32816. Emails: jzjiang@guet.edu.cn; david.tay@deakin.edu.au; qiyu.sun@ucf.edu; hmoysh@guet.edu.cn. This work is partially supported by the National Natural Science Foundations of China (Grant No. 61761011, 61871425), and the National Science Foundation (DMS1816313).

In the language of optimization, the centralized manner yields the (globally) optimal or exact solution to the corresponding optimization problem. This approach is therefore preferable, in principle, if the cost of the communication between the central processing node and other nodes are low. However, for graphs of large size, the centralized manner is often impractical as the communication resource required is prohibitive. Furthermore, there is a lack of robustness with this approach, e.g. a failure in the central node due to a malicious attack. Due to these considerations, the distributed manner is therefore preferable in practice, where for the corrupted nodes, the recovered values are obtained via local interactions with their neighbors. There is no need to share information globally and this has the added benefit of increased privacy protection. There are also some scenarios where only distributed processing is possible. For example, due to energy constraints, sensors in some wireless networks have limited communication range and can only communicate with its local neighbors. In some literature the term 'distributed' is taken to mean immediate neighbors that are one hop (one edge) away from a node in consideration. We shall adopt a more generalized interpretation of 'distributed' where the neighbors can be several hops away, but the maximum number of hops is small relative to the diameter of the graph. There is however a lack of distributed algorithms for time-varying graph signal recovery. The recovery algorithms in [16], [29], [31] require centralized processing and can lead to good recovery performance for graphs of small and moderate orders. The distributed recovery algorithm in [28] relies on the gradient-descent, and suffers from slow convergence. These considerations motivate us to develop efficient distributed algorithms for the recovery of time-varying graph signals.

A. Notation

We use the common convention of representing matrices and vectors with bold letters and scalars with normal letters. For a matrix $\mathbf{C}$, denote its transpose and pseudo-inverse by $\mathbf{C}^T$ and $\mathbf{C}^\dagger$ respectively. Let $\mathbf{0}(\mathbf{1})$ be the vector of appropriate size with all entries taking the value $0(1)$. For a set F, denote its cardinality by $\mu(F)$. For the recovery problem on graph $\mathcal{G}=(V,E)$, denote $\mathcal{M}\subseteq V$ ($\mathcal{U}\subseteq V$) as the set consisting of uncorrupted (corrupted) vertices.

B. Problem statement and related work

Graph signal recovery aims to restore the missing or corrupted samples on a subset of vertices using the uncorrupted samples on other vertices, also known as the graph signal-inpainting, reconstruction, forecasting or inference. In general, the model of the corrupted graph signal is given by

$$\mathbf{b}=\begin{bmatrix}\mathbf{b}_\mathcal{M}\\ \mathbf{b}_\mathcal{U}\end{bmatrix}=\mathbf{x}+\boldsymbol{\varepsilon}=\begin{bmatrix}\mathbf{x}_\mathcal{M}\\ \mathbf{x}_\mathcal{U}\end{bmatrix}+\begin{bmatrix}\boldsymbol{\varepsilon}_\mathcal{M}\\ \boldsymbol{\varepsilon}_\mathcal{U}\end{bmatrix}, \tag{I.1}$$

where $\mathbf{b}$ is the measurement, $\mathbf{x}$ is the original signal, $\mathbf{x}_\mathcal{M}\in\mathbb{C}^{|\mathcal{M}|}$ is the uncorrupted part of $\mathbf{x}$ and $\mathbf{x}_\mathcal{U}\in\mathbb{C}^{|\mathcal{U}|}$ is the corrupted or missing part. Signal values residing on $k\in\mathcal{M}$ are contaminated by low level noise, i.e., $b_k=x_k+\varepsilon_k$ with $|\varepsilon_k|\le\delta, k\in\mathcal{M}$ is the uncorrupted signal with low-level noise added. Without loss of generality, the uncorrupted nodes are indexed first and the corrupted nodes are indexed next. When the signal at the each vertex is time-varying, we have vector valued time series and (I.1) can be written as $\mathbf{b}_t=\mathbf{x}_t+\boldsymbol{\varepsilon}_t$ for $t=0,\ldots,T-1$, where T denotes the number of time instants. The ensemble to vectors can be concatenated to give $\mathbf{X}=[\mathbf{x}_0|\mathbf{x}_1|\ldots|\mathbf{x}_{T-1}]$.

There are several approaches that have been proposed for recovering the graph signals in both the time-static and time-varying cases. Some *a priori* assumption about the underlying signal is first made. This is then exploited in formulating the recovery algorithm. Using the notion of low-pass bandlimited (w.r.t. graph spectrum) signals, recovery algorithms were proposed in [26], [27]. The techniques in [26], [27] however require the bandwidth of the signal and the eigenbasis (of the graph Laplacian) to be known, and this may not be feasible in practice, especially with large graphs. Another common assumption is that the graph signal is smooth with respect to the underlying graph topology, i.e. signal values of two neighboring vertices do not vary significantly, and this is observed in many practical scenarios. Low-pass bandlimited signals are smooth but smooth signals need not be strictly bandlimited. The notions of smooth and low-pass are related but it is easier to incorporate the former in any recovery algorithm. A commonly used framework for recovery (and other mathematically similar problems) is the use of the Tikhonov regularization in an optimization formulation, e.g. ridge regression and LASSO. For static graph signals, the recovery can be formulated as [14], [24], [25]

$$\min_{\mathbf{x}}\quad \|\mathbf{B}_\mathcal{M}\mathbf{x}-\tilde{\mathbf{b}}_\mathcal{M}\|_2^2+\beta_0 R_\mathcal{G}(\mathbf{x}) \tag{I.2}$$

where $R_\mathcal{G}(\mathbf{x})$ is the regularization that measure the non-smoothness, β_0 is a trade-off factor between the fidelity (first term) and non-smoothness, $\tilde{\mathbf{b}}_\mathcal{M}=\mathbf{B}_\mathcal{M}\mathbf{b}$, and

$$\mathbf{B}_\mathcal{M}=\begin{bmatrix}\mathbf{I}_{|\mathcal{M}|} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}.$$

A common class of regularizer is given by [4], [14], [32].

$$R_\mathcal{G}(\mathbf{x})=S_p(\mathbf{x})\equiv\left\|(\mathbf{I}-\mathbf{W}_n)\mathbf{x}\right\|_p^p \tag{I.3}$$

where $\mathbf{W}_n=\mathbf{W}/\lambda_{\max}(\mathbf{W})$, and $\mathbf{W}$ is the adjacency matrix (to be defined explicitly later) with $\lambda_{max}(\mathbf{W})$ denoting the largest magnitude eigenvalue. Common p values are $p=2$ (ℓ_2-norm), and $p=1$ (ℓ_1-norm) in which $S_1(\mathbf{x})$ is also known as the total variation. When $p=2$, the minimization in (I.2) becomes a least squares problem and it admits a closed form solution,

$$\begin{aligned}\mathbf{x}^\star&=\left(\mathbf{B}_\mathcal{M}^T\mathbf{B}_\mathcal{M}+\beta_0(\mathbf{I}-\mathbf{W}_n)^T(\mathbf{I}-\mathbf{W}_n)\right)^{-1}\mathbf{B}_\mathcal{M}^T\tilde{\mathbf{b}}_\mathcal{M}\\ &=\left(\mathbf{B}_\mathcal{M}+\beta_0(\mathbf{I}-\mathbf{W}_n)^T(\mathbf{I}-\mathbf{W}_n)\right)^{-1}\tilde{\mathbf{b}}_\mathcal{M}.\end{aligned} \tag{I.4}$$

Recovering $\mathbf{x}$ directly via the formula (I.4) requires centralized processing. The recovered signal $\mathbf{x}^\star$ is obtained by solving a linear system of equations. However, with this approach, the computational burden is high when the graph is of large order. To overcome this problem, a distributed approach was presented in [25], where the linear equations is solved by applying the iterative formula

$$\mathbf{x}^{(0)} = \mathbf{0}, \quad \mathbf{x}^{(m+1)} = (\mathbf{I} - \alpha_0 \mathbf{C}_{\mathcal{M}})\mathbf{x}^{(m)} + \alpha_0 \tilde{\mathbf{b}}_{\mathcal{M}} \qquad \text{(I.5)}$$

that can be implemented in a distributed manner in each iteration, where $\mathbf{C}_{\mathcal{M}} = \mathbf{B}_{\mathcal{M}} + \beta_0(\mathbf{I} - \mathbf{W}_n)^T(\mathbf{I} - \mathbf{W}_n)$. Here α_0 is a global parameter which should satisfy $0 < \alpha_0 \leq 2/\lambda_{\max}(\mathbf{C}_{\mathcal{M}})$ to ensure the convergence of (I.5), where $\lambda_{\max}(\mathbf{C}_{\mathcal{M}})$ denotes the largest magnitude eigenvalue of $\mathbf{C}_{\mathcal{M}}$. At each iteration in (I.5), the value of each node is updated via interaction with its neighbors. This distributed algorithm can lead to good recovery results, but it suffers from a low convergence rate due to the gradient-like nature of the algorithm. The algoritm also requires the determination of $\lambda_{\max}$, which can be computationally costly when the graph is of large size. To avoid eigenvalue calculation, we propose a different regularization, which is also more general, that is based on highpass graph filters.

For time-varying signals, a straightforward approach is to apply the methods discussed above on each time instant t independently. However, better results can be achieved by exploiting the correlation of the signal across time, see [16], [28]-[31]. The approaches in [28]-[30] are essentially based on the Tikhonov regularization in (I.2) but the first (fidelity) term will now consider the signal for all time instants, i.e. the ensemble $\mathbf{X} = \{\mathbf{x}_0, \cdots, \mathbf{x}_{T-1}\}$. In [28], the assumption is that the time difference signal $\Delta_t \equiv \mathbf{x}_t - \mathbf{x}_{t-1}$ is smooth and the regularizer $R_{\mathcal{G}}(\mathbf{x})$ used is a Laplacian quadratic form of Δ_t. In [29], a sequence of graphs is used to model the time evolution and the sequence is combined, via Cartesian product, to give an extended graph. The extended graph models both correlations across vertices and time. The regularizer $R_{\mathcal{G}}(\mathbf{x})$, a.k.a. space-time kernel [29], is then on the extended graph signal $\mathbf{X}$ and eigendecomposition is required to determine the kernel. The regularizer jointly penalizes the non-smoothness in time/vertex but for online implementation, there is limited flexibility in specifying the temporal frequency response. In [30], the regularizer $R_{\mathcal{G}}(\mathbf{x})$ consists of two terms to separately model the non-smoothness over vertex and time. Applications considered in [30] were in the denoising of dynamic meshes and the inpainting of time-lapse video. The denoising problem leads to a joint vertex-temporal filter where the FFC (Fast-Fourier-Chebyshev) filter was proposed. The FFC can be implemented distributively w.r.t. the graph topology but cannot have online (real-time) processing. The FFC however cannot be applied to the inpainting problem as the associated operator cannot be expressed as a function of the shift matrix. The efficient distributed implementation for inpainting was not addressed in [30]. More recently, concepts from the Vector Autoregressive (VAR) model [33], [34] were used to model time-varying graph signals [15], [16], [31]. The graph topology is embedded in the modelling by imposing some structure in the matrix coefficient of the VAR. The resulting graph VAR (GVAR) models have matrix coefficients with sparsity structures that are related to the adjacency/Laplacian. In [16], the correlation at the current time instant is also considered but only one previous time step is used in the VAR model. The recovery algorithms in [31] and [16] do not explicitly factor in any smoothness assumption over the vertices at the current time instant. It should be mentioned that the works in [15], [16], [31] is concerned with graph signal modeling which has more general applications, e.g. graph topology identification.

C. Main contributions

Most of the methods reviewed above do not scale well with the size of the graph and/or they cannot be implemented efficiently in a distributed manner. Most methods also do not consider the ℓ_1-norm which can promote sparsity. The exception is in [30] but the work does not consider efficient distributed implementation via online (real-time) processing. The motivation of this work is therefore to develop novel recovery algorithms that will address the shortcomings. The contributions of this work can be summarised as follows:

1) A tailored formulation of the recovery problem is formulated, using the Tikhonov regularization framework, that allows for efficient distributed implementation of the algorithms. Separate penalty terms, that accounts for graph topology smoothness and temporal correlations respectively, are used. This gives flexibility in specifying the trade-offs with the fidelity terms. Two types of norms, namely the ℓ_1- and ℓ_2-norms, are considered for the penalty terms, and the former promotes some form of sparsity.
2) A new distributed algorithm is developed for solving the recovery problem with the ℓ_2-norm. The algorithm is an inexact Newton-like algorithm that does not require update of the inverse Hessian matrix, which is computationally expensive. The key idea behind the approach is the decomposition of a large graph (representing the entire network) into a sequence of small size subgraphs that are overlapping. The optimization is performed locally on each subgraph and the fusion of the local solutions asymptotically approximates the global solution. With this decomposition, the proposed algorithm solves for the recovered signal by only using information from a localized subgraph of vertices. It is shown that the algorithm has a high convergence rate and matrix eigendecomposition is not required. The algorithm also has the light/heavy property (to be detailed later) where the computational load of some nodes are light. The convergence of the distributed algorithm is also shown under conditions that are satisfied in practice.
3) The solution of the mixed ℓ_1/ℓ_2-norm problem requires a decomposition of the objective function. The alternating direction methods of multiplier (ADMM) is ideally suited for its solution. However a direct application of the ADMM will not give a distributed algorithm (from a graph signal processing perspective). An approximate form of ADMM which has distributed implementation is developed.

D. Organization

In Section II, we briefly review some preliminaries on graphs and introduce the concept of overlapping graph decomposition (see (II.1)). We also briefly review relevant graph signal processing concepts and present a new definition of smoothness of graph signals in the vertex domain (see Definition II.2). In Section III, we first introduce the notion of temporal correlation in graph signals (see Definition III.1). We then formulate the recovery problem for time-varying graph signals (see (III.5)). In Section IV, we consider the ℓ_2-norm formulation and develop an iterative algorithm to solve the recovery problem which can be implemented in a distributed manner (see Algorithm IV.1). We also show that the algorithm converges at an exponential rate (see Theorem IV.2). In Section V, we consider the formulation with mixed ℓ_1- and ℓ_2-norms. A distributed algorithm that is based on the ADMM is developed here (see Algorithm V.1). In Section VI several numerical examples are presented that will show the performance of the two proposed distributed algorithms. Comparisons with other methods are also found here. Concluding remarks are found in Section VII. All proofs are found in the Appendix.

II. Preliminaries

A. Fundamentals of Graphs

Let $\mathcal{G} := (V, E)$ be a graph, where $V = \{1, 2, \cdots, N\}$ is the set of vertices and E is the set of edges [35], [36]. Denote the weighted adjacency matrix by $\mathbf{W}$, whose (i,j)th entry $[W]_{i,j}$ is the weight of the edge between vertices i and j that represents the correlation between vertices $i, j \in V$. The combinatorial Laplacian matrix is defined by $\mathbf{L} := \mathbf{D} - \mathbf{W}$. The random-walk normalized Laplacian is defined as $\mathbf{L}^{\text{rw}} := \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$. Throughout this paper, we only consider undirected graphs $\mathcal{G}$ without self-loops and multiple edges. This means that the unnormalized adjacency and Laplacian matrices are symmetric and the main diagonal of the adjacency are all zeros. The random-walk normalized Laplacian is however not symmetric. The adjacency and Laplacian matrices are examples of the graph shift matrix $\mathbf{S}$, which represents a fundamental operator of graph signals just like the unit delay z^{-1} in traditional signal processing.

For graph $\mathcal{G} = (V, E)$, the *geodesic distance* $\rho(i,j)$ is defined as the number of edges in the shortest path between vertices i and j, i.e. the smallest number of hops from i to j and vice-versa [35]. The geodesic distance on a graph can be used as a measure of the communication cost between two given agents. When two agents are not neighbors (i.e., not connected via a direct link), they can still communicate with each other via a chain of intermediate agents along a path of shortest distance. When the geodesic distance increases, the number of agents involved, and hence the cost, also increases. The diameter of a graph $\mathcal{D}$ is defined as the greatest geodesic-distance between any two pair of vertices, i.e. $\mathcal{D} \equiv \max\limits_{i,j\in V} \rho(i,j)$.

The *r-neighborhood vertices* of an uncorrupted node $k \in \mathcal{M}$ is defined as $B(k,r) := \{j \in V : \rho(k,j) \le r\}$. Associated with these vertices are edges whose endpoints are in the set $B(k,r)$, to be denoted by $E(k,r)$. The vertex and edge subsets together form the *r-neighborhood subgraph* which is defined as $\mathcal{G}_{k,r} := \big(B(k,r), E(k,r)\big)$. The distributed algorithms to be presented later are based on constructing a suitable subgraph for every uncorrupted vertex in the set $\mathcal{M}$. The radius $r \ge 1$ is chosen large enough such that for every vertex $i \in V$, there are at least two vertices $k, l \in \mathcal{M}$ such that $i \in B(k,r) \cap B(l,r)$, i.e. every vertex in the graph $\mathcal{G}$ belongs to at least two subgraphs. We then decompose the graph $\mathcal{G}$ into a family of subgraphs $\mathcal{G}_{k,r}, k \in \mathcal{M}$, i.e.

$$\mathcal{G} = \cup_{k\in\mathcal{M}} \mathcal{G}_{k,r}, \tag{II.1}$$

which are overlapping, i.e. $\mathcal{G}_{k,r} \cap \mathcal{G}_{l,r} \ne \emptyset$ for some $k, l \in \mathcal{M}$. We remark that the overlapping decomposition is a key idea in the distributed algorithms - see Algorithm IV.1.

We now introduce the notion of the *geodesic-width* of a linear graph operator $\mathbf{A}$, which is related to the concept of geodesic distance.

Definition II.1. The *geodesic-width* $\sigma := \sigma(\mathbf{A})$ of a graph operator $\mathbf{A} = [a(i,j)]_{i,j\in V}$ is the smallest nonnegative integer σ such that $a(i,j) = 0$ for all $i, j \in V$ with $\rho(i,j) > \sigma$.

For an operator $\mathbf{A}$ with geodesic-width σ, each element of the processed signal $\mathbf{y} = \mathbf{A}\mathbf{x}$ can be evaluated as $y(i) = \sum_{\rho(i,j)\le\sigma} A(i,j)x(j)$. The summation over the index set $\rho(i,j) \le \sigma$ indicates that, for a given node i, only the input signal values $x(j)$ in the σ-hop neighbourhood subgraph $B(i,\sigma)$ will contribute to the output $y(i)$.

The graphs considered in this work have the polynomial growth property, which is defined as: there exist positive constants $D_1(\mathcal{G})$ and d such that

$$\mu(B(i,r)) \le D_1(\mathcal{G})(r+1)^d \tag{II.2}$$

for all $i \in V$ and $r \ge 0$. The minimum value of the constants d and $D_1(\mathcal{G})$ in (II.2) are termed as the *Beurling dimension* and *density* of the graph $\mathcal{G}$ respectively [35]. In other words, the number of vertices in the r-neighborhood is bounded by some polynomial in the radius r.

The overlapped decomposition (II.1) plays a crucial role in the proposed recovery algorithm. For the implementation of the proposed distributed algorithms, the radius r should be substantially smaller than the diameter $\mathcal{D}$ of the graph to economize the computational and communication costs, while on the other hand, the radius r cannot be chosen too small as the distributed algorithm has faster convergence for larger radius r, see (IV.14) and Theorem IV.2.

B. Graph Signal

Signals residing on a graph $\mathcal{G}$ can be represented by the vector $\mathbf{x} = [x_1\ x_2\ \cdots\ x_N]^T$, where the element x_i is indexed by the vertex $i \in V$. In many real world applications, the data $\mathbf{x}$ belongs to some finite sequence space, i.e. $\mathbf{x} \in \ell^p$ $(1 \le p \le \infty)$ [13], [35]. In other words, the graph signals in many applications are bounded and/or with finite energy.

For signal recovery, *a prior* assumption is that the signals are smooth with respect to the graph topology. The conventional smoothness definition in (I.3) requires the calculation of $\lambda_{\max}(\mathbf{W})$ which can be computationally expensive when the graph size N is large. In this paper, we introduce a new measure of non-smoothness.

Definition II.2. A signal $\mathbf{x}$ on graph $\mathcal{G} = (V, E)$ is smooth if

$$S_{\mathcal{G}} = \|\mathbf{H}_1 \mathbf{x}\|_p^p, \quad 1 \leq p < \infty \tag{II.3}$$

is small, where $\mathbf{H}_1$ is a high-pass graph filter having zero response to the constant signal, i.e., $\mathbf{H}_1 \mathbf{1} = \mathbf{0}$.

Now the measure of non-smoothness (II.3) is a generalization of the total variation in (I.3), where $\mathbf{H}_1 = \mathbf{I} - \mathbf{W}_n$. Definition II.2 is a generic definition of smoothness that encompasses previous definitions. From the graph spectral perspective, a smooth signal has components mainly in the low frequencies. Therefore the high frequency components are close to zero and it is reasonable to specify the level of non-smoothness via the highpass filtered graph signal $\mathbf{H}_1\mathbf{x}$.

We next address the requirements for the graph filter $\mathbf{H}_1$ in Definition II.2. The zero response property $\mathbf{H}_1\mathbf{1} = \mathbf{0}$ implies a highpass characteristic for the filter. The graph filter $\mathbf{H}_1$ should be bounded [13] and have small geodesic-width σ to allow for distributed algorithms. Graph filters are usually constructed from matrix polynomials of the graph shift matrix $\mathbf{S}$,

$$\mathbf{H}_1 = P_1(\mathbf{S}) = p_0 \mathbf{I} + \sum_{l=1}^{n} p_l \mathbf{S}^l, \tag{II.4}$$

where p_l, $l = 0, \cdots, n$ denote the polynomial coefficients. Polynomial filters can be implemented without eigendecomposition. If $\mathbf{S} = \mathbf{L}^{\text{rw}}$ the condition $\mathbf{H}_1\mathbf{1} = \mathbf{0}$ requires $P_1(0) = 0$, since $\mathbf{L}^{\text{rw}}\mathbf{1} = \mathbf{0}$ ($\mathbf{L}^{\text{rw}}$ has eigenvalue 0 and eigenvector $\mathbf{1}$). This implies $p_0 = 0$. A simple example of a polynomial filter is the spline filter:

$$\mathbf{H}_1 = \left(\frac{1}{2}\mathbf{L}^{\text{rw}}\right)^n, \tag{II.5}$$

where $n \geq 1$ is the prescribed order. Since $\mathbf{L}^{\text{rw}}\mathbf{1} = \mathbf{0}$, it can be readily verified that $\mathbf{H}_1\mathbf{1} = \mathbf{0}$, i.e. the spline filter also has zero response to the constant signal. In this paper, $\mathbf{H}_1$ is the spline filter for the non-smoothness measure (II.3). However other types of filters could potentially be used for $\mathbf{H}_1$ as long as they have the zero response property.

III. Time-Varying Graph Signals Recovery

In this section we present the problem formulation of signal recovery via the Tikhonov regularization framework. This framework is a powerful and versatile technique in many signal and data analysis problems. The key idea is to introduce penalty term(s) to reflect some assumptions of the signal of interest, e.g. smooth signal. This has been successfully applied for static graph signals via the penalty $R_{\mathcal{G}}(\mathbf{x})$ in (I.2) for smoothness over graph topology. We first consider exploiting the temporal smoothness of the time-varying signals on graph by leveraging the VAR model. Then, the recovery problem of the time-varying signals is formulated as different optimization problems with either the ℓ_2 norm or the mixed ℓ_1/ℓ_2-norms.

A. Exploiting temporal correlation

In many real-world applications, sensors continuously acquire data which can be modelled as time-varying graph signals [28]. The ensemble of the time sequence of graph signals is denote as $\mathbf{X} = \{\mathbf{x}_0, \cdots, \mathbf{x}_{T-1}\}$, where T denotes the number of time instants and $\mathbf{x}_t$ denotes a static graph signal at time instance t. We refer to $\mathbf{x}_t$ as the t^{th} snapshot of $\mathbf{X}$. A plausible assumption, in practice, is that the acquired data vary smoothly over time. For instance, the consecutive global sea pressure data recorded by a pressure sensor network are smooth over time, i.e. readings typically do not change significantly over time. Furthermore, the graph signal values within a localized neighborhood of vertices are usually correlated with each other. Specifically, the signal value $x_t(k)$ of vertex k at time t is not only correlated with $x_{t-l}(k)$, but also correlated with its neighbors $x_{t-l}(i)$, $i \in B(k, r)$. These correlations will be exploited in the development of the recovery of time-varying signals on graphs. To formalize this notion, we first invoke the vector autoregressive (VAR) model [33] to give the predicted signal:

$$\hat{\mathbf{x}}_t = \sum_{l=1}^{t} \mathbf{C}_{t,l} \mathbf{x}_{t-l} \tag{III.1}$$

where $\mathbf{C}_{t,l}, t \geq 1$ are the matrix coefficients accounting for the correlations. The requirements of $\mathbf{C}_{t,l}$ are that: (i) as an operator, it should behave like a low-pass graph filter to capture the smooth time variation; (ii) the geodesic-width should be small to capture localised correlation. These requirements can be achieved if $\mathbf{C}_{t,l}$ is modelled as a polynomial of the Laplacian matrix $\mathbf{C}_l = \hat{c}_l(\mathbf{L}^{\text{rw}}) \equiv \sum_{m=0}^{K} c_{l,m} (\mathbf{L}^{\text{rw}})^m$ with $c_{l,0} \neq 0$. This is then similar to the graph polynomial VAR model in [31]. Now other parametrizations for $\mathbf{C}_{t,l}$, such as the edge-variant filter [38], are possible but with polynomial functions, the determinations of the matrix coefficients is relatively straightforward as will be described below. In [16], the model considered is given by $\mathbf{x}_t = \mathbf{A}_0\mathbf{x}_t + \mathbf{A}_1\mathbf{x}_{t-1}$ which has only one time-lag, whereas (III.1) is more general as the number of time-lags is arbitrary. The first term with $\mathbf{A}_0\mathbf{x}_t$ models the correlation between vertices of the current time signal. Note that the current time signal $\mathbf{x}_t$ is not included in (III.1) as the current time correlation across vertices is considered separately via the non-smoothness term in (II.3). Now the works in [16] and [31] are primarily focussed on graph signal modelling via VAR, whereas our work uses VAR as means to incorporate the notion of temporal correlation. Based on the model (III.1), we can define the temporal correlation as follows.

Definition III.1. A time-varying signal $\mathbf{X} = \{\mathbf{x}_0, \cdots, \mathbf{x}_{T-1}\}$ is temporally correlated if there exist matrices $\mathbf{C}_{t,l}$ such that

$$S_T = \sum_{t=1}^{T-1} \left\| \mathbf{x}_t - \sum_{l=1}^{t} \mathbf{C}_{t,l} \mathbf{x}_{t-l} \right\|_q^q, \tag{III.2}$$

is small for $1 \leq q < \infty$.

Now S_T can be used as a penalty term to reflect the temporal correlation assumption just like $S_{\mathcal{G}}$ in (II.3) is used as a penalty measure to reflect the smoothness across the vertices. If $\mathbf{C}_{t,l} = \mathbf{C}_l$ for $t = 1, \cdots, T-1$, the temporal evolution dynamic is time-invariant and this will be assumed here.

In many practical applications, we only have one time realization of the data and have no prior knowledge of the temporal evolution dynamics. In such scenarios, a learning approach can be used to estimate the matrix coefficients from the dataset. For instance, we may learn the evolution dynamic by solving the optimization problem with a p-norm objective function,

$$\underset{\mathbf{C}_l, l=1,\cdots,t}{\text{minimize}} \sum_{t=1}^{T-1} \left\| \mathbf{x}_t - \sum_{l=1}^{t} \mathbf{C}_l \mathbf{x}_{t-l} \right\|_p^p, \tag{III.3}$$

where $\mathbf{x}_{t-l}$, $l = 1, \cdots, t$ are the training data. Using the polynomial model for the $\mathbf{C}_l$, the optimization problem (III.3) reduces to determining the optimal coefficients $c_{l,m}$, $l = 1, \cdots, T-1$, $m = 0, \cdots, K$. The problem is convex if $p \geq 1$ and can be efficiently solved using interior point algorithms, for which freeware such as CVX [39] are readily available. A simple example of (III.1) is given by

$$\hat{\mathbf{x}}_t = (\mathbf{I} - \tau \mathbf{L}_{\mathcal{G}}^{\text{rw}}) \mathbf{x}_{t-1}, \tag{III.4}$$

where $\tau \in [0, 1]$. The above system can be considered as discretization of the dynamical system $d\mathbf{x}/dt = \mathbf{L}_{\mathcal{G}}^{\text{rw}} \mathbf{x}$ with time step size τ.

B. Recovery problem formulation

For a time-varying graph signal $\mathbf{X} = \{\mathbf{x}_0, \cdots, \mathbf{x}_{T-1}\}$, recovery can be achieved through two different approaches. The first is to simultaneously recover the signals for all time instants, which is also known as batch processing. However with this approach, the computational cost is high and there will be a long processing delay for large T. The second approach is to sequentially recover $\mathbf{x}_t$, $t = 0, \cdots, T-1$, one time instant t at a time, which is also known as online processing. We adopt the second approach in this paper, but the proposed algorithms can also be used in the first approach. Additional matrix algebraic techniques, such as matrix vectorization and Kronecker product, are however needed with the first approach.

Based on the notion of smoothness discussed above, the recovery problem is formulated as an unconstrained optimization problem. The objective function consists of a data fidelity term and non-smoothness penalty terms. Using Definition II.2 and Definition III.1, the problem is given by

$$\min_{\mathbf{x}} \frac{1}{2} \left\| \mathbf{B}_{\mathcal{M}} \mathbf{x} - \tilde{\mathbf{b}}_{\mathcal{M}} \right\|_2^2 + \alpha \left\| \mathbf{H}_1 \mathbf{x} \right\|_p^p + \beta \left\| \mathbf{x} - \mathbf{x}_d \right\|_q^q, \tag{III.5}$$

where the parameters α, β are weighting factors, $\mathbf{H}_1$ is a highpass graph filter in Definition II.2, and the predicted signal is given by

$$\mathbf{x}_d = \sum_{l=1}^{t} \mathbf{C}_l \tilde{\mathbf{x}}_{t-l}.$$

Since recovery is performed sequentially, the prediction is based on $\tilde{\mathbf{x}}_{t-l}$ ($l = 1, \ldots, t$), which are the recovered signal from previous time instants, and are not the true underlying signals $\mathbf{x}_{t-l}$ ($l = 1, \ldots, t$). For $\mathbf{x}_0$ (when $t = 0$), we set $\beta = 0$ since there is no previously recovered data.

Different p and q values in (III.5) require different algorithms for its solution. We consider two cases in this work. The first case is when $p = q = 2$. We then have a least squares problem and a closed-form solution can be readily obtained. The second case is when $p = 1$ and $q = 2$ and the objective function is non-differentiable. Distributed algorithms to solve both problems will be formulated in the next two sections. The computational cost of these algorithms scales linearly with the graph size N.

IV. Graph signal recovery with the ℓ_2-norm

With $p = q = 2$, the minimization (III.5) becomes a least-squares problem,

$$\min_{\mathbf{x}} \quad \frac{1}{2} \left\| \mathbf{B}_{\mathcal{M}} \mathbf{x} - \tilde{\mathbf{b}}_{\mathcal{M}} \right\|_2^2 + \alpha \left\| \mathbf{H}_1 \mathbf{x} \right\|_2^2 + \beta \left\| \mathbf{x} - \mathbf{x}_d \right\|_2^2. \tag{IV.1}$$

A closed form solution can be readily derived and is given by

$$\tilde{\mathbf{x}} = \mathbf{D}_{\mathcal{M}}^{-1} \tilde{\mathbf{b}}_{\mathcal{M},d} \tag{IV.2}$$

where

$$\mathbf{D}_{\mathcal{M}} \equiv \mathbf{B}_{\mathcal{M}} + 2\alpha \mathbf{H}_1^T \mathbf{H}_1 + 2\beta \mathbf{I} \tag{IV.3}$$

and

$$\tilde{\mathbf{b}}_{\mathcal{M},d} \equiv \tilde{\mathbf{b}}_{\mathcal{M}} + 2\beta \mathbf{x}_d. \tag{IV.4}$$

A direct computation of (IV.2) requires the inversion of the matrix $\mathbf{D}_{\mathcal{M}}$. The computational burden for this inversion is high when the graph size N is large. There can also be numerical issue associated with inverting a large size matrix. This direct approach theoretically requires all measurement data $\tilde{\mathbf{b}}_{\mathcal{M},d}$ to be available before the solution can be obtained, i.e. a centralized approach. To avoid these issues a distributed algorithm is preferable [27], [40], [41]. Most distributed algorithms for graph signals are based on a polynomial approximation to a function of the graph shift matrix [41]. However $\mathbf{D}_{\mathcal{M}}^{-1}$ is not a function of the graph shift matrix. Therefore, the polynomial approximation approach is not applicable for this recovery problem.

Instead of polynomial approximation, we propose a different distributed algorithm to solve Problem (IV.1). The strategy behind the algorithm is to 'divide-and-conquer'. The key idea is to introduce a family of localized subproblems over subgraphs $\mathcal{G}_{k,2r}$ in the decomposition (II.1),

$$\min_{\mathbf{x}} \frac{1}{2} \left\| \mathbf{B}_{\mathcal{M}} \mathbf{M}_k^{2r} \mathbf{x} - \tilde{\mathbf{b}}_{\mathcal{M}} \right\|_2^2 + \alpha \left\| \mathbf{H}_1 \mathbf{M}_k^{2r} \mathbf{x} \right\|_2^2 + \beta \left\| \mathbf{M}_k^{2r} \mathbf{x} - \mathbf{x}_d \right\|_2^2 \tag{IV.5}$$

for $k \in \mathcal{M}$. Note that the radius is $2r$ here but is r in (II.1). The indicator operators $\mathbf{M}_k^r$ ($k \in \mathcal{M}$) are $|V| \times |V|$ diagonal matrices whose (i,i)th entries are unity if $i \in B(k,r)$ and zero otherwise. The indicator operator $\mathbf{M}_k^{2r}$ has the effect of zeroing out values in $\mathbf{x}$ that are outside the subgraphs $\mathcal{G}_{k,2r}$, i.e. $\mathbf{M}_k^r$ can be viewed as a localization operator.

Now the gradient, w.r.t. $\mathbf{x}$, of the objective function in (IV.5) is given by:

$$\begin{aligned} \nabla \quad = \quad & \left(\mathbf{B}_{\mathcal{M}} \mathbf{M}_k^{2r}\right)^T \left(\mathbf{B}_{\mathcal{M}} \mathbf{M}_k^{2r}\right) \mathbf{x} - \left(\mathbf{B}_{\mathcal{M}} \mathbf{M}_k^{2r}\right)^T \tilde{\mathbf{b}}_{\mathcal{M}} \\ & + 2\alpha \left(\mathbf{H}_1 \mathbf{M}_k^{2r}\right)^T \left(\mathbf{H}_1 \mathbf{M}_k^{2r}\right) \mathbf{x} + 2\beta \left(\mathbf{M}_k^{2r}\right)^T \left(\mathbf{M}_k^{2r}\right) \mathbf{x} \\ & - 2\beta \left(\mathbf{M}_k^{2r} \mathbf{x}\right)^T \mathbf{x}_d, \end{aligned}$$

where $\mathbf{D}_{\mathcal{M}}$ and $\tilde{\mathbf{b}}_{\mathcal{M},d}$ are defined in (IV.3) and (IV.4). Setting $\nabla = \mathbf{0}$ gives

$$\mathbf{M}_k^{2r} \mathbf{D}_{\mathcal{M}} \mathbf{M}_k^{2r} \mathbf{x} = \mathbf{M}_k^{2r} \tilde{\mathbf{b}}_{\mathcal{M},d}. \tag{IV.6}$$

As $\left(\mathbf{M}_k^{2r}\mathbf{D}_{\mathcal{M}}\mathbf{M}_k^{2r}\right)$ is rank deficient due to $\mathbf{M}_k^{2r}$, we seek the least error norm solution via the pseudo-inverse:

$$\mathbf{v}_{k,r} = \left(\mathbf{M}_k^{2r}\mathbf{D}_{\mathcal{M}}\mathbf{M}_k^{2r}\right)^{\dagger}\mathbf{M}_k^{2r}\tilde{\mathbf{b}}_{\mathcal{M},d},\ k \in \mathcal{M}, \tag{IV.7}$$

which represents the localized solution of the minimization problem (IV.1) in $B(k, 2r)$. A crucial observation is that $\mathbf{v}_{k,r}$ provides a local approximation to the global solution $\tilde{\mathbf{x}}$ of the minimization problem (IV.1) in $B(k, r)$ [13], [35], i.e.

$$\mathbf{v}_{k,r}(i) \approx \tilde{\mathbf{x}}(i),\ i \in B(k, r),$$

when r is large enough. Note that in the subproblems, the radius is $2r$ but with the local approximation, the radius is r, i.e. to obtain an approximate local solution, we need to solve a larger problem. Technically, other values that are greater than r could also be used. The choice of $2r$ is found by experimentation and represent a compromise between accuracy and complexity (as measured by the problem size).

Now the radius parameter r in the graph decomposition (II.1) is chosen so that, for each $i \in V$, there are at least two uncorrupted vertices $j, k \in \mathcal{M}$ such that $i \in B(j, r) \cap B(k, r)$. Therefore, by solving the series of localized subproblems (IV.5) and taking the overlap effect into account, we will have at least two solution values for each vertex $i \in V$. Now only the vertices in $\mathcal{M}$ that are at most r geodesic distance away from i, i.e. vertices in $B(i, r) \cap \mathcal{M}$, will contribute to the solution at i. Combining these solutions via a local patch gives the following aggregated value

$$v_r(i) = \frac{1}{|B(i, r) \cap \mathcal{M}|} \sum_{k \in B(i,r) \cap \mathcal{M}} v_{k,r}(i). \tag{IV.8}$$

Using (IV.7) and (IV.8), the patched solution for all vertices is given by:

$$\mathbf{v}_r = \Big(\sum_{k' \in \mathcal{M}} \mathbf{M}_{k'}^r\Big)^{-1} \sum_{k \in \mathcal{M}} \mathbf{M}_k^r \mathbf{v}_{k,r} = \mathbf{J}\tilde{\mathbf{b}}_{\mathcal{M},d} \tag{IV.9}$$

where

$$\mathbf{J} \equiv \Big(\sum_{k' \in \mathcal{M}} \mathbf{M}_{k'}^r\Big)^{-1} \sum_{k \in \mathcal{M}} \mathbf{M}_k^r \left(\mathbf{M}_k^{2r}\mathbf{D}_{\mathcal{M}}\mathbf{M}_k^{2r}\right)^{\dagger}\mathbf{M}_k^{2r}. \tag{IV.10}$$

Note that the role $\mathbf{M}_k^r$ in (IV.9) is to localize the solution $\mathbf{v}_{k,r}$ to $B(k, r)$. Now the patched solution $\mathbf{v}_{k,r}$ can be considered a first approximation to the exact solution $\tilde{\mathbf{x}}$ in (IV.2) and $\mathbf{J}$ considered as an approximation to $\mathbf{D}_{\mathcal{M}}^{-1}$. The next Lemma gives an error bound on this approximation.

Lemma IV.1. For graph $\mathcal{G}$, the patched solution satisfies

$$\|\mathbf{v}_r - \tilde{\mathbf{x}}\|_2 \le \delta_r \|\tilde{\mathbf{x}}\|_2, \tag{IV.11}$$

where the error matrix norm is defined as

$$\delta_r \equiv \|\mathbf{I} - \mathbf{J}\mathbf{D}_{\mathcal{M}}\|_2. \tag{IV.12}$$

Lemma IV.1 can be readily derived by using $\tilde{\mathbf{b}}_{\mathcal{M}} = \mathbf{D}_{\mathcal{M}}\tilde{\mathbf{x}}$. Consider the following quantity:

$$\delta_{r,\sigma} := \frac{(D_1(\mathcal{G}))^2(2\sigma+1)^d\kappa^2}{\kappa - 1} \exp\Big(-\frac{\theta}{2\sigma}r\Big)(3r + 2\sigma + 1)^d, \tag{IV.13}$$

where $\theta = \ln(\kappa/(\kappa - 1))$, $\kappa > 1$ is the condition number of the matrix $\mathbf{D}_{\mathcal{M}} := \mathbf{B}_{\mathcal{M}} + 2\alpha\mathbf{H}_1^T\mathbf{H}_1 + 2\beta\mathbf{I}$, $\sigma \ge 1$ is the geodesic-width of the graph filter $\mathbf{H}_1$, and d and $D_1(\mathcal{G})$ are the Beurling dimension and density of the graph $\mathcal{G}$ respectively. It is shown in the appendix that

$$\delta_r \le \delta_{r,\sigma}, \tag{IV.14}$$

i.e. $\delta_{r,\sigma}$ is an upper bound for the error norm. Now when $r \to \infty$, $\delta_{r,\sigma} \to 0$. Therefore we have

$$\lim_{r \to \infty} \delta_r = 0. \tag{IV.15}$$

Theoretically, this means that the inverse matrix $\mathbf{D}_{\mathcal{M}}^{-1}$ can be approximated by $\mathbf{J}$ arbitrarily close if a sufficiently large radius r is chosen. The actual δ_r value in practice however is much smaller than the upper bound $\delta_{r,\sigma}$ - see discussion after Theorem IV.2.

The exact solution in (IV.2) via matrix inversion is not practical for large graphs and is not distributed. We next develop an simple yet efficient distributed method for solving (IV.1). The method is based on an approximation to the classical Newton's method that exploits second order derivative information to accelerate convergence. With $\phi(\mathbf{x})$ denoting the objective function in (IV.1), the conventional (exact) Newton's iteration is given by

$$\begin{aligned}\mathbf{x}^{(m)} &= \mathbf{x}^{(m-1)} - \left(\nabla^2\phi(\mathbf{x}^{(m-1)})\right)^{-1}\nabla\phi(\mathbf{x}^{(m-1)}) \\ &= \mathbf{x}^{(m-1)} - \mathbf{D}_{\mathcal{M}}^{-1}\left(\mathbf{D}_{\mathcal{M}}\mathbf{x}^{(m-1)} - \tilde{\mathbf{b}}_{\mathcal{M},d}\right).\end{aligned} \tag{IV.16}$$

The iterations in (IV.16) require the knowledge of $\mathbf{D}_{\mathcal{M}}^{-1}$ but we are trying to avoid the calculation of $\mathbf{D}_{\mathcal{M}}^{-1}$ directly. We therefore propose to replace the inverse Hessian $\mathbf{D}_{\mathcal{M}}^{-1}$ with $\mathbf{J}$ in (IV.16). The rationale is that $\mathbf{J}$ is an approximation to $\mathbf{D}_{\mathcal{M}}^{-1}$. This gives an inexact-Newton's method with iterations given by

$$\mathbf{x}^{(m)} = \mathbf{J}\tilde{\mathbf{b}}_{\mathcal{M},d} + \left(\mathbf{I} - \mathbf{J}\mathbf{D}_{\mathcal{M}}\right)\mathbf{x}^{(m-1)} \tag{IV.17}$$

Now (IV.17) can also be equivalently expressed as follows.

$$\begin{cases}\mathbf{v}^{(m)} = \mathbf{J}\tilde{\mathbf{b}}^{(m-1)} \\ \tilde{\mathbf{b}}^{(m)} = \tilde{\mathbf{b}}^{(m-1)} - \mathbf{D}_{\mathcal{M}}\mathbf{v}^{(m)} \\ \mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} + \mathbf{v}^{(m)}\end{cases} \tag{IV.18}$$

for $m \ge 1$, where

$$\mathbf{x}^{(0)} = \mathbf{0},\quad \tilde{\mathbf{b}}^{(0)} = \tilde{\mathbf{b}}_{\mathcal{M},d}. \tag{IV.19}$$

The equivalence between (IV.17) and (IV.18) with the initial condition (IV.19) can be readily shown using the fact that $\mathbf{x}^{(m)} - \mathbf{x}^{(m-1)} = \mathbf{v}^{(m)}$. The iterative equations in (IV.18) forms the basis of our distributed recovery algorithm. The next Theorem establishes the condition for convergence of the algorithm to the exact solution.

Theorem IV.2. If the radius parameter r is chosen such that

$$\delta_r < 1 \tag{IV.20}$$

then $\mathbf{x}^{(m)}$ for $m \ge 0$, using (IV.18) and (IV.19), converges to the true optimal solution $\tilde{\mathbf{x}}$ in (IV.2) at an exponential rate, i.e.

$$\|\mathbf{x}^{(m)} - \tilde{\mathbf{x}}\|_2 \le {\delta_r}^m \|\tilde{\mathbf{x}}\|_2,\ m \ge 0. \tag{IV.21}$$

TABLE I: Error matrix norm δ_r for various graphs considered in the experiment in Section VI.

Graph	N	δ_r
Circulant	256	0.001
Random Geometric	4096	8×10^{-4}
Sea pressure	500	5×10^{-4}
US temperature	218	3×10^{-4}
Sea temperature	100	0.031

Using (IV.14), we can see that the condition (IV.20) is satisfied, and hence the convergence of $\{x^{(m)}, m \geq 0\}$ to the ground truth $\tilde{x}$ is guaranteed, for all r satisfying

$$r \geq \big(r_1 + \frac{4d}{\theta}\ln r_1\big)\sigma. \qquad \text{(IV.22)}$$

where

$$r_1 = 2 + \frac{2}{\theta}\ln\big(18^d \sigma^{2d}\kappa(D_1(\mathcal{G}))^2\big) \geq \frac{2d}{\theta}.$$

Note that the requirement (IV.22) is a sufficient but not a necessary condition for convergence. The theoretical lower bound estimate in (IV.22) is quite loose and numerical simulations performed in Section VI indicate that $r = 2$ or 3 is sufficient to ensure that (IV.20) is satisfied. Table I shows that actual δ_r values for the examples considered in Section VI. Theorem IV.2 tells us that convergence to the exact solution will be achieved as long as $\mathbf{J}$ is a reasonable approximation to $\mathbf{D}_{\mathcal{M}}^{-1}$ to ensure $\delta_r < 1$, i.e. it does not have to be arbitrarily close. In most cases, as seen in Table I, δ_r is quite small, which will then result in fast convergence rates as shown in Section VI.

Now the iterative algorithm can be implemented in a distributed manner, as described in Algorithm IV.1. By virtue of the polynomial growth property of typical graphs, the number of uncorrupted neighbors is small and independent of the graph size N. Therefore the patching operation can also implemented in a distributed manner. Multiplication with $\mathbf{D}_{\mathcal{M}}$ can be also implemented in a distributed manner as its geodesic-width 2σ is typically small. At each step of the iteration in Algorithm IV.1, every vertex $k \in \mathcal{M}$ is required to: (i) store data of size $O((r+\sigma)^{2d})$; (ii) perform $O((r+\sigma)^{2d})$ arithmetic operations for local recovery; (iii) perform $O((r+\sigma)^{2d})$ arithmetic operations for the update; and (iv) transmit data to its $(2r+2\sigma)$-neighbors. In total, the complexity of the algorithm is $O\big(\mu(\mathcal{M})K_I(r+\sigma)^{2d} + NK_I(1/2+\sigma)^{2d}\big)$ where K_I denotes the number of iterations. The complexity grows linearly with the graph size N and this indicates that the algorithm has a good scaling property.

Light/Heavy Computation Property: There are three computational steps in Algorithm IV.1; namely (i) Local Recovery (LR), (ii) Patch (P) and (iii) Update (U). Now only the uncorrupted nodes need to perform all three steps, i.e. heavy computation load. The corrupted nodes only need to perform the P and U steps, i.e. light computation load. This property is also found in the algorithm proposed in [27] where the concept of a communication graph and a processing graph is proposed. The former is for diffusion of information (requiring light computations) and the latter is for data regression via the basis of the Graph Fourier Transform (requiring heavy computations). However the algorithm [27] is gradient descent based that utilizes the first order information only, whereas our proposed algorithm exploits the second order information that leads to fast convergence. Furthermore the work in [27], though considers sampling pattern that could be time varying, does not consider time-varying graph signals. Note that the distributed GTVR method [25] does not have this property.

Algorithm IV.1 Distributed Least Squared Recovery Algorithm (DLSRA)

```
Operation: For each $k \in \mathcal{M}$, calculate $\mathbf{J}_k = \big(\mathbf{M}_k^{2r}\mathbf{D}_{\mathcal{M}}\mathbf{M}_k^{2r}\big)^{\dagger}$.
Initialization: $\tilde{\mathbf{b}}_k^{(0)} = \mathbf{M}_k^{2r}\tilde{\mathbf{b}}_{\mathcal{M},d} = (\tilde{b}_{\mathcal{M},d}(i))_{i \in B(k,2r)}$ and $m = 0$.
Iteration:
  1) Local Recovery (LR) step
       $\mathbf{v}_{k,r} = \mathbf{J}_k \tilde{\mathbf{b}}_k^{(m)}, k \in \mathcal{M}$.
  2) Patch (P) step
       $v_r(i) = \frac{1}{\mu(B(i,r)\cap\mathcal{M})}\sum_{k \in B(i,r)\cap\mathcal{M}} v_{k,r}(i), i \in V$.
  3) Update (U) step
       $x^{(m+1)}(i) = x^{(m)}(i) + v_r(i)$
       $\tilde{b}^{(m+1)}(i) = \tilde{b}^{(m)}(i) - \sum_{j \in B(i,2\sigma)} D_{\mathcal{M}}(i,j)v_r(j), i \in V$.
  4) If $|v_r(i)| \leq \varepsilon, i \in V$, terminate the algorithm; Otherwise, form $\tilde{\mathbf{b}}_k^{(m+1)} = (\tilde{b}^{(m+1)}(i))_{i \in B(k,2r)}$ and set $m = m+1$.
```

We conclude this section by comparing the proposed method with other Newton-like methods.

Remark IV.3. The power of the Newton's method lies primary in the use of second derivative information via the Hessian. Different Newton-like methods primarily differ in how the inverse Hessian is approximated and/or updated. There are many Newton-like methods in the literature but we will focus primarily on distributed methods. In the decentralized quasi-Newton method proposed in [42], the localized Hessian was constructed by decentralizing the conventional Broyden-Fletcher-Goldfarb-Shanno (BFGS) update. It falls under the class of quasi-Newton methods where the secant condition is satisfied. In the approximate Newton method [43], the distributed approximation of the inverse Hessian was achieved by invoking the truncated Taylor series for matrix inverse and the secant condition is not satisfied. There are distinct differences between our proposed DLSRA and the existing ones. Firstly, the proposed construction of the approximate inverse Hessian leverages on the overlapping decomposition of the graph. This approach better exploits the graph topology for signal recovery when compared with (i) the quasi-Newton method where the Hessian approximation is constrained by the

BFGS formula; and (ii) the approximate Newton method that relies on the Taylor series. Secondly, the approximated inverse Hessian of the proposed method, i.e., matrix $\mathbf{J}$ in (IV.10), needs only to be calculated once and is fixed during the iterations. With the existing methods the approximated Hessian or Hessian inverse needs to updated at every iteration. Thirdly, the proposed algorithm has the Light/Heavy Computation Property described above, and this is not shared by the existing methods.

V. Recovery with mixed ℓ_1- and ℓ_2-norms

We now consider using the ℓ_1-norm for the non-smoothness measure $S_{\mathcal{G}}$. This is motivated from images processing where it was demonstrated in [37] that the ℓ_1-norm can sometimes lead to better results than the ℓ_2-norm. Images generally consist of smooth regions with some discontinuities, i.e. edges. Some graph signals may also have underlying discontinuities and we would like to preserve these discontinuity as much as possible during the recovery process. As the ℓ_1-norm is often used as a proxy for the ℓ_0-norm, we can use the ℓ_1-norm as a penalty to promote sparsity in the highpass components of the graph signal $\mathbf{x}$. This is equivalent to promoting smoothness in as many vertices as possible and there will be non-smooth vertices where the highpass component will be relatively large. These large highpass components are for the underlying discontinuities in the signal and this will be demonstrated in subsection VI-F.

With $p = 1$ and $q = 2$, the minimization (III.5) becomes

$$\min_{\mathbf{x}} \quad \frac{1}{2}\|\mathbf{B}_{\mathcal{M}}\mathbf{x} - \tilde{\mathbf{b}}_{\mathcal{M}}\|_2^2 + \alpha\big\|\mathbf{H}_1\mathbf{x}\big\|_1 + \beta\big\|\mathbf{x} - \mathbf{x}_d\big\|_2^2. \quad \text{(V.1)}$$

Such a formulation has been considered previously in the context of denoising signals in the regular domain. This formulation, which is known as the total variation approach for denoising [44], [45]. A formulation that uses a mixed norm was also considered in [30] for the inpainting of time-lapse video. However, unlike (V.1) which considers each time instant $\mathbf{x}_t$ separately and in succession, the formulation in [30] considers the entire ensemble $\mathbf{X}$ (of time signals) simultaneously. The latter precludes the use of online (real-time) processing and requires batch processing where measurements for all time instants must be available before processing. The objective function in (V.1) is similar to that found in LASSO, for which the alternative direction method of multipliers (ADMM) [46] (which can handle ℓ_1-norm terms) can be used for finding the solution. The conventional ADMM however cannot be implemented in a distributed manner. We develop here an approximate version of the ADMM algorithm that has a distributed implementation. By denoting $\mathbf{z} = \mathbf{H}_1\mathbf{x}$, we can reformulate problem (V.1) as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\|\mathbf{B}_{\mathcal{M}}\mathbf{x} - \tilde{\mathbf{b}}_{\mathcal{M}}\|_2^2 + \alpha\|\mathbf{z}\|_1 + \beta\big\|\mathbf{x} - \mathbf{x}_d\big\|_2^2 \\ \text{s.t.} \quad & \mathbf{H}_1\mathbf{x} - \mathbf{z} = \mathbf{0}. \end{aligned} \quad \text{(V.2)}$$

The augmented Lagrangian function of the above problem is

$$\begin{aligned} L_\gamma(\mathbf{x},\mathbf{z},\mathbf{w}) \;=\; & \frac{1}{2}\|\mathbf{B}_{\mathcal{M}}\mathbf{x} - \tilde{\mathbf{b}}_{\mathcal{M}}\|_2^2 + \alpha\|\mathbf{z}\|_1 + \beta\big\|\mathbf{x} - \mathbf{x}_d\big\|_2^2 \\ & + \gamma\mathbf{w}^T(\mathbf{H}_1\mathbf{x} - \mathbf{z}) + \frac{\gamma}{2}\|\mathbf{H}_1\mathbf{x} - \mathbf{z}\|_2^2, \end{aligned} \quad \text{(V.3)}$$

where $\mathbf{w}$ is the (scaled) dual variable and $\gamma > 0$ is the penalty parameter [46]. With the augmented Lagrangian, the (scaled) alternative direction method of multipliers (ADMM) seeks to iteratively find the solutions of (V.2) via the following iterations

$$\begin{cases} \mathbf{x}^{(m+1)} = \underset{\mathbf{x}}{\operatorname{argmin}} \;\; L_\gamma(\mathbf{x},\mathbf{z}^{(m)},\mathbf{w}^{(m)}) \\ \mathbf{z}^{(m+1)} = \underset{\mathbf{z}}{\operatorname{argmin}} \;\; L_\gamma(\mathbf{x}^{(m+1)},\mathbf{z},\mathbf{w}^{(m)}) \\ \mathbf{w}^{(m+1)} = \mathbf{w}^{(m)} + \mathbf{H}_1^T\mathbf{x}^{(m+1)} - \mathbf{z}^{(m+1)}. \end{cases} \quad \text{(V.4)}$$

The first two steps/equations are basically optimizing the augmented Lagrangian L_γ with two out of the three sets of variables fixed. The primal variable $\mathbf{x}^{(m+1)}$ can be obtained by solving a least squares problem, with fixed $\mathbf{z} = \mathbf{z}^{(m)}$ and $\mathbf{w} = \mathbf{w}^{(m)}$, representing the current solution for these variables. To solve for the auxiliary variable $\mathbf{z}^{(m+1)}$, with fixed $\mathbf{x} = \mathbf{x}^{(m+1)}$ and $\mathbf{w} = \mathbf{w}^{(m)}$, the proximal operator for the ℓ_1 norm, which is soft thresholding operator, can be used. The iterative algorithm for the solution of (V.2) is therefore given by the following three equations:

$$\begin{aligned} \mathbf{x}^{(m+1)} &= \mathbf{A}^{-1}\mathbf{c}^{(m)} && \text{(V.5)} \\ \mathbf{z}^{(m+1)} &= S_{\alpha/\gamma}\big(\mathbf{H}_1\mathbf{x}^{(m+1)} + \mathbf{w}^{(m)}\big) && \text{(V.6)} \\ \mathbf{w}^{(m+1)} &= \mathbf{w}^{(m)} + \mathbf{H}_1\mathbf{x}^{(m+1)} - \mathbf{z}^{(m+1)}, && \text{(V.7)} \end{aligned}$$

where

$$\mathbf{A} \equiv \mathbf{B}_{\mathcal{M}} + 2\beta\mathbf{I} + \gamma\mathbf{H}_1^T\mathbf{H}_1,$$

$$\mathbf{c}^{(m)} \equiv \tilde{\mathbf{b}}_{\mathcal{M}} + 2\beta\mathbf{x}_d + \gamma\mathbf{H}_1^T(\mathbf{z}^{(m)} - \mathbf{w}^{(m)}),$$

and $S_{\alpha/\gamma}(\mathbf{t})$ is the elementwise soft thresholding operator

$$[S_{\alpha/\gamma}(\mathbf{t})]_i = S_{\alpha/\gamma}(t_i) \equiv \begin{cases} t_i - \alpha/\gamma, & t_i > \alpha/\gamma \\ 0, & |t_i| \le \alpha/\gamma \\ t_i + \alpha/\gamma, & t_i < -\alpha/\gamma \end{cases}$$

for $\mathbf{t} = [t_1 \dots t_N]^T$ [46]. If we now examine (V.5)-(V.7), we can see that (V.6) and (V.7) can be implemented in a distributed manner because (i) $\mathbf{H}_1$ has small geodesic-width (and multiplication can be achieved distributively); and (ii) $S_{\alpha/\gamma}$ is an elementwise operator. The main obstacle to a completely distributed implementation is therefore (V.5). In conventional ADMM, the primal variable $\mathbf{x}^{(m+1)}$ is typically solved via the use of matrix factorization such as Cholesky, the use of certain matrix inverse Lemma, or some iterative methods such as gradient-based method and limited-memory BFGS method [46]. The matrix factorization and matrix inverse approach generally require centralized computations and are thus not suitable for distributed implementation. If the matrix $\mathbf{A}$ is diagonally dominant, $\mathbf{x}^{(m+1)}$ can be solved in a distributed manner using the Jacobi method, which has guaranteed convergence. However, $\mathbf{A}$ is not necessarily diagonally dominant in our problem, so the Jacobi method is unsuitable.

However the computation of $\mathbf{x}^{(m+1)}$ in (V.5) is similar to the computation of $\tilde{\mathbf{x}}$ in (IV.2). Therefore the iterative distributed algorithm (IV.18), which was formulated for the latter, can also be used for the former, but with appropriate modifications. The matrix $\mathbf{D}_{\mathcal{M}}$ is replaced with $\mathbf{A}$ and the vector $\tilde{\mathbf{b}}^{(m)}$ is replaced with $\mathbf{c}^{(m)}$, both as defined above. We denote the process for obtaining $\mathbf{x}^{(m+1)}$ using Algorithm (IV.18) as

$$\mathbf{x}^{(m+1)} = \text{DLSRA}(\mathbf{A}, \mathbf{c}^{(m)}). \tag{V.8}$$

Strictly speaking, DLSRA$(\mathbf{A}, \mathbf{c}^{(m)})$ solves (V.5) approximately. Therefore the ADMM algorithm we are proposing, where (V.5) replaced with (V.8) is an approximate ADMM. The algorithm, termed as DAMRA, is summarised in Algorithm V.1 and can be viewed as a distributed version of the ADMM algorithm for graph signals.

Algorithm V.1 Distributed ADMM Recovery Algorithm (DAMRA)

```
Iteration:
1) Given $\mathbf{z}^{(m)}, \mathbf{w}^{(m)}$, apply the Algorithm IV.1 to determine $\mathbf{x}^{(m+1)}$ in a distributed manner, where $\mathbf{D}_{\mathcal{M}}$ is substituted with $\mathbf{B}_{\mathcal{M}} + 2\beta\mathbf{I} + \gamma\mathbf{H}_1^T\mathbf{H}_1$ and vector $\tilde{\mathbf{b}}^{(m)}$ with $\tilde{\mathbf{b}}_{\mathcal{M}} + 2\beta\mathbf{x}_d + \gamma\mathbf{H}_1^T(\mathbf{z}^{(m)} - \mathbf{w}^{(m)})$ respectively. Denote the solution using this distributed algorithm by $\tilde{\mathbf{x}}^{(m+1)}$.
2) Calculate $s(i) = \sum_{j\in B(i,\sigma)} H_1(i,j)\tilde{x}^{(m+1)}(j)$ and $y(i) = s(i) + w^{(m)}(i)$, then update $z^{(m+1)}(i) = S_{\beta/\gamma}(y(i)), i \in V$.
3) Update $w^{(m+1)}(i) = w^{(m)}(i) + s(i) - z^{(m+1)}(i), i \in V$.
4) Evaluate $|\tilde{x}^{(m+1)}(i) - \tilde{x}^{(m)}(i)| \leq \varepsilon, i \in V$, if yes, terminate the iteration; Otherwise, set $m = m + 1$.
```

Convergence and approximation error: The convergence of the conventional ADMM has been theoretically shown in [46]. Our proposed DAMRA is, strictly speaking, an approximate version of ADMM and therefore convergence results from conventional ADMM cannot be directly applied. However if the approximation using DLSRA is good, it can be argued heuristically that convergence is also achieved with DAMRA but a rigorous proof is not currently available. This heuristic argument has been backed by many numerical simulations performed. Table II compares the result using the exact ADMM with the approximation via DAMRA for the random geometric graph (details in subsection VI-B). The result using the exact and inexact methods are practically indistinguishable. The convergence rate between the two are also virtually the same. These results support the assertion of the approximation using DAMRA does not result in any material difference in the performance.

Remark V.1. There have been variants of the ADMM algorithms, including some which have distributed implementation. The distributed linearized ADMM (DLADMM) algorithm in [47] is one that has been recently proposed. By using the gradient information of the cost functions, it alleviates the computational burden found in conventional ADMM for certain convex optimization problems. There are distinct differences between the proposed DAMRA and the DLADMM algorithm. Firstly, the cost functions are assumed to be separable in [47] but no such assumption is required in the proposed DAMRA. Secondly, during the update of the variables in [47], the original cost function is approximated by using the gradient and a regularization term, i.e. linearization of the function. This approximation results in a deviation from the optimal solution during the update, and a greater number of iterations will therefore be required. More importantly, since the technique in [47] is gradient based, the cost function must be differentiable, i.e. precludes ℓ_1-norm terms. Our proposed DAMRA can handle ℓ_1-norm terms and no approximation to the cost function is required.

TABLE II: Recovery performance comparison of the exact ADMM with the proposed approximation via DAMRA. The random geometric graph with N vertices is used over $T = 100$ time snapshots with a 20% corruption percentage. The maximum error is defined as $\max\limits_{1\leq t\leq T-1} \|\mathbf{x}_t^{\text{exact}} - \mathbf{x}_t^{\text{inexact}}\|_2 / \|\mathbf{x}_t^{\text{exact}}\|_2$.

N	256	512	4096
Root Mean Square Error			
Exact ADMM	0.303	0.295	0.277
DAMRA	0.303	0.295	0.277
Maximum error	9.4×10^{-11}	5.1×10^{-10}	1.5×10^{-9}
Average (over 100 time snapshots) number of iterations			
Exact ADMM	20	21	32
DAMRA	20	21	32

Remark V.2. It is observed that the problem (V.1) can be categorized as a LASSO problem which can be solved by proximal methods [48]. The proximal gradient method is one of the most typical proximal methods that solves the problem by the following iterations $\mathbf{x}_{k+1} = S_\gamma(\mathbf{x}_k - t_k\nabla f(\mathbf{x}_k))$, where S_γ is the soft-thresholding operator and the function f is the sum of all ℓ_2-norm terms. Since the proximal gradient method only uses the gradient of the function f, i.e., first-order information, it suffers from the slow convergence characteristics of first-order methods. However, the ADMM can exploit the Hessian matrix of the function (via (V.8)), i.e., the second-order information, thus it has much faster convergence speed. This is also verified by the numerical results in *Table 7.1, Fig 7.1* in [48]. One motivation of this work is to develop a distributed approach to realize the conventional ADMM method. This is achieved by incorporating the Algorithm IV.1 to distributively solve the linear system in (V.5).

VI. Numerical examples

In this section, we perform several simulations to show the performance of the two proposed distributed algorithms. Both synthetic and real datasets that resides on several different types of graphs are used. The graphs considered are the circulant graph, the random geometric graph, the graphs associated with the real-world datasets of global sea-level pressure [28], [49] and the US hourly temperature records [50]. Comparisons are also made with the centralized graph total variation regulation (CGTVR) method [14], [24], the distributed GTVR (DGTVR) method [25], the quasi-Newton method (QNM) [43], and the network Newton method (NNM) with truncated order of the Neumann series being 1 [42]. The step size of the QNM and NNM methods are chosen to gives the best performance. With the CGTVR and DGTVR methods, we use the cost function in (IV.1) instead of the cost function in the original formulation in [14], [24], [25]. This is to allow for a fair comparison with our proposed methods. These existing methods however do not consider time-varying signals in their original formulations. For a fair comparison, we modify the original problem formulation in the existing methods to include a time-domain non-smoothness term $\|\mathbf{x}-\mathbf{x}_d\|_2^2$ in the objective function.

For time-varying signals, $\mathbf{X}=\{\mathbf{x}_0,\cdots,\mathbf{x}_{T-1}\}$ denotes the ensemble of graph signals at different time instants. The corrupted signal model in the simulations is given by $\mathbf{b}_t=\mathbf{x}_t+\boldsymbol{\varepsilon}_t$ $(0\le t\le T-1)$. In the experiments, for a given corruption percentage, a fraction of the total number of vertices are randomly selected. The signal values on these vertices are set to zero to simulate corruption. For the other (uncorrupted) vertices, uniformly distributed noise ε over the range $[-0.01, 0.01]$ is added to the signal. In all the examples, the order of the highpass filter $\mathbf{H}_1$ is $n=2$. For corruption percentages $10\%, 20\%, 30\%$, the radius of the subgraphs $\mathcal{G}_{k,r}$ in (II.1) are $r=2,3,3$ respectively. The radius r is chosen to ensure the overlapped decomposition (II.1) is valid for a given corruption percentage. Note however that r is substantially smaller than the diameter $\mathcal{D}$ of the graphs considered here (values shown in the tables or their captions).

The parameters α, β, and γ are chosen using a cross-validation approach. For a given graph, we generate smooth (w.r.t. vertex and time) synthetic signals, i.e. ground truth. The synthetic signal are then corrupted and the performance of the algorithm using different parameters are tested. The parameters that gave the best results are then chosen. Note that the same parameter values are used in the objective function when comparing with other algorithms, e.g. quasi-Newton.

In all cases, the stopping criterion is $e_m=\|\mathbf{x}^{(m+1)}-\mathbf{x}^{(m)}\|_\infty\le 0.0001$, i.e. the maximum signal difference between two successive iterations is smaller than the tolerance 0.0001. The performance of the recovery is measured using $\mathrm{RMSE}=\|\mathbf{x}_o-\tilde{\mathbf{x}}\|_2/\sqrt{N}$ and $\mathrm{MRE}=\|\mathbf{x}_o-\tilde{\mathbf{x}}\|_\infty/\|\mathbf{x}_o\|_\infty$, where $\mathbf{x}_o$ and $\tilde{\mathbf{x}}$ are the ground truth and the recovered signals respectively. Averages of these measures over T time snapshots, for each corruption percentage, are used to compare the performances. The measure of the convergence speed of the algorithm is the average number of iterations (AIS), where again, the averaging is over the number of time snapshots. To compare computational cost, we also report on the average CPU time (in seconds) to recover one time snapshot, denoted as ACT. All simulations are performed on a desktop computer with i7-9300 CPU and 32G memory. The main focus of this work is to develop efficient distributed approximations of the centralized algorithm. The aim is *not* to achieve better performance, in terms of lower RMSE/MRE, but to achieve fast convergence. The results (to be presented later) will show that the RMSE/MRE of the distributed algorithm is usually very similar to the centralized algorithm. This means that the distributed approximations are very good.

A. Circulant graph signal

A circulant graph of size N is defined by the generating set $S=\{s_1,\cdots,s_K\}, s_k\le N/2$. For a given vertex i, edges exist between the vertex i and the vertices $(i\pm s_k)_N$, where $()_N$ denotes the modulo operator [51]. In the experiments, we consider a circulant graph with $N=256$ and generating set $S=\{1,3\}$. We generate an $N\times 100$ time-varying signal using the dynamic model $\mathbf{x}_{t+1}=(\mathbf{I}-\tau\mathbf{L}_{\mathcal{G}}^{\mathrm{rw}})\mathbf{x}_t+\mathbf{w}_t$, where $\mathbf{w}_t$ is random noise and $\tau=0.3$. The initial signal $\mathbf{x}_0$ is given by $x_0(k)=50\sin(4\pi k/N)$, $k=0,\cdots,N-1$. The noise vector $\mathbf{w}_t$ are drawn from a uniform distribution over $[-1,1]$. For every time instant t, a percentage of the signal values at randomly selected vertices are set to zero. For example, if the corruption percentage is 10%, we randomly select 10% of the vertices and set the corresponding signal values $x_t(i)$ $(i\in\mathcal{U})$ to zero. Table III compares the performance of different recovery methods. It is observed that the proposed DLSRA algorithm possesses the fastest convergence rate.

TABLE III: Recovery performance measures on the circulant graph with $N=256$ vertices and diameter $D=44$.

Corruption %	10%	20%	30%
RMSE/MRE			
Corrupted signal	10.088/0.972	14.388/0.983	17.646/0.988
Recovered signal (all techniques)	0.182/0.023	0.271/0.027	0.346/0.031
AIS (Average number of iterations)			
DGTVR	574	579	580
QNM	81	82	82
NNM	16	16	16
DLSRA	3	3	3
DAMRA	20	24	24
ACT (Average Computational Time)			
GTVR	0.0026	0.0026	0.0026
QNM	0.1912	0.2153	0.1751
NNM	5.6×10^{-4}	4.6×10^{-4}	4.7×10^{-4}
DLSRA	8.6×10^{-5}	5.8×10^{-5}	5.2×10^{-5}
DAMRA	0.0144	0.0182	0.0170

B. Random geometric graph and signal

A random geometric graph, with N vertices randomly deployed in the region $[0,1]^2$, has an edge between two vertices if the physical distance is not larger than $\sqrt{2}N^{-1/2}$ [52], [53], [54]. We generate an $N\times 100$ time-varying signal using the dynamic model $\mathbf{x}_{t+1}=(\mathbf{I}-\tau\mathbf{L}_{\mathcal{G}}^{\mathrm{rw}})\mathbf{x}_t+\mathbf{w}_t$, where $\mathbf{w}_t$ is random noise and $\tau=0.3$. The initial signal $\mathbf{x}_0$ is given by $x_0(k)=100\cos(\pi n_{k,x}/2)\sin(\pi n_{k,y}/2)$, $k=0,\cdots,N-1$ with vertex coordinates $(n_{k,x},n_{k,y})\in[0,1]^2$. The noise vectors $\mathbf{w}_t$ are uniformly distributed over $[-1,1]$. Tables IV compares the performance of the different methods.

We also perform signal recovery for larger random geometric graphs to have an appreciation of how the algorithms scale. The results are shown in Table V. The computational load of the QNM is prohibitive for large N due to burden in calculating the inverse of quasi-Hessian many times. The proposed DLSRA algorithm performs the best by an order of magnitude in most cases.

TABLE IV: Recovery performance measures on the random geometric graph with N vertices and diameter D.

Corruption %	10%	20%	30%
RMSE/MRE, $N = 256$, $D = 22$			
Corrupted signal	15.324/0.957	21.530/0.981	26.379/0.991
Recovered signal (all techniques)	0.207/0.015	0.302/0.017	0.387/0.021
AIS, $N = 256$, $D = 22$			
DGTVR	606	609	610
QNM	78	83	83
NNM	17	17	17
DLSRA	3	3	3
DAMRA	16	20	21
ACT, $N = 256$, $D = 22$			
DGTVR	0.0050	0.0051	0.0051
QNM	0.1940	0.1788	0.1711
NNM	9.1×10^{-4}	8.1×10^{-4}	8.3×10^{-4}
DLSRA	1.6×10^{-4}	1.2×10^{-4}	1.2×10^{-4}
DAMRA	0.0842	0.0738	0.0658
RMSE/MRE, $N = 512$, $D = 30$			
Corrupted signal	15.617/0.971	22.206/0.986	27.212/0.993
Recovered signal (all techniques)	0.203/0.015	0.295/0.018	0.373/0.020
AIS, $N = 512$, $D = 30$			
DGTVR	609	611	613
QNM	79	85	85
NNM	17	17	17
DLSRA	3	3	3
DAMRA	18	21	21
ACT, $N = 512$, $D = 30$			
CGTVR	0.0013	0.0013	0.0013
DGTVR	0.0105	0.0104	0.0105
QNM	0.7676	0.8253	0.8241
NNM	0.0018	0.0017	0.0017
DLSRA	3.1×10^{-4}	2.7×10^{-4}	2.7×10^{-4}
DAMRA	0.2458	0.2223	0.1963
RMSE/MRE, $N = 4096$, $D = 88$			
Corrupted signal	15.756/0.989	22.291/0.994	27.305/0.996
Recovered signal (all techniques)	0.191/0.016	0.277/0.018	0.351/0.020
AIS, $N = 4096$, $D = 88$			
DGTVR	593	622	623
QNM	95	100	104
NNM	23	24	24
DLSRA	3	3	3
DAMRA	30	32	32
ACT, $N = 4096$, $D = 88$			
DGTVR	0.0805	0.0801	0.0840
QNM	219.9960	229.4403	241.1448
NNM	0.0462	0.0438	0.0470
DLSRA	0.0045	0.0036	0.0034
DAMRA	12.1015	10.7330	9.3783

C. The global sea-level pressure data

The global sea-level pressure dataset, from a real-world application, was originally published by the Joint Institute for the Study of the Atmosphere and Ocean [49]. The dataset consists of $T = 4599$ pentad-mean pressure snapshots ranging from the year 1948 to the year 2010. Each snapshot records pressure data from 500 stations deployed worldwide. The range of the pressure is from 94.71 kPa to 110.06 kPa. We use the 5-nearest neighbors algorithm to construct a graph of $N = 500$ nodes that captures the local interactions between the 500 stations, as shown in Fig. 1 (top). The resulting time-varying graph signal is therefore of dimension 500×4599 $(= 2299500)$. In the experiments, the time dynamics of the graph signal, characterized by the parameter τ in (III.4), is estimated by using the first 1000 snapshots of $\mathbf{x}_t$ as the training dataset. Using an optimization approach, the estimated value is $\tau = 0.3819$. For each time instant t $(t = 0, \cdots, 4598)$, a certain percentage of the values of the graph signal $\mathbf{x}_t$, randomly chosen, is corrupted. Table VI compares the performance of the different methods. It is observed that the DAMRA algorithm gives the best performance and the DLSRA algorithm has the fastest convergence.

TABLE V: Recovery performance comparisons on larger random geometric graph with N vertices and diameter D. Corruption at 20%.

N	4096	10000	20000	30000
D	88	126	175	215
RMSE				
Corrupted signal	22.291	22.349	22.444	22.431
Recovered signal (all techniques)	0.277	0.276	0.276	0.275
AIS				
DGTVR	622	618	620	619
QNM	100	90	91	92
NNM	24	17	17	17
DLSRA	3	3	3	3
ACT				
CGTVR	0.0146	0.0536	0.1370	0.26307
DGTVR	0.0801	0.2810	0.6656	1.0512
QNM	229.44	1.8×10^3	1.5×10^4	4.9×10^4
NNM	0.0438	0.0926	0.2295	0.5363
DLSRA	0.0036	0.0106	0.0244	0.0335

TABLE VI: Recovery performance measures on the Global Sea Pressure data with 500 vertices and diameter $D = 22$.

Corruption %	10%	20%	30%
RMSE/MRE			
Corrupted signal	32.017/0.988	45.276/0.992	55.453/0.995
Recovered Signal (except DAMRA)	0.135/0.013	0.200/0.015	0.258/0.017
DAMRA	0.122/0.012	0.174/0.015	0.226/0.017
AIS			
DGTVR	462	532	5762
QNM	93	100	104
NNM	79	91	985
DLSRA	4	4	4
DAMRA	212	241	278
ACT			
DGTVR	0.0080	0.0091	0.0098
QNM	0.8522	0.8934	0.9433
NNM	0.0050	0.0055	0.0059
DLSRA	3.0×10^{-4}	3.6×10^{-4}	3.8×10^{-4}
DAMRA	0.1646	0.3106	0.2934

D. US temperature sensor graph and records

The US temperature sensor network acquires temperature data on an hourly basis from 218 stations that are near to major cities across the United States [50]. Here, the graph is constructed by using the 6-nearest neighbors algorithm.

In the experiments, the temperature dataset is the 24 hours temperature record on August 1st, 2010 [50]. The time-varying

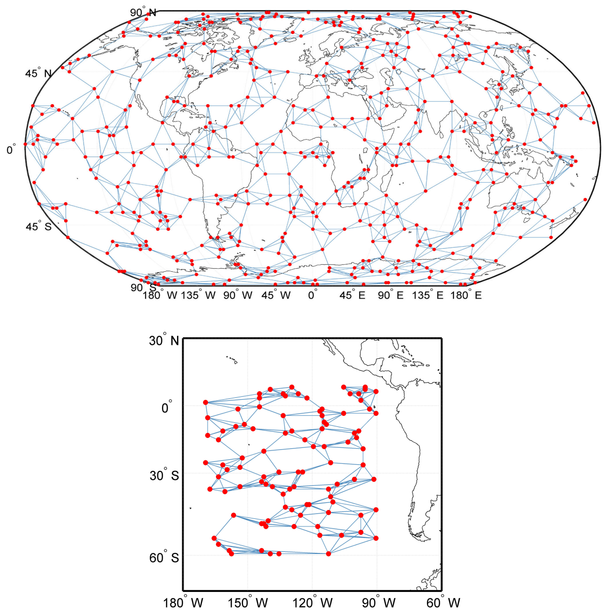
90° N
45° N
0°
45° S
90° S
180° W 135° W 90° W 45° W 0° 45° E 90° E 135° E 180° E
30° N
0°
30° S
60° S
180° W 150° W 120° W 90° W 60° W

Fig. 1: Top: Sea-level pressure graph with 500 vertices. The Beurling dimension is 2 and the density is 3.2800. Bottom: Sea surface temperature graph with $N = 100$. The Beurling dimension is 2 and the density is 2.8750.

signal is then of dimension 218×24. The first-order diffusion model (III.4) is used to model the time dynamics of the signal. Using the first 10 time snapshots of the signal $\mathbf{x}_t$, the estimated parameter of the model, using an optimization approach, is $\tau = 0.022$. For each time instant t ($t = 0, \cdots, 23$), a certain percentage of the values of the graph signal $\mathbf{x}_t$, randomly chosen, is corrupted. Table VII compares the performance of the different methods. It is observed that the DAMRA algorithm gives the best performance and the DLSRA algorithm has the fastest convergence.

TABLE VII: Recovery performance on the US hourly temperature graph with 218 vertices and diameter $D = 22$.

Corruption %	10%	20%	30%
RMSE/MRE			
Corrupted signal	23.787/0.917	34.005/0.938	41.671/0.954
Recovered signal (except DAMRA)	0.648/0.054	1.127/0.077	1.698/0.128
DAMRA	0.636/0.052	1.105/0.074	1.678/0.122
AIS			
DGTVR	4617	4713	4785
QNM	141	147	148
NNM	23	23	24
DLSRA	4	3	3
DAMRA	61	70	77
ACT			
CGTVR	0.0053	0.0050	0.0049
DGTVR	0.0352	0.0357	0.0360
QNM	0.9082	0.9475	0.9444
NNM	0.0014	0.0009	0.0009
DLSRA	2.8×10^{-4}	1.3×10^{-4}	1.0×10^{-4}
DAMRA	0.0396	0.0712	0.0644

TABLE VIII: Reconstruction performance on the sea surface temperature dataset with 100 vertices and diameter $D = 14$.

Sampling rate	90%	80%	70%
RMSE			
Corrupted signal	6.553	9.324	11.43
Recovered signal (all methods)	0.061	0.097	0.135
AIS			
Method in [28]	28	46	61
DLSRA	3	2	3
ACT			
Method in [28]	9×10^{-4}	15×10^{-4}	20×10^{-4}
DLSRA	2.9×10^{-5}	3.2×10^{-5}	4.2×10^{-5}

E. *Comparison with the recovery methods in [28]*

In [28], the modeling of the time-varying nature of the signals is achieved by assuming that the temporal difference signal $\Delta_t \equiv \mathbf{x}_t - \mathbf{x}_{t-1}$ is smooth. The penalty measure used in [28] is based on the quadratic form of Δ_t with the Laplacian matrix as the coefficients in the quadratic form. We however use some form of linear prediction to estimate the signal at the current time instant $\mathbf{x}_d = \sum_{l=1}^{t} \mathbf{C}_l \tilde{\mathbf{x}}_{t-l}$. The predicted signal is assumed to be close to the actual observed signal $\mathbf{x}_t$. Only the ℓ_2-norm is considered in [28], but in our work both the ℓ_2- and ℓ_1-norms, which can promote sparsity, are considered. The most important difference however is with respect to the distributed implementation. The approach in [28] employs the classical gradient descent method that only exploit the first order information, while our proposed algorithms exploit the second order derivative information that leads to fast convergence. The algorithm in [28] also does not have the light/heavy property discussed in Section IV.

We next compare the performance of the distributed algorithm in [28] with our DLSRA in the reconstruction of the sea surface temperature, using the dataset published by the Earth System Research Laboratory [55]. We adopt the dataset in [28], acquired from 100 stations across the Pacific ocean from 170° west to 90° west and from 60° south to 10° north with a dynamic range from -1.32°C to 30.72°C. Each station records, over a period of 1733 months, the monthly mean sea surface temperature from January 1870 to May 2014. The underlying graph $\mathcal{G}$ has 100 vertices, each of which represents an observation station, and is constructed by connecting the 5-nearest neighboring stations, in terms of physical distance [55], as shown in Fig.1 (bottom).

Three sampling rates, 90%, 80% and 70% (the corresponding corruption percentages are 10%, 20% and 30%), are used in the simulations. Table VIII compares the performance of the different methods. It is seen that all methods give the same performance, indicating convergence to the optimal solution (which is obtained by the centralized method). However the DLSRA has significantly faster convergence than the method in [28]. The computational time is also substantially lower in the former.

F. Recovery of piecewise smooth signals

The underlying signals we have considered so far are smooth over the entire graph vertex domain. However there could be situations where the underlying signal is discontinuous over limited number of locations but is smooth otherwise, i.e. piecewise smooth. These discontinuities manifest themselves in the highpass components of the signal. This is similar to edges with images. We would like to compare between the performance of the recovery algorithms using the l_2-norm (DLSRA) with the l_1-norm (DAMRA). Consider the time-invariant signal on Minnesota traffic graph signal shown in Fig. 2 which is piecewise smooth. A 10% corruption is applied to this signal and the difference between the corrupted signal and original signal is also shown in the figure. Using the l_2 penalty norm, the recovered signal has RMSE $= 0.0625$. Using the l_1 penalty norm the recovered signal has RMSE $= 0.0044$ which is substantially lower. Fig. 2 also show the error signals using the two norms. The error near the discontinuities is smaller with the l_1-norm. This result is consistent with what is well known in images, the l_1-norm is better than the l_2-norm for preserving edge features in images.

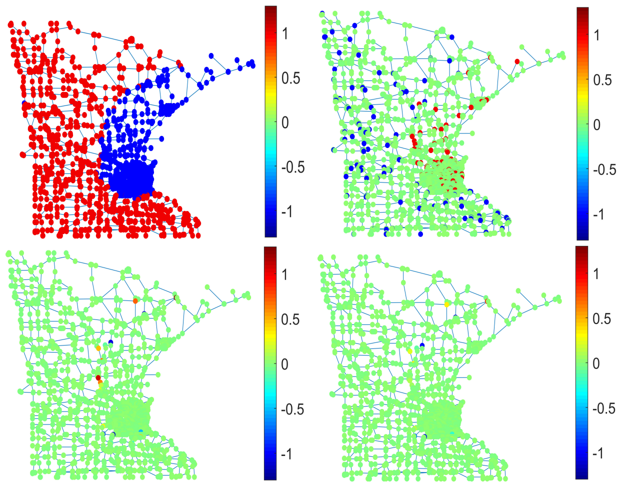

Fig. 2: Minnesota traffic graph with 2642 vertices and 3303 edges. Top left: underlying original piecewise smooth signal. Top right: difference between the corrupted signal and original signal. Bottom left: error (original - recovered) using the ℓ_2-norm. Bottom right: error using the ℓ_1-norm.

VII. Conclusions

The work here has addressed the recovery of time-varying graph signals. New notions of smoothness over time and graph topology have been proposed. These notions, via the use of non-smoothness penalty terms, were then exploited in the optimization problem formulations. In the first formulation, only the ℓ_2-norm appeared in the objective function. In the second formulation, both the ℓ_1- and ℓ_2-norms appeared in the objective function. Two distributed algorithms, abbreviated as DLSRA and DARMA, were proposed to solve the two problems. Extensive numerical simulations demonstrate that these algorithms give excellent quality solutions. However, the main advantage with the DLSRA algorithm is that it has fast convergence.

VIII. Appendix

A. Proof of (IV.14)

Write $\mathbf{M}_k^r(\mathbf{M}_k^{2r}\mathbf{D}_\mathcal{M}\mathbf{M}_k^{2r})^\dagger\mathbf{M}_k^{2r}\mathbf{D}_\mathcal{M}(\mathbf{M}_k^{2r+2\sigma}-\mathbf{M}_k^{2r}) = (g_k(i,j))_{i,j\in V}, k\in\mathcal{M}$. Applying the formula *(A.3)* from [13], we obtain

$$|g_k(i,j)| \le D_1(\mathcal{G})(2\sigma+1)^d\kappa\exp\Big(-\frac{\theta}{2\sigma}r+\theta\Big) \qquad \text{(VIII.1)}$$

for $i\in B(k,r)$ and $j\in B(k,2r+2\sigma)\backslash B(k,2r)$, and

$$g_k(i,j)=0 \qquad \text{(VIII.2)}$$

when either $i\notin B(k,r)$ or $j\notin B(k,2r+2\sigma)\backslash B(k,2r)$. Write $\mathbf{I}-\mathbf{J}\mathbf{D}_\mathcal{M}=(\tilde g(i,j))_{i,j\in V}$. By (IV.10) and the observation that $\mathbf{D}_\mathcal{M}$ has geodesic-width at most 2σ, we have

$$\mathbf{I}-\mathbf{J}\mathbf{D}_\mathcal{M} = \Big(\sum_{k'\in\mathcal{M}}\mathbf{M}_{k'}^r\Big)^{-1}\sum_{k\in\mathcal{M}}\mathbf{M}_k^r(\mathbf{M}_k^{2r}\mathbf{D}_\mathcal{M}\mathbf{M}_k^{2r})^\dagger \times\mathbf{M}_k^{2r}\mathbf{D}_\mathcal{M}(\mathbf{M}_k^{2r+2\sigma}-\mathbf{M}_k^{2r}),$$

which implies that

$$\tilde g(i,j)=\frac{\sum_{k\in\mathcal{M}\cap B(i,r)}g_k(i,j)}{\sum_{k\in\mathcal{M}\cap B(i,r)}1},\ i,j\in V. \qquad \text{(VIII.3)}$$

This together with (VIII.1) and (VIII.2) implies that

$$|\tilde g(i,j)| \le D_1(\mathcal{G})(2\sigma+1)^d\kappa\exp\Big(-\frac{\theta}{2\sigma}r+\theta\Big) \qquad \text{(VIII.4)}$$

when $\rho(i,j)\le 3r+2\sigma$, and

$$\tilde g(i,j)=0 \qquad \text{(VIII.5)}$$

when $\rho(i,j)>3r+2\sigma$. Therefore the estimate (IV.14) follows from (II.2), (VIII.4) and (VIII.5).

B. Proof of Theorem IV.2

Set $\mathbf{e}^{(m)}=\tilde{\mathbf{x}}-\mathbf{x}^{(m)}$. From formula (IV.17), we have

$$\mathbf{e}^{(m)}=\big(\mathbf{I}-\mathbf{J}\mathbf{D}_\mathcal{M}\big)\mathbf{e}^{(m-1)}. \qquad \text{(VIII.6)}$$

From (IV.9) and (IV.11), we have

$$\|(\mathbf{I}-\mathbf{J}\mathbf{D}_\mathcal{M})\tilde{\mathbf{x}}\|_p\le\delta\|\tilde{\mathbf{x}}\|_p \qquad \text{(VIII.7)}$$

Since there is no restriction on the signal vector $\tilde{\mathbf{x}}$, we can conclude that (VIII.7) holds true for any arbitrary signal vector. Therefore, we have

$$\|\mathbf{e}^{(m)}\|_p\le\delta\|\mathbf{e}^{(m-1)}\|_p \qquad \text{(VIII.8)}$$

where $\delta<1$ implies that the convergence rate is exponential.

References

[1] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[2] Aliaksei Sandryhaila and Jose MF Moura. Big data analysis with signal processing on graphs. *IEEE Signal Processing Magazine*, 31(5):80–90, 2014.

[3] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7):1644–1656, 2013.

[4] Aliaksei Sandryhaila and Jose MF Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, 2014.

[5] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José M F Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[6] Gene Cheung, Enrico Magli, Yuichi Tanaka, and Michael K Ng. Graph spectral image processing. *Proceedings of the IEEE*, 106(5):907–930, 2018.

[7] Xuesong Shi, Hui Feng, Muyuan Zhai, Tao Yang, and Bo Hu. Infinite impulse response graph filters in wireless sensor networks. *IEEE Signal Processing Letters*, 22(8):1113–1117, 2015.

[8] Adnan Gavili and Xiao-Ping Zhang. On the shift operator, graph frequency, and optimal filtering in graph signal processing. *IEEE Transactions on Signal Processing*, 65(23):6303–6318, 2017.

[9] Santiago Segarra, Antonio G Marques, and Alejandro Ribeiro. Optimal graph-filter design and applications to distributed linear network operators. *IEEE Transactions on Signal Processing*, 65(15):4117–4131, 2017.

[10] Sunil K Narang and Antonio Ortega. Perfect reconstruction two-channel wavelet filter banks for graph structured data. *IEEE Transactions on Signal Processing*, 60(6):2786–2799, 2012.

[11] Sunil K Narang and Antonio Ortega. Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs. *IEEE transactions on signal processing*, 61(19):4673–4685, 2013.

[12] Yuichi Tanaka and Akie Sakiyama. *M*-channel oversampled graph filter banks. *IEEE Transactions on Signal Processing*, 62(14):3578–3590, 2014.

[13] Junzheng Jiang, Cheng Cheng, and Qiyu Sun. Nonsubsampled graph filter banks: Theory and distributed algorithms. *IEEE Transactions on Signal Processing*, 67(15):3938–3953, 2019.

[14] Siheng Chen, Aliaksei Sandryhaila, José M F Moura, and Jelena Kovačević. Signal recovery on graphs: Variation minimization. *IEEE Transactions on Signal Processing*, 63(17):4609–4624, 2015.

[15] Jonathan Mei and José MF Moura. Signal processing on graphs: Causal modeling of unstructured data. *IEEE Transactions on Signal Processing*, 65(8):2077–2092, 2016.

[16] Vassilis N Ioannidis, Yanning Shen, and Georgios B Giannakis. Semi-blind inference of topologies and dynamical processes over dynamic graphs. *IEEE Transactions on Signal Processing*, 67(9):2263–2274, 2019.

[17] Gonzalo Mateos, Santiago Segarra, Antonio G Marques, and Alejandro Ribeiro. Connecting the dots: Identifying network structure via graph signal processing. *IEEE Signal Processing Magazine*, 36(3):16–43, 2019.

[18] Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

[19] Georgios B Giannakis, Yanning Shen, and Georgios Vasileios Karanikolas. Topology identification and learning over graphs: Accounting for nonlinearities and dynamics. *Proceedings of the IEEE*, 106(5):787–807, 2018.

[20] Christine Guillemot and Olivier Le Meur. Image inpainting: Overview and recent advances. *IEEE signal processing magazine*, 31(1):127–144, 2014.

[21] Veepin Kumar, Jayanta Mukherjee, and Shyamal Kumar Das Mandal. Image inpainting through metric labeling via guided patch mixing. *IEEE Transactions on Image Processing*, 25(11):5212–5226, 2016.

[22] Ali Mosleh, Nizar Bouguila, and Abdessamad Ben Hamza. Automatic inpainting scheme for video text detection and removal. *IEEE Transactions on image processing*, 22(11):4460–4472, 2013.

[23] Mounira Ebdelli, Olivier Le Meur, and Christine Guillemot. Video inpainting with short-term windows: application to object removal and error concealment. *IEEE Transactions on Image Processing*, 24(10):3034–3047, 2015.

[24] Siheng Chen, Aliaksei Sandryhaila, George Lederman, Zihao Wang, José MF Moura, Piervincenzo Rizzo, Jacobo Bielak, James H Garrett, and Jelena Kovačević. Signal inpainting on graphs via total variation minimization. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8267–8271. IEEE, 2014.

[25] Siheng Chen, Aliaksei Sandryhaila, and Jelena Kovačević. Distributed algorithm for graph signal inpainting. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3731–3735. IEEE, 2015.

[26] Mikhail Tsitsvero, Sergio Barbarossa, and Paolo Di Lorenzo. Signals on graphs: Uncertainty principle and sampling. *IEEE Transactions on Signal Processing*, 64(18):4845–4860, 2016.

[27] Paolo Di Lorenzo, Paolo Banelli, Sergio Barbarossa, and Stefania Sardellitti. Distributed adaptive learning of graph signals. *IEEE Transactions on Signal Processing*, 65(16):4193–4208, 2017.

[28] Kai Qiu, Xianghui Mao, Xinyue Shen, Xiaohan Wang, Tiejian Li, and Yuantao Gu. Time-varying graph signal reconstruction. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):870–883, 2017.

[29] Daniel Romero, Vassilis N Ioannidis, and Georgios B Giannakis. Kernel-based reconstruction of space-time functions on dynamic graphs. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):856–869, 2017.

[30] Francesco Grassi, Andreas Loukas, Nathanaël, and Benjamin Ricaud. A time-vertex signal processing framework: scalable processing and meaningful representations for time-series on graphs. *IEEE Transactions on Signal Processing*, 66(3):817–829, 2018.

[31] Elvin Isufi, Andreas Loukas, Nathanael Perraudin, and Geert Leus. Forecasting time series with varma recursions on graphs. *IEEE Transactions on Signal Processing*, 67(18):4870–4885, 2019.

[32] Siheng Chen, Rohan Varma, Aarti Singh, and Jelena Kovačević. Signal recovery on graphs: Fundamental limits of sampling strategies. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):539–554, 2016.

[33] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[34] David Kaplan. *Structural equation modeling: Foundations and extensions*, volume 10. Sage Publications, 2008.

[35] Cheng Cheng, Yingchun Jiang, and Qiyu Sun. Spatially distributed sampling and reconstruction. *Applied and Computational Harmonic Analysis*, 47(1):109–148, 2019.

[36] Fan R. K. Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[37] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[38] Mario Coutino, Elvin Isufi, and Geert Leus. Advances in distributed graph filtering. *IEEE Transactions on Signal Processing*, 67(9):2320–2333, 2019.

[39] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming (web page and software), 2009.

[40] Federico S Cattivelli and Ali H Sayed. Diffusion LMS strategies for distributed estimation. *IEEE Transactions on Signal Processing*, 58(3):1035–1048, 2010.

[41] David I Shuman, Pierre Vandergheynst, Daniel Kressner, and Pascal Frossard. Distributed signal processing via chebyshev polynomial approximation. *IEEE Transactions on Signal and Information Processing over Networks*, 4(4):736–751, 2018.

[42] Aryan Mokhtari, Qing Ling, and Alejandro Ribeiro. Network newton distributed optimization methods. *IEEE Transactions on Signal Processing*, 65(1):146–161, 2017.

[43] Mark Eisen, Aryan Mokhtari, and Alejandro Ribeiro. Decentralized quasi-newton methods. *IEEE Transactions on Signal Processing*, 65(10):2613–2628, 2017.

[44] Jérôme Darbon and Marc Sigelle. Image restoration with discrete constrained total variation part i: Fast and exact optimization. *Journal of Mathematical Imaging and Vision*, 26(3):261–276, 2006.

[45] Laurent Condat. A direct algorithm for 1-D total variation denoising. *IEEE Signal Processing Letters*, 20(11):1054–1057, 2013.

[46] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[47] Xuanyu Cao and KJ Ray Liu. Distributed linearized ADMM for network cost minimization. *IEEE Transactions on Signal and Information Processing over Networks*, 4(3):626–638, 2018.

[48] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):123–231, 2014.

[49] Sea-level pressure, 1948−2010, http://research.jisao.washington.edu/data_sets/reanalysis, 2016.

[50] Jin Zeng, Gene Cheung, and Antonio Ortega. Bipartite approximation for graph wavelet signal decomposition. *IEEE Transactions on Signal Processing*, 65(20):5466–5480, 2017.

[51] Venkatesan N Ekambaram, Giulia C Fanti, Babak Ayazifar, and Kannan Ramchandran. Circulant structures and graph signal processing. In *2013 IEEE International Conference on Image Processing*, pages 834–838. IEEE, 2013.

[52] David I Shuman, Mohammad Javad Faraji, and Pierre Vandergheynst. A multiscale pyramid transform for graph signals. *IEEE Transactions on Signal Processing*, 64(8):2119–2134, 2016.

[53] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

[54] Mathew Penrose et al. *Random geometric graphs*, volume 5. Oxford University press, 2003.

[55] Sea surface temperature (SST) v2, http://www.esrl.noaa.gov/psd/data/gridded/data.noaa.oisst.v2.html, 2015.