Thanou and Frossard *EURASIP Journal on Advances in Signal Processing* (2018) 2018:67
https://doi.org/10.1186/s13634-018-0584-2

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Learning of robust spectral graph dictionaries for distributed processing

Dorina Thanou[1*] and Pascal Frossard[2]

Abstract

We consider the problem of distributed representation of signals in sensor networks, where sensors exchange quantized information with their neighbors. The signals of interest are assumed to have a sparse representation with spectral graph dictionaries. We further model the spectral dictionaries as polynomials of the graph Laplacian operator. We first study the impact of the quantization noise in the distributed computation of matrix-vector multiplications, such as the forward and the adjoint operator, which are used in many classical signal processing tasks. It occurs that the performance is clearly penalized by the quantization noise, whose impact directly depends on the structure of the spectral graph dictionary. Next, we focus on the problem of sparse signal representation and propose an algorithm to learn polynomial graph dictionaries that are both adapted to the graph signals of interest and robust to quantization noise. Simulation results show that the learned dictionaries are efficient in processing graph signals in sensor networks where bandwidth constraints impose quantization of the messages exchanged in the network.

Keywords: Distributed processing, Graph signal processing, Quantization, Polynomial dictionaries, Sparse approximation

1 Introduction

Wireless sensor networks have been widely used for applications such as surveillance, weather monitoring, or automatic control, that often involve distributed signal processing methods. Such methods are typically designed by assuming local inter-sensor communication, i.e., communication between neighbor sensors, in order to achieve a global objective over the network. While in theory the performance depends mainly on the sensor network topology, it is largely connected to the power or communication constraints and limited precision operations in practical systems. In general, the information exchanged by the sensors has to be quantized prior to transmission due to limited communication bandwidth and limited computational power. This quantization process may lead to significant performance degradation, if the system is not designed to handle it properly.

In distributed settings, the set of sensors is generally represented by the vertices of a graph, whose edge weights capture the pairwise relationships between the vertices. A graph signal is defined as a function that assigns a real value to each vertex, which corresponds to the quantity measured by the sensor, such as the current temperature or the road traffic level at a particular time instance. Graph dictionary representations are certainly powerful and promising tools for representing graph signals [1]. In particular, when graph signals mostly capture the effect of a few processes on a graph topology, the graph signals can be modeled as the linear combinations of a small number of constitutive components in a spectral graph dictionary [2, 3]. Such dictionaries, which can also be seen as a set of spectral filters on graphs, incorporate the intrinsic geometric structure of the irregular graph domain into the atoms and are able to capture different processes evolving on the graph. Moreover, they can exhibit a polynomial structure [3] or can be approximated by polynomials of a graph matrix [4, 5] (e.g., spectral graph wavelets). As such, they can be implemented distributively and therefore represent ideal operators for graph signal processing in sensor networks[1].

The distributed processing of graph signals in sensor networks from a graph signal representation perspective has received a considerable amount of attention recently. The works in [4, 5] study how a general class of linear graph operators can be approximated by shifted

*Correspondence: dorina.thanou@epfl.ch
[1]Swiss Data Science Center, EPFL, Lausanne, Switzerland
Full list of author information is available at the end of the article

Chebyshev polynomials and can eventually be applied to distributed processing tasks such as smoothing, denoising, and semi-supervised classification. A distributed least square reconstruction algorithm of bandlimited graph signals has been proposed in [6], and a distributed inpainting algorithm for smooth graph signals was introduced in [7]. The finite-time behavior of distributed algorithms on arbitrary graphs is studied in [8], showing that any arbitrary initial condition after many distributed linear iterations can be forced to lie on a specific subspace. More recently, distributed strategies for adaptive learning of bandlimited graph signals from a probabilistic sampling point of view were developed in [9]. The works of [10–12] study the design of graph filters, i.e., distributed linear operators for in the context of graph signal processing, with [11, 12] focusing specifically on ARMA and IIR filters, respectively. However, neither of these works takes into consideration the effect of the quantization noise that is of significant importance in realistic sensor network scenarios with rate and communication constraints. Limited attention to the effect of quantization is given in [13, 14], in the context of linear prediction with graph filters. However, the focus of those works is the design of generic graph filters without taking into account the effect of quantization in the design itself. The effect of quantization in the design of distributed representations for graph signals has been studied in our preliminary work [15], and more recently in [16]. The latter however focuses on the approximation of the frequency response of known graph FIR filters with filters that are robust to quantization noise, rather than the distributed processing of the graph signals.

In this paper, we build on our previous work [15], and we study the effect of quantization in distributed graph signal representations. In particular, we first derive the quantization error that appears in the distributed computation of different operators defined on graph spectral dictionaries with polynomial structures. Our analysis is quite generic and can be applied to every graph spectral dictionary that can be approximated by polynomials of the graph Laplacian (e.g., spectral graph wavelets approximated with Chebyshev polynomials) [4, 5]. We then consider the problem of sparse representation of graph signals that is implemented in a distributed way with an iterative soft thresholding algorithm. We analyze the convergence of the algorithm and show how it depends on the quantization noise, whose influence is itself governed by the characteristics of the dictionary. We finally propose an algorithm for learning polynomial graph dictionaries that permits to control the robustness of distributed algorithms to quantization noise. Experimental results illustrate the dictionary design trade-offs between accurate signal representation and robustness to quantization errors. They show in particular that it is necessary to sacrifice on signal approximation performance for ensuring proper convergence of distributed algorithms in low-bit rate settings. To the best of our knowledge, the work done in this paper is definitely one of the first steps toward designing quantization-aware dictionaries for distributed signal processing.

The rest of the paper is organized as follows. In Section 2, we model the sensor network with a graph, and we recall the use of polynomial graph dictionaries for distributed processing of graph signals. We study the quantization error that appears in the distributed computations with polynomial graph dictionaries in Section 3, and in Section 4, we analyze the specific case of the sparse approximation of graph signals. In Section 5, we propose an algorithm for learning polynomial graph dictionaries that are robust to quantization noise. Finally, in Section 6, we evaluate the performance of our algorithm in both synthetic and real world signals.

2 Polynomial graph dictionaries for distributed signal representation

For the sake of completeness, we recall some of the basic concepts of signal representation on graphs, and we introduce notations that are needed for the rest of this paper. First, we model the sensor network topology as a weighted graph, whose connectivity defines the communication channels. Moreover, we recall briefly the sparse signal model that is based on polynomial dictionaries of the graph Laplacian. Such dictionaries lead to efficient distributed computations, as illustrated in what follows.

2.1 Notation

Throughout the paper, lowercase normal (e.g., a), lowercase bold (e.g., $\mathbf{x}$), and uppercase bold (e.g., $\mathbf{D}$) letters denote scalars, vectors, and matrices, respectively. Unless otherwise stated, calligraphic capital letters (e.g., $\mathcal{V}$) represent sets.

2.2 Distributed sensor network topology

We consider a sensor network topology that is modeled as a weighted, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V} \in \{1, \ldots, N\}$ represents the set of sensor nodes and $N = |\mathcal{V}|$ denotes the number of nodes. An edge denoted by an unordered pair $\{i, j\} \in \mathcal{E}$ represents a communication link between two sensor nodes i and j. Moreover, a positive weight $\mathbf{W}_{ij} > 0$ is assigned to each edge $\{i, j\} \in \mathcal{E}$. $\mathbf{D}$ is a diagonal degree matrix that contains as elements the sum of each row of the matrix $\mathbf{W}$. The set of neighbors for node i is finally denoted as $\mathcal{N}_i = \{j | \{i, j\} \in \mathcal{E}\}$. The normalized graph Laplacian operator is finally defined as $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$. We denote its eigenvectors by $\boldsymbol{\chi} = [\boldsymbol{\chi}_1, \boldsymbol{\chi}_2, ..., \boldsymbol{\chi}_N]$, and the spectrum of the eigenvalues by:

$$\boldsymbol{\Lambda} := \left\{0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{(N-1)} \leq 2\right\}.$$

The eigenvalues of the graph Laplacian provide a notion of frequency on the graph and the corresponding eigenvectors define the graph Fourier transform [1]. In particular, for any function $\boldsymbol{y}$ defined on the vertices of the graph, the graph Fourier transform $\hat{\boldsymbol{y}}$ at frequency λ_ℓ is defined as:

$$\hat{\boldsymbol{y}}(\lambda_\ell) = \langle \boldsymbol{y}, \boldsymbol{\chi}_\ell \rangle = \sum_{n=1}^{N} \boldsymbol{y}(n) \boldsymbol{\chi}_\ell^*(n),$$

while the inverse graph Fourier transform is obtained by projecting back in the (orthonormal) graph Fourier basis.

We note that the underlying assumption of this work is that the structure of the signal is captured by the communication graph. This assumption is generally true in sensor network applications, where the communication is restricted to neighboring sensors. As an example, the transportation graph is expected to have a strong influence on the traffic sensor measurements. The communication graph in that case can be considered as a proxy for the transportation graph.

2.3 Sparse graph signal model

We model the sensor signals as sparse linear combinations of (overlapping) graph patterns $\widehat{g}(\cdot)$, positioned at different vertices [3]. Each pattern defined in the graph spectral domain captures the form of the graph signal in the neighborhood of a vertex, and it can be considered as a function whose values depend on the local connectivity around that vertex. We represent the translation of such a pattern to different vertices of the graph [17] through a graph operator defined as:

$$\widehat{g}(\mathbf{L}) = \boldsymbol{\chi} \widehat{g}(\boldsymbol{\Lambda}) \boldsymbol{\chi}^T. \tag{1}$$

The generating kernel $\widehat{g}(\cdot)$, which is a function of the eigenvalues of the Laplacian, characterizes the graph pattern in the spectral domain. One can design graph operators consisting of localized atoms in the vertex domain by taking the kernel $\widehat{g}(\cdot)$ in (1) to be a smooth polynomial function of degree K [3, 17]:

$$\widehat{g}(\lambda_\ell) = \sum_{k=0}^{K} \alpha_k \lambda_\ell^k, \quad \ell = 0, ..., N-1. \tag{2}$$

A graph dictionary is then defined as a concatenation of subdictionaries in the form $\boldsymbol{\mathcal{D}} = \left[\widehat{g_1}(\mathbf{L}), \widehat{g_2}(\mathbf{L}), ..., \widehat{g_S}(\mathbf{L})\right]$, where each subdictionary s is defined as:

$$\widehat{g_s}(\mathbf{L}) = \boldsymbol{\chi} \left(\sum_{k=0}^{K} \alpha_{sk} \boldsymbol{\Lambda}^k \right) \boldsymbol{\chi}^T = \sum_{k=0}^{K} \alpha_{sk} \mathbf{L}^k. \tag{3}$$

A subdictionary $\widehat{g}(\mathbf{L})$ corresponds to a matrix, each column of which is an atom positioned at a different vertex of the graph. For the sake of simplicity, we assume that all the subdictionaries are of the same order K. However, all the results presented in the manuscript hold for subdictionaries with different polynomial degrees.

Finally, a graph signal $\boldsymbol{y}$ can be expressed as a linear combination of a set of atoms generated from different graph kernels $\left\{\widehat{g_s}(\cdot)\right\}_{s=1,2,...,S}$,

$$\boldsymbol{y} = \sum_{s=1}^{S} \widehat{g_s}(\mathbf{L}) \boldsymbol{x_s} = \sum_{s=1}^{S} \boldsymbol{\mathcal{D}}_s \boldsymbol{x_s},$$

where we have set $\boldsymbol{\mathcal{D}}_s = \widehat{g_s}(\mathbf{L}) \in \mathbb{R}^{N \times N}$, and $\boldsymbol{x_s} \in \mathbb{R}^N$ are the coefficients in the linear combination. One can then learn the polynomial coefficients numerically from a set of training signals that live on the graph as shown in [3] in order to adapt the dictionary to specific classes of signals. Another example of the dictionary $\boldsymbol{\mathcal{D}}$ is the spectral graph wavelet dictionary [17] with pre-defined spline coefficients, or more generally the union of graph Fourier multipliers that can be efficiently approximated with Chebyshev polynomials [5].

2.4 Distributed computation of the graph operators

An important benefit of polynomial graph dictionaries described above is the fact that they can be efficiently stored and implemented in distributed signal processing tasks. Each polynomial dictionary can be constructed locally, i.e., by exchanging only information between nodes that are connected by an edge on the graph. For the sake of completeness, we recall here the distributed computation of some of these operators. More details can be found in [4].

The distributed computation of $\boldsymbol{\mathcal{D}}^T \boldsymbol{y}$ requires first the computation of the different powers of the Laplacian matrix, i.e., $\left\{\mathbf{L}^0 \boldsymbol{y}, \mathbf{L}^1 \boldsymbol{y}, \mathbf{L}^2 \boldsymbol{y}, ..., \mathbf{L}^K \boldsymbol{y}\right\}$, in a distributed way. The latter can be done efficiently by successive multiplications of the matrix $\mathbf{L}$ with the signal $\boldsymbol{y}$ over K iterations. We introduce a new variable $\boldsymbol{z}_k \in \mathbb{R}^N$ in the computation of $\boldsymbol{\mathcal{D}}^T \boldsymbol{y}$, which represents the value transmitted during the k^{th} iteration, with $\boldsymbol{z}_0 = \boldsymbol{y}$. Initially, each node transmits its component of $\boldsymbol{z}_0$ only to its one-hop neighbors on the graph. After receiving the values from its neighbors, it updates its component as a linear combination of its own value in $\boldsymbol{z}_0$ and the values received from its neighbors as follows:

$$\boldsymbol{z}_1 = \mathbf{L} \boldsymbol{z}_0. \tag{4}$$

At the next iteration, the values of $\boldsymbol{z}_1$ are exchanged locally in the network. The procedure is repeated over K iterations, and the exchanged messages are computed based on the previous recursive update relationship. After knowing $\{\boldsymbol{z}_0(n), \boldsymbol{z}_1(n), ..., \boldsymbol{z}_K(n)\}$, each node n can compute the n^{th} component of $\boldsymbol{\mathcal{D}}_s \boldsymbol{y}$ by a simple linear combination with the polynomial coefficients i.e., $(\boldsymbol{\mathcal{D}}_s \boldsymbol{y})(n) =$

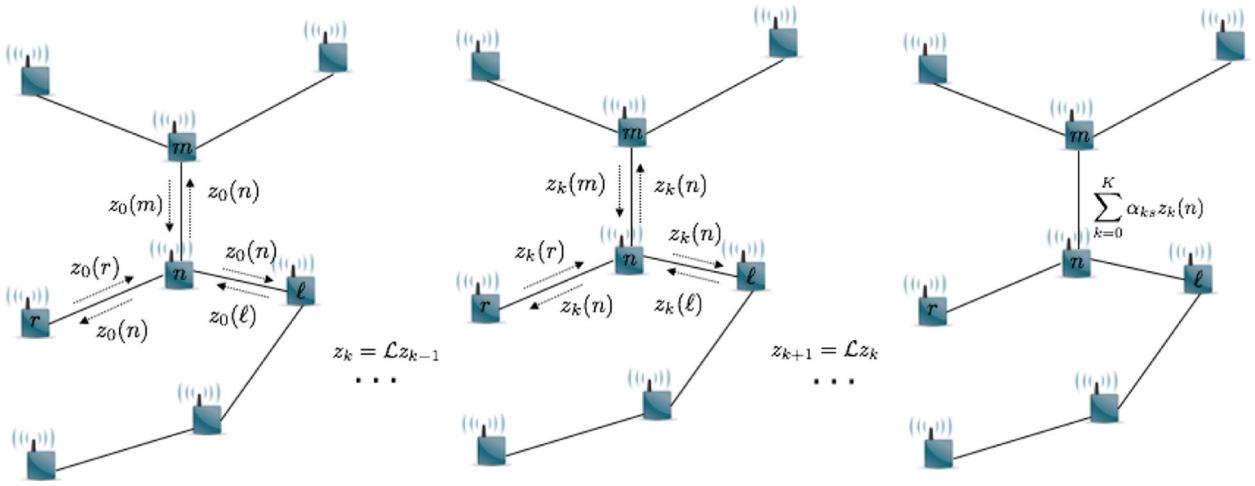


Fig. 1 Distributed computation of $\mathcal{D}_s \boldsymbol{y}$ in a sensor network. Sensor n exchanges messages with its one-hop neighbors for K iterations. After each iteration, the received messages are filtered with weights defined by the graph Laplacian, i.e., $\boldsymbol{z}_k(n) = (\mathbf{L}\boldsymbol{z}_{k-1})(n)$, and are transmitted locally in the network. $(\mathcal{D}_s \boldsymbol{y})(n)$ is computed as a linear combination of the messages exchanged during the K iterations

$\sum_{k=1}^{K} \alpha_{sk} \boldsymbol{z}_k(n)$. The same can be done for the different subdictionaries and their polynomial coefficients. An illustration of the process is shown in Fig. 1. The main steps are given in Algorithm 1.

Algorithm 1 Distributed computation of $\mathcal{D}^T \boldsymbol{y}$

1: **Input at node n:** $\boldsymbol{y}(n), \mathbf{L}_{n,:}, \alpha = [\alpha_1; ...; \alpha_S]$
2: **Output at node n:** $(\mathcal{D}^T \boldsymbol{y})\big((s-1)N + n\big)$ for all $s = \{1, ..., S\}$
3: Transmit $\boldsymbol{z}_0(n) = \boldsymbol{y}(n)$ to all neighbors in $\mathcal{N}_n$
4: Receive $\boldsymbol{z}_0(m) = \boldsymbol{y}(m)$ from all neighbors in $\mathcal{N}_n$
5: **for** $k = 2, ..., K$ **do:**
6: Transmit $\boldsymbol{z}_{k-1}(n) = (\mathbf{L}^T \boldsymbol{z}_{k-2})(n)$ to all the neighbors
7: Receive $\boldsymbol{z}_{k-1}(m) = (\mathbf{L}^T \boldsymbol{z}_{k-2})(m)$ from all the neighbors $m \in \mathcal{N}_n$.
8: **end for**
9: Compute $\boldsymbol{z}_K(n) = (\mathbf{L}\boldsymbol{z}_{K-1})(n)$
10: **for** $s = \{1, ..., S\}$ **do**
11: Compute $(\mathcal{D}^T \boldsymbol{y})\big((s-1)N + n\big) = \sum_{k=0}^{K} \alpha_{ks} \boldsymbol{z}_k(n)$
12: **end for**

Following the same reasoning, the forward operator $\mathcal{D}\boldsymbol{x}$ can be computed in a distributed way. We recall that $\mathcal{D}\boldsymbol{x} = \sum_{s=1}^{S} \mathcal{D}_s \boldsymbol{x}_s$, where $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_s) \in \mathbb{R}^{SN}$ is a vector with the sparse codes $\boldsymbol{x}_s$ corresponding to subdictionary $\mathcal{D}_s$. Each of the components in the summation is computed by sending iteratively the powers of the Laplacian as follows. For each of the subdictionaries, we define a new variable $\boldsymbol{z}_{s,0} = \boldsymbol{x}_s$. The transmitted value of this variable by sensor n at iteration k is:

$$\boldsymbol{z}_{s,k}(n) = (\mathbf{L}\boldsymbol{z}_{s,k-1})(n).$$

Each sensor can then compute its component in $\mathcal{D}x$, which can be expressed as follows in a vector form:

$$\mathcal{D}\boldsymbol{x} = \sum_{s=1}^{S} \mathcal{D}_s \boldsymbol{x}_s = \sum_{s=1}^{S} \sum_{k=0}^{K} \alpha_{sk} \boldsymbol{z}_{s,k}.$$

The main steps of the distributed algorithm for the computation of $\mathcal{D}\boldsymbol{x}$ are shown in Algorithm 2. Finally, the operator $\mathcal{D}^T\mathcal{D}\boldsymbol{x}$ can be implemented in distributed settings by first computing $\mathcal{D}\boldsymbol{x}$ and then $\mathcal{D}^T\mathcal{D}\boldsymbol{x}$ by following Algorithms 2 and 1, respectively.

Algorithm 2 Distributed computation of $\mathcal{D}\boldsymbol{x}$

1: **Input at node n:** $\boldsymbol{x}\big((s-1)N + n\big)$, for all $s = \{1, ..., S\}, \mathbf{L}_{n,:}, \alpha = [\alpha_1; ...; \alpha_S]$
2: **Output at node n:** $(\mathcal{D}\boldsymbol{x})(n)$
3: Set $\boldsymbol{z}_{s,0}(n) = \boldsymbol{x}\big((s-1)N + m\big)$ for all $s = \{1, .., S\}$.
4: **for** $k = 2, ..., K$ **do:**
5: Transmit $\boldsymbol{z}_{s,k-1}(n) = (\mathbf{L}\boldsymbol{z}_{s,k-2})(n)$ to all the neighbors, for all $s = \{1, 2, ..., S\}$.
6: Receive $\boldsymbol{z}_{s,k-1}(m)$ from all $m \in \mathcal{N}_n$, for all $s = \{1, 2, ..., S\}$.
7: **end for**
8: Compute $\boldsymbol{z}_{s,K}(n) = (\mathbf{L}\boldsymbol{z}_{s,K-1})(n)$, for all $s = \{1, ..., S\}$.
9: Compute and output $(\mathcal{D}\boldsymbol{x})(n) = \sum_{s=1}^{S} \sum_{k=0}^{K} \alpha_{ks} \boldsymbol{z}_{s,k}(n)$.

Such operators are particularly useful in the distributed implementation of signal processing tasks related to learning or regularization on graphs, such as denoising [4], semi-supervised learning [18], signal reconstruction [3], interpolation and reconstruction of bandlimited graph signals [19]. These types of applications typically require

the computation of quantities such as the forward application of the dictionary and its adjoint to be performed only by local exchange of information.

3 Distributed processing with quantization

We now study the effect of quantization in distributed signal processing with polynomial dictionaries by modeling the propagation of the quantization error in the different dictionary-based operators.

Given a graph signal $\boldsymbol{y}$, and the representation of the signal in a polynomial graph dictionary $\boldsymbol{\mathcal{D}}$, i.e., $\boldsymbol{y} = \boldsymbol{\mathcal{D}x}$, we study the computation of three basic operators, i.e., the forward operator $\boldsymbol{\mathcal{D}x}$, the adjoint operator $\boldsymbol{\mathcal{D}}^T\boldsymbol{y}$, and the operator $\boldsymbol{\mathcal{D}}^T\boldsymbol{\mathcal{D}x}$ in distributed settings when the sensors exchange quantized messages. Although our main focus is on graph dictionaries that are given directly in a polynomial form of the graph Laplacian operator, we note that the following results hold for every graph dictionary that can be approximated by a polynomial of any graph connectivity matrix (e.g., graph shift operator, adjacency matrix). In what follows, we assume that at every iteration, each sensor measurement is quantized with a uniform quantizer whose parameters are defined by the initial sensors states. In particular, we define a finite interval of size $\Delta S = S_0^{(\max)} - S_0^{(\min)}$. The parameters $S_0^{(\min)}$ and $S_0^{(\max)}$ represent the minimum and the maximum values of the interval, respectively, that are defined a priori. In the case of a q-bit uniform quantizer, the parameter $\Delta = \Delta S/2^q$ is the quantization step-size, which drives the error of the quantizer.

3.1 Distributed computation of $\mathcal{D}^T y$ with quantization

As we already saw in Section 2.4, the distributed computation of $\boldsymbol{\mathcal{D}}^T\boldsymbol{y}$ requires first the computation of the different powers of the Laplacian matrix, i.e., $\{\mathbf{L}^0\boldsymbol{y}, \mathbf{L}^1\boldsymbol{y}, \mathbf{L}^2\boldsymbol{y}, ..., \mathbf{L}^K\boldsymbol{y}\}$, in a distributed way. The latter can be done efficiently by successive multiplications of the matrix $\mathbf{L}$ with the signal $\boldsymbol{y}$ over K iterations, which as we will see next contains some noise that is accumulated over the iterations when the messages are quantized. Recall that the variable $\boldsymbol{z}_k$ in the computation of $\boldsymbol{\mathcal{D}}^T\boldsymbol{y}$ captures the sensors' values at the k^{th} iteration, with $\boldsymbol{z}_0 = \boldsymbol{y}$. Before the sensors exchange information, the value of this variable at sensor n and iteration k, i.e., $\boldsymbol{z}_k(n)$, is now quantized such that:

$$\tilde{\boldsymbol{z}}_k(n) = \boldsymbol{z}_k(n) + \boldsymbol{\epsilon}_k(n), \tag{5}$$

where $\boldsymbol{\epsilon}_k(n)$ is the quantization error in k, and $\tilde{\boldsymbol{z}}_k(n)$ is the quantized value that the sensor n sends to its neighbors. In particular, in the case of a q-bit uniform quantizer, the quantized values can be written as:

$$\tilde{\boldsymbol{z}}_k(n) = \left\lfloor \frac{\boldsymbol{z}_k(n) - \boldsymbol{z}_0^{(\min)}}{\Delta} \right\rfloor \cdot \Delta + \frac{\Delta}{2} + \boldsymbol{z}_0^{(\min)}.$$

Then, each node updates the local value of the variable $\boldsymbol{z}_k$ as a linear combination of its own quantized value and the quantized values received from its neighbors $\tilde{\boldsymbol{z}}_k(i)$ with $i \in \mathcal{N}_n$, based on the recursive update relationship in a vectorized form:

$$\boldsymbol{z}_{k+1} = \mathbf{L}(\boldsymbol{z}_k + \boldsymbol{\epsilon}_k). \tag{6}$$

By taking into consideration the quantization error from the previous iterations, Eq. (6) can be re-written as:

$$\boldsymbol{z}_{k+1} = \mathbf{L}^{k+1}\boldsymbol{z}_0 + \sum_{l=0}^{k} \mathbf{L}^{k+1-l}\boldsymbol{\epsilon}_l. \tag{7}$$

We observe that the quantization process involved in the transmission of the different powers of the Laplacian induces some quantization noise that is accumulated over the K iterations and is represented by the second term of Eq. (7).

We now compute $\widetilde{\boldsymbol{\mathcal{D}}_s^T}\boldsymbol{y}$, the quantized vector corresponding to $\boldsymbol{\mathcal{D}}_s^T\boldsymbol{y}$, by applying the polynomial coefficients on the values generated by the sequence $\{\boldsymbol{z}_0, \boldsymbol{z}_1, ..., \boldsymbol{z}_K\}$ given by Eq. (7). It reads as:

$$\begin{aligned}
\widetilde{\boldsymbol{\mathcal{D}}_s^T}\boldsymbol{y} &= \sum_{k=0}^{K} \alpha_{sk}\boldsymbol{z}_k = \sum_{k=0}^{K} \alpha_{sk}\mathbf{L}^k\boldsymbol{y} + \sum_{l=1}^{K}\left[\sum_{j=1}^{l-1} \alpha_{sl}\mathbf{L}^{l-j}\boldsymbol{\epsilon}_j\right] \\
&= \sum_{k=0}^{K} \alpha_{sk}\mathbf{L}^k\boldsymbol{y} + \sum_{l=0}^{K-1}\left[\sum_{j=1}^{K-l} \alpha_{s(l+j)}\mathbf{L}^j\right]\boldsymbol{\epsilon}_l \\
&= \boldsymbol{\mathcal{D}}_s^T\boldsymbol{y} + E\left(\boldsymbol{\mathcal{D}}_s^T\boldsymbol{y}\right),
\end{aligned} \tag{8}$$

where

$$E\left(\boldsymbol{\mathcal{D}}_s^T\boldsymbol{y}\right) = \sum_{l=0}^{K-1}\left[\sum_{j=1}^{K-l} \alpha_{s(l+j)}\mathbf{L}^j\right]\boldsymbol{\epsilon}_l,$$

is the overall accumulated quantization noise that occurs in the distributed computation of $\boldsymbol{\mathcal{D}}_s^T\boldsymbol{y}$. Finally, the global operation $\widetilde{\boldsymbol{\mathcal{D}}^T}\boldsymbol{y} = \left\{\widetilde{\boldsymbol{\mathcal{D}}_s^T}\boldsymbol{y}\right\}_{s=1}^{S}$ in the presence of quantization can be written as:

$$\widetilde{\boldsymbol{\mathcal{D}}^T}\boldsymbol{y} = \boldsymbol{\mathcal{D}}^T\boldsymbol{y} + E\left(\boldsymbol{\mathcal{D}}^T\boldsymbol{y}\right),$$

where $E\left(\boldsymbol{\mathcal{D}}^T\boldsymbol{y}\right) = \left\{E\left(\boldsymbol{\mathcal{D}}_s^T\boldsymbol{y}\right)\right\}_{s=1}^{S}$ is an error vector in $\mathbb{R}^{SN}$ that contains as entries the error obtained by applying the S different sets of polynomial coefficients to the accumulated quantization noise. The distributed algorithm for computing $\widetilde{\boldsymbol{\mathcal{D}}^T}\boldsymbol{y}$ is summarized in Algorithm 3.

3.2 Distributed computation of $\mathcal{D}x$ with quantization

We recall that $\boldsymbol{\mathcal{D}x} = \sum_{s=1}^{S} \boldsymbol{\mathcal{D}}_s\boldsymbol{x}_s$, where $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_S) \in \mathbb{R}^{SN}$ is a vector containing as entries the sparse codes $\boldsymbol{x}_s$ corresponding to subdictionary $\boldsymbol{\mathcal{D}}_s$. Each of the components in the summation is computed

Algorithm 3 Distributed computation of $\mathcal{D}^T\boldsymbol{y}$ with quantization

1: **Input at node n:** $\boldsymbol{y}(n), \mathbf{L}_{n,:}, \alpha = [\alpha_1; ...; \alpha_S]$, quantization stepsize Δ
2: **Output at node n:** $\left(\widetilde{\mathcal{D}^T\boldsymbol{y}}\right)\big((s-1)N+n\big)$ for all $s = \{1, ..., S\}$
3: Quantize and transmit $\tilde{\boldsymbol{y}}(n) = \boldsymbol{y}(n) + \boldsymbol{\epsilon}_0(n)$ to all $m \in \mathcal{N}_n$.
4: Receive $\tilde{\boldsymbol{y}}(m)$ from neighbors in $\mathcal{N}_n$.
5: Set $\boldsymbol{z}_0(n) = \boldsymbol{y}(n), \quad \tilde{\boldsymbol{z}}_0(n) = \tilde{\boldsymbol{y}}(n)$.
6: **for** $k = 2, ..., K$ **do:**
7: Compute $\boldsymbol{z}_{k-1}(n) = \left(\mathbf{L}^T\tilde{\boldsymbol{z}}_{k-2}\right)(n)$.
8: Quantize and transmit $\tilde{\boldsymbol{z}}_{k-1}(n) = \boldsymbol{z}_{k-1}(n) + \boldsymbol{\epsilon}_{k-1}(n)$ to all the neighbors.
9: Receive $\tilde{\boldsymbol{z}}_{k-1}(m)$ from all the neighbors $m \in \mathcal{N}_n$.
10: **end for**
11: Compute $\boldsymbol{z}_K(n) = (\mathbf{L}\tilde{\boldsymbol{z}}_{k-1})(n)$.
12: **for** $s = \{1, ..., S\}$ **do**
13: Compute $\widetilde{(\mathcal{D}^T\boldsymbol{y})}\big((s-1)N+n\big) = \sum_{k=0}^{K}\alpha_{ks}\boldsymbol{z}_k(n)$.
14: **end for**

by sending iteratively the powers of the Laplacian as described in Section 2.4. The quantization effect through the iterative process is significant. To elaborate on that, for each of the subdictionaries, we define a new variable $\boldsymbol{z}_{s,0} = \boldsymbol{x}_s$. The quantized value of this variable at sensor n and iteration k that sensor n sends to its neighbors reads:

$$\tilde{\boldsymbol{z}}_{s,k}(n) = \boldsymbol{z}_{s,k}(n) + \boldsymbol{\zeta}_{s,k}(n), \tag{9}$$

where $\boldsymbol{\zeta}_{s,k}(n)$ is the quantization error, and $\boldsymbol{z}_{s,k}(n)$ is the value of the sensor before quantization that is computed as:

$$\boldsymbol{z}_{s,k}(n) = \left(\mathbf{L}\tilde{\boldsymbol{z}}_{s,k-1}\right)(n).$$

By taking into consideration the quantization error components from the previous iterations, the values of the sensors at iteration $k+1$ are defined as:

$$\boldsymbol{z}_{s,k+1} = \mathbf{L}^{k+1}\boldsymbol{z}_{s,0} + \sum_{l=0}^{k}\mathbf{L}^{k+1-l}\boldsymbol{\zeta}_{s,l}. \tag{10}$$

Using Eq. (10), we obtain the operator $\mathcal{D}\boldsymbol{x}$ with quantization:

$$\begin{aligned}\widetilde{\mathcal{D}\boldsymbol{x}} &= \sum_{s=1}^{S}\mathcal{D}_s\boldsymbol{x}_s = \sum_{s=1}^{S}\sum_{k=0}^{K}\alpha_{sk}\boldsymbol{z}_{s,k}\\ &= \sum_{s=1}^{S}\left\{\sum_{k=0}^{K}\alpha_{sk}\mathbf{L}^k\boldsymbol{x}_s + \sum_{l=0}^{K-1}\left[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right]\boldsymbol{\zeta}_{s,l}\right\}\\ &= \sum_{s=1}^{S}\left[\mathcal{D}_s\boldsymbol{x}_s + E\left(\mathcal{D}_s\boldsymbol{x}_s\right)\right],\end{aligned} \tag{11}$$

where the accumulated quantization noise corresponding to subdictionary $\mathcal{D}_s$ is defined as:

$$E\left(\mathcal{D}_s\boldsymbol{x}_s\right) = \sum_{l=0}^{K-1}\left[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right]\boldsymbol{\zeta}_{s,l}.$$

The main steps of the distributed algorithm for the computation of $\mathcal{D}\boldsymbol{x}$ with quantized messages are shown in Algorithm 4.

Algorithm 4 Distributed computation of $\mathcal{D}\boldsymbol{x}$ with quantization

1: **Input at node n:** $\boldsymbol{x}\big((s-1)N+n\big)$ for all $s = \{1, ..., S\}, \mathbf{L}_{n,:}, \alpha = [\alpha_1; ...; \alpha_S]$, quantization stepsize Δ
2: **Output at node n:** $(\widetilde{\mathcal{D}\boldsymbol{x}})(n)$
3: Quantize and transmit $\tilde{\boldsymbol{x}}\big((s-1)N+n\big) = \boldsymbol{x}\big((s-1)N+n\big) + \boldsymbol{\zeta}_{s,l}(n)$, for $s = \{1, ..., S\}$, to all $m \in \mathcal{N}_n$.
4: Receive $\tilde{\boldsymbol{x}}\big((s-1)N+n\big)$, for all $s = \{1, ..., S\}$, from $m \in \mathcal{N}_n$.
5: Set $\boldsymbol{z}_{s,0}(n) = \tilde{\boldsymbol{x}}\big((s-1)N+n\big)$, for all $s = \{1, ..., S\}$.
6: **for** $k = 2, ..., K$ **do:**
7: Compute $\boldsymbol{z}_{s,k-1}(n) = (\mathbf{L}^T\tilde{\boldsymbol{z}}_{s,k-2})(n)$, for all $s = \{1, 2, ..., S\}$.
8: Quantize and transmit $\tilde{\boldsymbol{z}}_{s,k-1}(n) = \boldsymbol{z}_{s,k-1}(n) + \boldsymbol{\zeta}_{s,k-1}(n)$ to all the neighbors, for all $s = \{1, 2, ..., S\}$.
9: Receive $\tilde{\boldsymbol{z}}_{s,k-1}(m)$ from all $m \in \mathcal{N}_n$, for all $s = \{1, 2, ..., S\}$.
10: **end for**
11: Compute $\boldsymbol{z}_{s,K}(n) = (\mathbf{L}\tilde{\boldsymbol{z}}_{s,K-1})(n)$, for all $s = \{1, ..., S\}$.
12: Compute and output $(\widetilde{\mathcal{D}\boldsymbol{x}})(n) = \sum_{s=1}^{S}\sum_{k=0}^{K}\alpha_{ks}\boldsymbol{z}_{s,k}(n)$.

3.3 Distributed computation of $\mathcal{D}^T\mathcal{D}\boldsymbol{x}$ with quantization

Finally, we illustrate the distributed computation of $\mathcal{D}^T\mathcal{D}\boldsymbol{x}$ when sensor messages are quantized. It follows the same reasoning as the one of the previous two operators. In particular, we assume that the operator $\mathcal{D}\boldsymbol{x}$ has already been sent and the corresponding entries of $\widetilde{\mathcal{D}\boldsymbol{x}}$ are known to the sensors. Thus, the new variable in this case

is defined as $\boldsymbol{z}_0 = \widetilde{\mathcal{D}\boldsymbol{x}}$. The sensors' values at iteration $k+1$ are:

$$\boldsymbol{z}_{k+1} = \mathbf{L}^{k+1}\boldsymbol{z}_0 + \sum_{l=0}^{k} \mathbf{L}^{k+1-l}\boldsymbol{\xi}_l, \tag{12}$$

where $\boldsymbol{\xi}_l$ is the quantization error vector $\boldsymbol{\xi}_l = (\boldsymbol{\xi}_l(1), \boldsymbol{\xi}_l(2), ..., \boldsymbol{\xi}_l(N))$ that occurs after transmitting $\tilde{\boldsymbol{z}}_l$. By combining Eqs. (8), (11), and (12), for each subdictionary $\mathcal{D}_s$, we can compute $\mathcal{D}_s^T\mathcal{D}x$ with quantization as follows:

$$\begin{aligned}
\widetilde{\mathcal{D}_s^T\mathcal{D}\boldsymbol{x}} &= \sum_{k=0}^{K} \alpha_{sk}\mathbf{L}^k\boldsymbol{z}_k \\
&= \sum_{k=0}^{K} \alpha_{sk}\mathbf{L}^k\boldsymbol{z}_0 + \sum_{l=1}^{K}\left[\sum_{j=1}^{l-1}\alpha_{sl}\mathbf{L}^{l-j}\boldsymbol{\xi}_j\right] \\
&= \sum_{k=0}^{K} \alpha_{sk}\mathbf{L}^k\widetilde{\mathcal{D}\boldsymbol{x}} + \sum_{l=0}^{K-1}\left[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right]\boldsymbol{\xi}_l \\
&= \sum_{k=0}^{K} \alpha_{sk}\mathbf{L}^k \sum_{s'=1}^{S}\left\{\sum_{k'=0}^{K}\alpha_{s'k'}\mathbf{L}^{k'}x_{s'}\right. \\
&\quad \left. + \sum_{l'=0}^{K-1}\left[\sum_{j'=1}^{K-l'}\alpha_{s'(l'+j')}\mathbf{L}^{j'}\right]\zeta_{s',l'}\right\} \\
&\quad + \sum_{l=0}^{K-1}\left[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right]\boldsymbol{\xi}_l \\
&= \mathcal{D}_s^T\mathcal{D}\boldsymbol{x} + \sum_{k=0}^{K}\alpha_{sk}\mathbf{L}^k E(\mathcal{D}\boldsymbol{x}) + E\left(\mathcal{D}_s^T\mathcal{D}\boldsymbol{x}\right),
\end{aligned} \tag{13}$$

where we have set

$$E(\mathcal{D}x) = \sum_{s'=1}^{S}\sum_{l'=0}^{K-1}\left[\sum_{j'=1}^{K-l'}\alpha_{s'(l'+j')}\mathbf{L}^{j'}\right]\zeta_{s',l'},$$

$$E\left(\mathcal{D}_s^T\mathcal{D}x\right) = \sum_{l=0}^{K-1}\left[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right]\boldsymbol{\xi}_l.$$

Finally, the operation $\widetilde{\mathcal{D}^T\mathcal{D}\boldsymbol{x}} = \left\{\widetilde{\mathcal{D}_1^T\mathcal{D}\boldsymbol{x}}\right\}_{s=1}^{S}$ can be written as:

$$\widetilde{\mathcal{D}^T\mathcal{D}\boldsymbol{x}} = \mathcal{D}^T\mathcal{D}\boldsymbol{x} + \mathcal{D}^T E(\mathcal{D}\boldsymbol{x}) + E\left(\mathcal{D}^T\mathcal{D}\boldsymbol{x}\right),$$

where $E\left(\mathcal{D}^T\mathcal{D}\boldsymbol{x}\right) = \left\{E\left(\mathcal{D}_s^T\mathcal{D}\boldsymbol{x}\right)\right\}_{s=1}^{S}$. Again, we observe that there is an error accumulated from the computation of both steps $\mathcal{D}\boldsymbol{x}$ and $\mathcal{D}^T\mathcal{D}\boldsymbol{x}$ that depends on the quantization noise and the structure of the dictionary through the coefficients $\{\alpha_{sk}\}_{s=1,k=0}^{S,K}$. The main steps of the algorithm are shown in Algorithm 5.

Algorithm 5 Distributed computation of $\mathcal{D}^T\mathcal{D}\boldsymbol{x}$ with quantization

1: **Input at node *n*:** $(\widetilde{\mathcal{D}\boldsymbol{x}})(n), \mathbf{L}_{n,:}, \alpha = [\alpha_1; ...; \alpha_S]$, quantization stepsize Δ
2: **Output at node *n*:** $(\widetilde{\mathcal{D}^T\mathcal{D}\boldsymbol{x}})\big((s-1)N+n\big)$ for all $s = \{1, .., S\}$
3: Set $\boldsymbol{z}_0(n) = \widetilde{\mathcal{D}\boldsymbol{x}}(n)$.
4: Quantize and transmit $\tilde{\boldsymbol{z}}_0(n) = \boldsymbol{z}_0(n) + \boldsymbol{\xi}_0(n)$ to all $m \in \mathcal{N}_n$.
5: Receive $\tilde{\boldsymbol{z}}_0(m)$ from neighbors in $\mathcal{N}_n$.
6: **for** $k = 2, ..., K$ **do:**
7: Compute $\boldsymbol{z}_{k-1}(n) = \left(\mathbf{L}^T\tilde{\boldsymbol{z}}_{k-2}\right)(n)$.
8: Quantize and transmit $\tilde{\boldsymbol{z}}_{k-1}(n) = \boldsymbol{z}_{k-1}(n) + \boldsymbol{\xi}_{k-1}(n)$ to all the neighbors.
9: Receive $\tilde{\boldsymbol{z}}_{k-1}(m)$ from all the neighbors $m \in \mathcal{N}_n$.
10: **end for**
11: Compute $\boldsymbol{z}_K(n) = \left(\mathbf{L}\tilde{\boldsymbol{z}}_{k-1}\right)(n)$.
12: **for** $s = \{1, .., S\}$ **do**
13: Compute $(\widetilde{\mathcal{D}^T\mathcal{D}\boldsymbol{x}})\big((s-1)N+n\big) = \sum_{k=0}^{K}\alpha_{ks}\boldsymbol{z}_k(n)$.
14: **end for**

Finally, we note that while the above analysis of the quantization noise is based on the normalized graph Laplacian matrix, it holds for any other graph matrix that captures the communication pattern of the network, such as the combinatorial Laplacian, and the adjacency matrix. In the next section, we give an illustrative example of the use of such operators in the distributed sparse representation of graph signals.

4 Distributed sparse graph signal regularization with quantization

In the following, we use the above operators for the distributed computation of a sparse representation of a signal $\boldsymbol{y}$ with respect to a dictionary $\mathcal{D}$, under communication constraints. The sparse representation in a dictionary $\mathcal{D}$ can be found by solving a LASSO minimization problem [20] as shown next.

4.1 Illustrative application of distributed graph signal processing

We consider the distributed processing scenario where each node n of the graph computes the sparse decomposition in a polynomial dictionary by solving a sparse regularization problem of the form:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} ||\boldsymbol{y} - \mathcal{D}\boldsymbol{x}||_2^2 + \kappa\|\boldsymbol{x}\|_1, \tag{14}$$

where κ is a parameter that controls the sparsity level. The above problem is also called LASSO [21] or basis pursuit denoising [22] and is used to perform denoising with

sparse prior. We thus start with the underlying assumption that the signal $\boldsymbol{y}$ is sparse in a polynomial graph dictionary, whose coefficients are known to all the sensors. Moreover, node n knows its own component of a signal $\boldsymbol{y} \in \mathbb{R}^N$ (i.e., $\boldsymbol{y}(n)$) and the n^{th} row of the corresponding Laplacian matrix $\mathbf{L}_{n,:}$. The above problem can be solved by an iterative soft thresholding algorithm (ISTA) [23], in which the update of the estimated coefficients is given by:

$$\boldsymbol{x}^{(t)} = \mathcal{S}_{\kappa\tau}\left(\boldsymbol{x}^{(t-1)} + 2\tau\boldsymbol{\mathcal{D}}^T\left(\boldsymbol{y} - \boldsymbol{\mathcal{D}}\boldsymbol{x}^{(t-1)}\right)\right), \quad t = 1, 2, \ldots \tag{15}$$

where τ is the gradient stepsize, and $\mathcal{S}_{\kappa\tau}$ is the soft thresholding operator:

$$\mathcal{S}_{\kappa\tau}(z) = \begin{cases} 0, & \text{if } |z| \leq \kappa\tau \\ z - \text{sgn}(z)\kappa\tau, & \text{otherwise,} \end{cases}$$

which corresponds to the proximal operator of the $\kappa\|\boldsymbol{x}\|_1$ function. Thus, the whole algorithm is a particular instance of the general family of proximal gradient methods [24]. By combining the distributed computation of the operations $\boldsymbol{\mathcal{D}}^T\boldsymbol{y}$, $\boldsymbol{\mathcal{D}}^T\boldsymbol{\mathcal{D}}\boldsymbol{x}$, and $\boldsymbol{\mathcal{D}}\boldsymbol{x}$, as described in the previous subsection, each iteration of ISTA can be solved in a distributed way [4]. In particular, in the first iteration, each node n must compute $(\boldsymbol{\mathcal{D}}_s\boldsymbol{y})(n)$ for all the subdictionaries, via Algorithm 1. In each iteration $(t+1)$, it must compute first $(\boldsymbol{\mathcal{D}}\boldsymbol{x}^{(t)})(n)$, and sequentially apply $\boldsymbol{\mathcal{D}}^T\boldsymbol{\mathcal{D}}\boldsymbol{x}^{(t)}$ via Algorithms 2 and 1, respectively. The solution of (14) is found after a stopping criteria is satisfied (e.g., a fixed number of iterations is executed). The estimate of the signal at each node is then given by computing $\hat{\boldsymbol{y}} = \boldsymbol{\mathcal{D}}\boldsymbol{x}^*$ via Algorithm 2.

The computational complexity for each iteration of ISTA depends mainly on the matrix operations that are involved in the process, i.e., the dictionary forward and adjoint operators. We briefly discuss these steps here. We recall that $\boldsymbol{\mathcal{D}}^T\boldsymbol{y} = \sum_{s=1}^{S}\sum_{k=0}^{K}\alpha_{sk}\mathbf{L}^k\boldsymbol{y}$. The computational cost of the iterative sparse matrix-vector multiplication required to compute $\left\{\mathbf{L}^k\boldsymbol{y}\right\}_{k=0,2,\ldots,K}$ is $O(K|\mathcal{E}|)$, where $|\mathcal{E}|$ is the cardinality of the edge set of the graph. Therefore, the total computational cost to compute $\boldsymbol{\mathcal{D}}^T\boldsymbol{y}$ is $O(K|\mathcal{E}| + NSK)$. We further note that, by following a procedure similar to the one in [17], the term $\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{D}}^T\boldsymbol{y}$ can also be computed in a fast way by exploiting the fact that $\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{D}}^T\boldsymbol{y} = \sum_{s=1}^{S}\widehat{g_s}^2(\mathbf{L})\boldsymbol{y}$. This leads to a polynomial of degree $K' = 2K$ that can be efficiently computed. Similar reasoning can be followed to compute the complexity of the other operations involved in the process. Since S, K are relatively small, the complexity of ISTA at each iteration is mainly dominated by the size of the graph, and the number of edges.

4.2 Performance of ISTA under quantization constraints

The first step of ISTA requires the computation of the gradient of the fitting term of Eq. (14), i.e., $\|\boldsymbol{y} - \boldsymbol{\mathcal{D}}\boldsymbol{x}\|^2$, which implies the computation of the operations $\boldsymbol{\mathcal{D}}^T\boldsymbol{y}$, $\boldsymbol{\mathcal{D}}^T\boldsymbol{\mathcal{D}}\boldsymbol{x}$ in each iteration of the algorithm. When the messages exchanged by the sensors are quantized, the quantization noise induced by each of these operations introduces an error in the gradient, such that:

$$\tilde{\boldsymbol{x}}^{(t)} = \mathcal{S}_{\kappa\tau}\left(\boldsymbol{x}^{(t-1)} + 2\tau\left(\boldsymbol{\mathcal{D}}^T\boldsymbol{y} - \boldsymbol{\mathcal{D}}^T\boldsymbol{\mathcal{D}}\boldsymbol{x}^{(t-1)} + \boldsymbol{e}^{(t-1)}\right)\right), \tag{16}$$

where $\boldsymbol{e}^{(t-1)}$ is the total gradient error, which according to Eqs. (8), (11), and (13) can be expressed as:

$$\boldsymbol{e}^{(t-1)} = E^{(t-1)}\left(\boldsymbol{\mathcal{D}}^T\boldsymbol{y}\right) - \boldsymbol{\mathcal{D}}^T E^{(t-1)}(\boldsymbol{\mathcal{D}}\boldsymbol{x}) - E^{(t-1)}\left(\boldsymbol{\mathcal{D}}^T\boldsymbol{\mathcal{D}}\boldsymbol{x}\right).$$

The convergence of the sparse graph signal representation by the ISTA algorithm then depends on the sequence of errors over the iterations. It can be characterized by the following result from [25] that applies to the general family of proximal gradient methods such as ISTA.

Theorem 1 ([25]) *Let f be a differentiable with Lipschitz continuous gradient function on some compact set with Lipschitz constant L, g a lower semi-continuous and convex function, and $\{\tau_t\}$ a sequence of gradient stepsizes that satisfy the conditions:*

$$0 < \beta \leq \tau_t \leq \min(1, 2/L - \beta), \text{with } 0 < \beta < \frac{1}{L}.$$

Then, the sequence generated by the iterates

$$\boldsymbol{x}^{(t)} = prox_{\tau_{t-1}g}\left(\boldsymbol{x}^{(t-1)} - \tau_{t-1}\nabla f\left(\boldsymbol{x}^{(t-1)}\right) + \tau_{t-1}\boldsymbol{e}^{(t-1)}\right) \tag{17}$$

converges to a stationary point $\mathbf{x}^*$ if, given a fixed $\bar{\tau}$, the following condition on the gradient error holds:

$$\forall\tau \in (0, \bar{\tau}], \quad \tau\|\boldsymbol{e}\| \leq \bar{\epsilon}, \quad \text{for some } \bar{\epsilon} \geq 0.$$

The above result indicates that the proximal gradient method converges to an approximate stationary point if the norm of the gradient error is uniformly bounded. Furthermore, if the number of perturbed gradient computations is finite, or if the gradient error norm converges toward 0, then the sequence limit point is the exact solution of the initial problem. Therefore, we have to make sure that the error in the gradient is bounded so that the distributed sparse graph signal representation algorithm converges.

Assume that we use a uniform quantizer in our distributed signal processing algorithm which is widely used and simple to implement. Let us further assume that our quantizer has a quantization stepsize Δ for the magnitude and one bit for the sign. With such a quantizer, an upper

bound on the norm of the error is given by the following lemma.

Lemma 1 *Let* $\mathbf{e^{(t-1)}}$ *be the error vector due to quantization in the computation of the gradient at iteration* $t-1$ *as defined in Eq. (16) and* Δ *the quantization stepsize of a uniform quantizer. Then, the quantization error is bounded as:*

$$\left\|\boldsymbol{e}^{(t-1)}\right\| \leq \sqrt{N}\frac{\Delta}{2}\sum_{s=1}^{S}\left\{2\sum_{l=0}^{K-1}\left\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right\| + c\sum_{s'=1}^{S}\sum_{l'=0}^{K-1}\left\|\sum_{j'=1}^{K-l'}\alpha_{s'(l'+j')}\mathbf{L}^{j'}\right\|\right\}. \tag{18}$$

The proof of Lemma 1 is provided in the "Appendix" section.

The above inequality shows that the error at each iteration of the gradient is upper bounded by the quantization stepsize, multiplied by a matrix polynomial of the graph Laplacian $\mathbf{L}$. The quantization errors depend on the number of bits, i.e., the rate constraints on data exchanged in the sensor network. For a uniform quantizer, the magnitude of the quantization error is upper bounded by the quantization stepsize. In particular, if the norm $\left\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right\|$ is bounded by a constant $\eta > 0$, i.e., $\left\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right\| \leq \eta$ for $l \in \{1,...,K-1\}$, $s \in \{1,...,S\}$, Eq. (18) becomes:

$$\left\|\boldsymbol{e}^{(t-1)}\right\| \leq \sqrt{N}\frac{\Delta}{2}SK(2+c)\eta. \tag{19}$$

Thus, the error of the gradient at each iteration is bounded, which implies that ISTA converges to a stationary point of the iteration (17) according to Theorem 1. When $\Delta \to 0$, i.e., the bit rate tends to infinity, and the quantization noise tends to zero, $\|\boldsymbol{e}^{(t-1)}\| \to 0$, independently of η. However, when the number of bits is limited and fixed, the quantization noise depends on the characteristics of the dictionary. When the norm of the polynomial of the Laplacian matrix goes to zero (i.e., $\eta \to 0$), the error also tends to 0 (i.e., $\left\|\boldsymbol{e}^{(t-1)}\right\| \to 0$). Finally, the upper bound on the error due to quantization in Eq. (19) indicates that the higher the degree K of the polynomial, the more the error tends to be accumulated over the iterations. This is quite intuitive as higher polynomial degree requires more information to be exchanged between the sensors, at the cost of more propagation of the quantization noise. A large value of K at the same time guarantees that the polynomial functions can better approximate the underlying spectral kernels and thus the graph signals. It indicates that there is a trade-off in the design of the dictionary, between the representation performance of the polynomial dictionary and the propagation of the quantization noise.

5 Polynomial dictionary learning with quantization

We use the study of the previous section to include the quantization parameter in the dictionary design, and we introduce an algorithm to learn polynomial dictionaries that are robust to quantization noise. Our approach consists in controlling the norm of the total error in each step of the gradient computation when solving ISTA-based algorithms in a distributed way. When the quantization step size and the graph are given, the total error due to quantized communication can be controlled by choosing the proper values for the polynomial coefficients $\{\alpha_{sk}\}_{s=1,k=0}^{S,K}$ such that the gradient error stays bounded. From Eq. (18), the polynomial coefficients need to be computed in such a way that the spectral norm $\left\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right\|$, for $l \in \{1,...,K-1\}$, is bounded for a fixed Δ. We recall that the spectral norm is defined as $\left\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right\| = \lambda_{\max}\left(\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right)$. Since the matrix $\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j$ is symmetric, the spectral norm is simply its largest eigenvalue. Therefore, constraining the spectral norm becomes equivalent to constraining the eigenvalues of the corresponding matrix.

5.1 Dictionary learning algorithm

Based on the above analysis, we propose here to control the maximum eigenvalue of the matrix $\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j$ for constructing dictionaries that are robust to the quantization noise. Namely, we choose the polynomial coefficients such that the spectral norm of $\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j$ is bounded and small. We design the learning algorithm to have explicit control on the propagation of the quantization error. In details, given a set of training signals $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_M] \in \mathbb{R}^{N\times M}$, all living on the weighted graph $\mathcal{G}$, our objective is to learn a polynomial graph dictionary $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{N\times NS}$, which can efficiently represent all of the signals in $\boldsymbol{y}$ as linear combinations of only a few of its atoms and at the same time be robust to the quantization error, when applied to distributed setting with rate constraints. Since $\boldsymbol{\mathcal{D}}$ has the form (3), our problem is equivalent to learning the parameters $\{\alpha_{sk}\}_{s=1,2,...,S;\ k=1,2,...,K}$ that characterize the set of generating kernels, $\left\{\widehat{g_s}(\cdot)\right\}_{s=1,2,...,S}$. We denote these parameters in vector form as $\alpha = [\alpha_1; ...; \alpha_S]$, where α_s is a column vector with $(K+1)$ entries. In order to take into account the effect of the quantization noise, we impose an additional constraint which bounds the eigenvalues

of the matrix $\sum_{j=1}^{K-l} \alpha_{s(l+j)} \mathbf{L}^j$ for $l \in \{1, ..., K-1\}$ and $s \in \{1, ..., S\}$.

Formally, the dictionary learning problem can be cast as the following optimization problem:

$$\begin{aligned} \underset{\alpha \in \mathbb{R}^{(K+1)S},\, \boldsymbol{X} \in \mathbb{R}^{SN \times M}}{\text{argmin}} \quad & \left\{ ||\boldsymbol{Y} - \boldsymbol{\mathcal{D}}\boldsymbol{X}||_F^2 + \mu \|\alpha\|_2^2 \right\} \\ \text{subject to} \quad & \|\boldsymbol{x_m}\|_0 \leq T_0, \quad \forall m \in \{1, ..., M\}, \\ & \boldsymbol{\mathcal{D}}_s = \sum_{k=0}^{K} \alpha_{sk} \mathbf{L}^k, \forall s \in \{1, 2, ..., S\} \\ & 0I \preceq \boldsymbol{\mathcal{D}}_s \preceq cI, \quad \forall s \in \{1, 2, ..., S\} \\ & (c - \epsilon_1)I \preceq \sum_{s=1}^{S} \boldsymbol{\mathcal{D}}_s \preceq (c + \epsilon_2)I, \\ & -\eta I \preceq \sum_{j=1}^{K-l} \alpha_{s(l+j)} \mathbf{L}^j \preceq \eta I, \\ & \forall l \in \{1, ..., K-1\}, \end{aligned} \tag{20}$$

where $\boldsymbol{\mathcal{D}} = [\boldsymbol{\mathcal{D}}_1, \boldsymbol{\mathcal{D}}_2, \ldots, \boldsymbol{\mathcal{D}}_S]$, $\boldsymbol{x_m}$ corresponds to column m of the coefficient matrix $\boldsymbol{X}$, T_0 is the sparsity level of the coefficients of each signal, and $\mathbf{I}$ is the identity matrix. The above dictionary learning formulation is inspired by [3], including some additional constraints that take into account the quantization noise. The spectral constraints provide some control over the spectral representation of the atoms and the stability of signal reconstruction with the learned dictionary as discussed in [3]. In particular, they guarantee that the learned kernels are positive, and the obtained dictionary is a frame. The optimization problem (20) is not convex, but it can be approximately solved in a computationally efficient manner by alternating between the sparse coding and dictionary update steps. In the first step, we fix the parameters α (and accordingly fix the dictionary $\boldsymbol{\mathcal{D}}$) and solve the sparse coding step using orthogonal matching pursuit (OMP) [26]. In the second step, we fix the coefficients $\boldsymbol{X}$ and update the dictionary by finding the vector of parameters, α, that solves the polynomial coefficient update step using interior point methods [27].

We notice that the parameter η is a design parameter that intuitively should decrease when the quantization stepsize increases (See Eq. (19) for a fixed accumulated quantization error $\left\|\boldsymbol{e}^{(t-1)}\right\|$). In particular, a large quantization stepsize implies that the accumulated quantization error tends to be large. In that case, a small value of η penalizes the propagation of the quantization noise by learning a dictionary that is more robust in distributed settings, at the cost of a reduced flexibility in the search space of the polynomial coefficients. The latter implies a loss in the accurate recovery of the underlying spectral kernels that generate the true dictionary atoms. A large value of η gives more flexibility for a solution of (20) to learn a set of polynomial coefficients that are good for approximating the kernels in ideal communication settings, without restricting the accumulated quantization noise however. As a result, if the quantization stepsize is small (i.e., the quantization is fine enough), η can be chosen relatively big, so that it does not affect the solution of the optimization problem (20).

As mentioned in the introduction, the problem of quantization-aware graph signal representations has been recently studied in [16], from the viewpoint of designing robust FIR filters. In particular, a zero-mean Gaussian i.i.d. initial signal is filtered by a graph filter which is built on a graph shift operator, and it results on an output graph signal. Contrary to our work, the desired frequency response of the filter is known, and the main objective is to find an approximation of that response with a robust graph filter that (i) approximates sufficiently well the desired response and (ii) reduces the amount of quantization noise at the output graph signal. In our formulation of (20), our signals are generated from a sparse signal model, whose representation matrix, i.e., dictionary, consists of a set of graph filters, whose frequency response is not known. Thus, the goal is to find a set of desired filters that can approximate a set of graph signals, given some predefined constrains that penalize the propagation of the quantization noise. Thus, the focus is more on the representation and processing of the signal rather than on the approximation of a desired frequency response of a filter.

Finally, we note that all the above developments are based on the ISTA iterations. Since ISTA is in general slow, it implies more data exchange between sensors that goes in contradiction with the initial aim of transmitting fewer data. However, the results of Section 3 could be used to compute the accumulated approximation error for any other algorithm that contains these simple matrix-vector multiplications. ISTA is an example of such algorithm that is widely used. A similar philosophy could be used for fast iterative shrinkage-thresholding algorithm (FISTA) [23] for example. It is straighforward to derive the quantization error bounds in this case. In other words, depending on the target sparsity algorithm, we can derive the required constraints that could be applied in the polynomial dictionary learning phase. Of course, the exact constraints might end up being different than those derived in the paper for the specific case of ISTA. Still, constraining the polynomial coefficients based on the accumulated quantization error is something that will improve the performance at low bit rate. Our objective in this paper is to propose a dictionary learning algorithm that specifically include constraints imposed by quantization in distributed processing on graphs and to illustrate one specific case, i.e., ISTA.

5.2 Analysis of the learned dictionary

To quantify the effect of the quantization constraints on the polynomial coefficients, we derive a few representative bounds. In particular, using the fact that the constraints affect the spectrum of the matrix, for $l = K-1$, we obtain that:

$$\begin{aligned} &-\eta I \preceq \sum_{j=1}^{1} \alpha_{s(K-1+j)} \mathbf{L}^j \preceq \eta I, \\ \implies &-\eta \leq \quad \alpha_{sK}\lambda \quad \leq \eta, \quad \forall \lambda \in \Lambda \\ \overset{\lambda=\lambda_{\max}}{\implies} &-\frac{\eta}{\lambda_{\max}} \leq \quad \alpha_{sK} \quad \leq \frac{\eta}{\lambda_{\max}}. \end{aligned} \tag{21}$$

We note that since $0 \leq \lambda \leq \lambda_{\max} = 2$, the magnitude of the coefficient α_{sK} is defined by the largest eigenvalue $\lambda_{\max}$.

Following a similar reasoning, for $l = K-2$, we obtain that:

$$\begin{aligned} &-\eta I \preceq \sum_{j=1}^{2} \alpha_{s(K-2+j)} \mathbf{L}^j \preceq \eta I, \\ \implies &-\eta \leq \quad \alpha_{s(K-1)}\lambda + \alpha_{sK}\lambda^2 \quad \leq \eta, \quad \forall \lambda \in \Lambda \end{aligned} \tag{22}$$

$$\begin{aligned} \overset{\lambda=\lambda_{\max}}{\implies} &-\frac{\eta}{\lambda_{\max}} \leq \quad \alpha_{s(K-1)} + \alpha_{sK}\lambda_{\max} \quad \leq \frac{\eta}{\lambda_{\max}} \\ \overset{(21)}{\implies} &-\frac{\eta}{\lambda_{\max}} - \eta \leq \quad \alpha_{s(K-1)} \quad \leq \frac{\eta}{\lambda_{\max}} + \eta \\ \implies &-\eta\left(\frac{1+\lambda_{\max}}{\lambda_{\max}}\right) \leq \quad \alpha_{s(K-1)} \quad \leq \eta\left(\frac{1+\lambda_{\max}}{\lambda_{\max}}\right). \end{aligned} \tag{23}$$

Similarly, for $l = K-3$, we obtain that:

$$\begin{aligned} &-\eta I \preceq \sum_{j=1}^{3} \alpha_{s(K-3+j)} \mathbf{L}^j \preceq \eta I, \\ \implies &-\eta \leq \quad \alpha_{s(K-2)}\lambda + \alpha_{s(K-1)}\lambda^2 + \alpha_{sK}\lambda^3 \quad \leq \eta, \quad \forall \lambda \in \Lambda \\ \overset{\lambda=\lambda_{\max}}{\implies} &-\frac{\eta}{\lambda_{\max}} \leq \quad \alpha_{s(K-2)} + \alpha_{s(K-1)}\lambda_{\max} + \alpha_{sK}\lambda_{\max}^2 \quad \leq \frac{\eta}{\lambda_{\max}} \\ \overset{(22)}{\implies} &-\frac{\eta}{\lambda_{\max}} - \eta \leq \quad \alpha_{s(K-2)} \quad \leq \frac{\eta}{\lambda_{\max}} + \eta \\ \implies &-\eta\left(\frac{1+\lambda_{\max}}{\lambda_{\max}}\right) \leq \quad \alpha_{s(K-2)} \quad \leq \eta\left(\frac{1+\lambda_{\max}}{\lambda_{\max}}\right). \end{aligned} \tag{24}$$

Finally, using the above developments, we can recursively bound $\alpha_{s(K-3)}$. Following similar reasoning, it can be easily derived that:

$$-\eta\left(\frac{1+\lambda_{\max}}{\lambda_{\max}}\right) \leq \quad \alpha_{s(K-j)} \quad \leq \eta\left(\frac{1+\lambda_{\max}}{\lambda_{\max}}\right), \ \forall j \in 1, 2, ..., K. \tag{25}$$

We note that the above bounds are quite conservative as they are based on the largest eigenvalue of the graph Laplacian. These types of inequalities however show that by adding the quantization constraints, we restrict the magnitude of the coefficients. The effect of this trade-off is studied numerically in the next section.

6 Results and discussion

We first study the performance of our dictionary learning algorithm for the distributed approximation of synthetic signals. Then, we study the application of the proposed framework in the denoising of real world signals.

6.1 Synthetic signals

6.1.1 Settings

We generate a graph by randomly placing $N = 500$ vertices in the unit square. We set the edge weights based on a thresholded Gaussian kernel function so that $\boldsymbol{W}_{ij} = e^{-\frac{[\text{dist}\,(i,j)]^2}{2\theta^2}}$ if the physical distance $\text{dist}(i,j)$ between vertices i and j is less than or equal to δ, and zero otherwise. We fix $\theta = 0.04$ and $\delta = 0.09$ in our experiments and ensure that the graph is connected. In our first set of experiments, we construct a set of synthetic training signals consisting of localized patterns on the graph, drawn from a dictionary that is a concatenation of $S = 3$ subdictionaries. Each subdictionary is a polynomial of the graph Laplacian of degree $K = 15$ and captures one of the three constitutive components of our synthetic signal class. We generate the graph signals by linearly combining $T_0 \leq 10$ random atoms from the dictionary with random coefficients. We then learn a dictionary from a set of 1000 training signals for different values of the parameter η that controls the robustness to quantization error in distributed signal processing.

6.1.2 Learned kernels

First, we look in more details on the effect of η on the dictionary learning outcome and study the effect of this parameter in the learned kernels. In Fig. 2a, we illustrate the original kernels of the underlying dictionary, and in Fig. 2b–d, we plot the ones recovered by solving the dictionary learning algorithm of Eq. (20) for different values of η. As expected, we observe that, as we increase the value of η, the recovered kernels become more similar to the original ones. The effect of the parameter η is more obvious in Fig. 3, where we plot the values of the polynomial coefficients. We observe that, when η is small, the polynomial coefficients become small in magnitude. These results are consistent with the bounds on the polynomial coefficients derived in Section 5. In summary, the dictionary learning algorithm is not able to capture relatively complicated kernels, which seems to be the price to pay for improved robustness to quantization.

6.1.3 Approximation performance

In the next set of experiments, we quantify the loss in the approximation performance by focusing on centralized

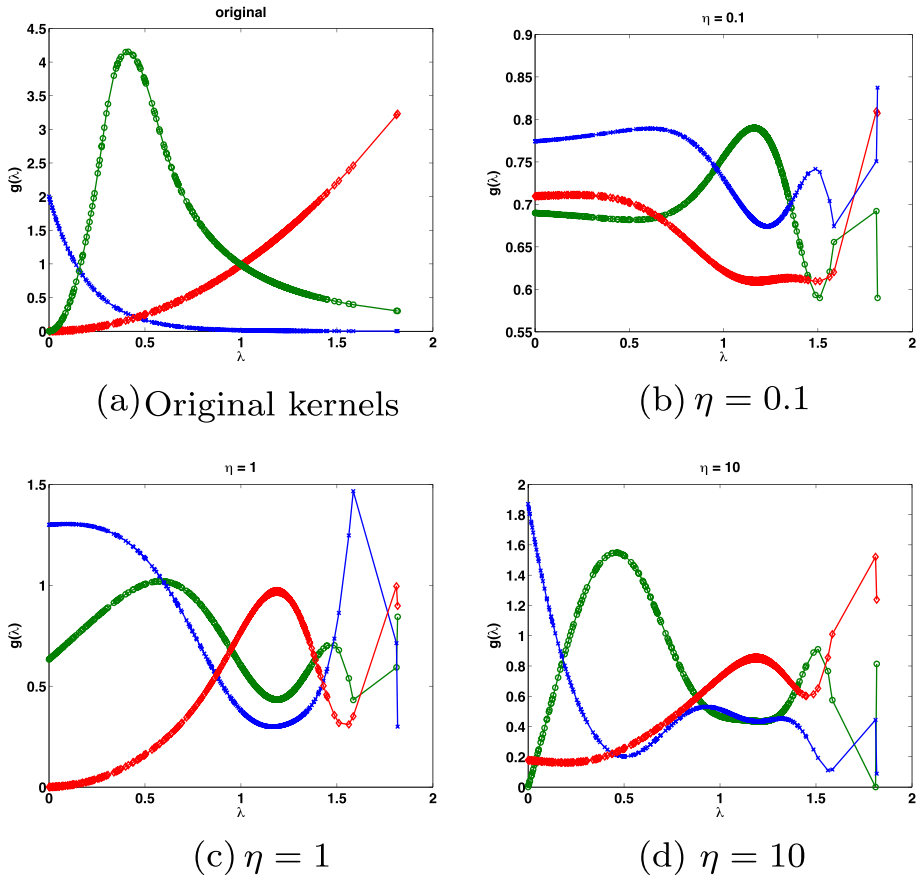


Fig. 2 Illustration of the kernels recovered for different values of η in the dictionary learning problem. (**a**) Original kernels. (**b**) η=0.1. **c** η= 1. (**d**) $\eta = 10$

settings without rate constraints. We approximate 1000 testing signals, generated in the same way as the training signals, by computing the sparse approximation in the learned dictionaries with OMP, for different sparsity levels. For the sake of comparison, we also compute the approximation performance achieved by applying OMP on the spectral graph wavelet dictionary [17]. The obtained results are illustrated in Fig. 4. Each point in the curve corresponds to the signal to approximation noise ratio (SNR in dB) for different sparsity levels, and it is computed as the average value over all the testing signals. As we reduce the values of the parameter η in the dictionary learning algorithm, the approximation performance in the ideal scenario of infinite bit rate

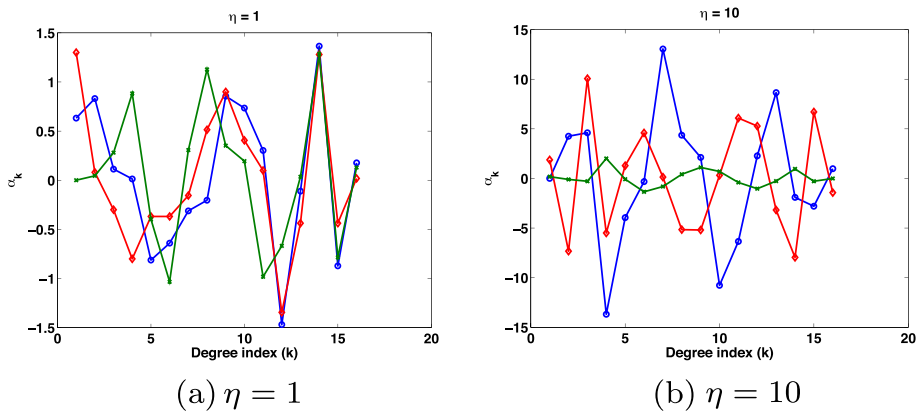


Fig. 3 Polynomial coefficient values, for each of the dictionary kernels, for different values of η in solving the optimization problem (20). (**a**) $\eta = 1$. (**b**) $\eta = 10$

deteriorates significantly. It can become even worse than the one achieved with the spectral graph wavelet dictionary, which is not learned to efficiently represent the training signals. This behavior is consistent with the conclusion drawn from Figs. 2 and 3. The more we reduce the search space (i.e., the smaller the value of η), the worse is the approximation performance of the graph signals from the learned dictionary.

6.1.4 Distributed approximation performance

Next, we move to the settings with communication constraints, and we study the distributed approximation of testing signals using the iterative soft thresholding algorithm with iterations defined by Eq. (16). The testing signals are generated in the same way as the training ones. We assume that the messages exchanged by the sensors are uniformly quantized before transmission. In particular, for each message, we send 1 bit for the sign and quantize the magnitude of the data to be transmitted to neighbor sensors. For each signal $\boldsymbol{y}$, the quantization range of the transmitted messages is defined to be $[0, \|\boldsymbol{y}\|_\infty]$, and it is known by all the sensors. We fix the bit rate to 6 bits per message, and we run ISTA for 300 iterations and different values of the sparsity parameter κ in Eq. (16). We learn different polynomial dictionaries by solving the optimization problem (20) for different values of η. For the sake of comparison, we show also the approximation performance obtained with the spectral graph wavelet dictionary approximated by a Chebyshev polynomial of order $K = 30$. In Fig. 5, we illustrate the approximation performance in terms of SNR obtained for different numbers of atoms in the representation. The number of atoms is measured by counting the number of non-zero elements in the sparse codes for a particular value of κ. Interestingly, we observe that the best representation performance is obtained when η is very small. As we increase the η, the effect of the quantization noise becomes significantly high, which leads to a dramatically low SNR in the distributed approximation algorithm. The worst performance is obtained when $\eta = \infty$, which is equivalent to ignoring the robustness constraint in the dictionary learning algorithm. This confirms that the robustness constraint can indeed reduce the effect of the quantization noise. However, comparing to the performance obtained in the case of infinite bit rate (red curve in Fig. 5), we observe a saturation in the maximum SNR that is significantly lower than the one in ideal communication conditions. This is the price to pay for introducing the quantization constraint and reducing the search space in the dictionary learning problem of (20).

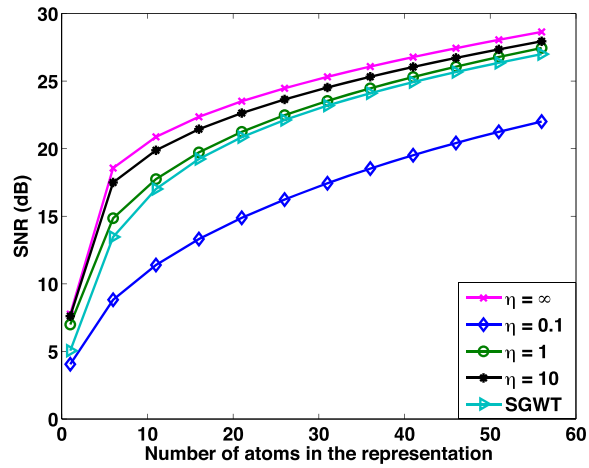


Fig. 4 Approximation performance (SNR) versus sparsity level, achieved with the polynomial graph dictionary for different values of η in centralized settings

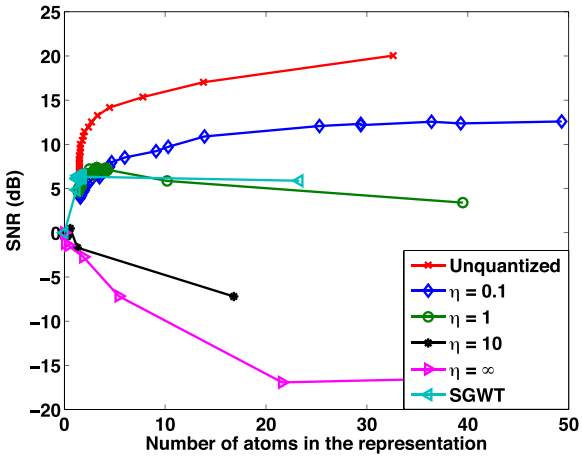


Fig. 5 Distributed approximation performance (SNR) versus sparsity level, achieved with different polynomial graph dictionary learned with different values of η in distributed settings, with a bit rate of 6 bits per message

6.1.5 Distributed denoising of graph signals

In another set of experiments, we apply our dictionaries in distributed denoising applications. We add some Gaussian noise to the testing signals such that their initial SNR is 10 dB. We then use the dictionaries learned with the different parameters η to denoise the signals by imposing a sparse prior. Denoising is performed by applying the iterative soft thresholding algorithm of Eq. (16) for different values of the parameter κ. In Fig. 6, we illustrate the SNR obtained after distributed denoising with different numbers of bits per messages, for each of the learned dictionaries. For each bit rate, we keep the value of κ that corresponds to the highest SNR. The obtained results indicate that a small η at low bit rate can bring significant gain in terms of denoising performance. When

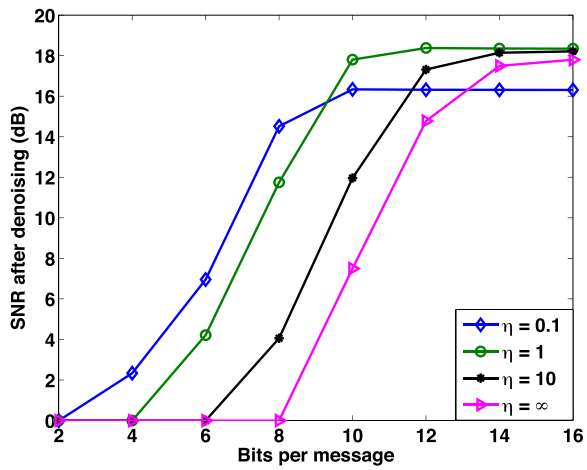


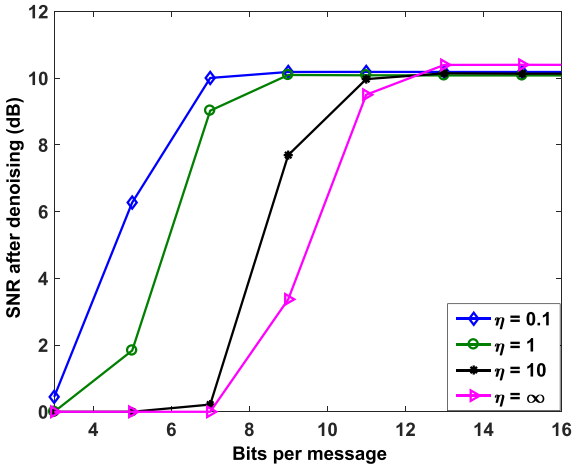
Fig. 6 Denoising performance (in dB) versus bits per message using dictionaries learned with different values of η. The SNR of the input signals is 10 dB

the bit rate is high, the denoising performance obtained with the dictionary corresponding to a small η ($\eta = 0.1$) saturates to a low SNR value. Due to the quantization constraints, the solution of the optimization problem (20) is not necessarily the optimal one, and the representation performance of the dictionary is reduced. These results are consistent with the ones obtained in the previous experiments.

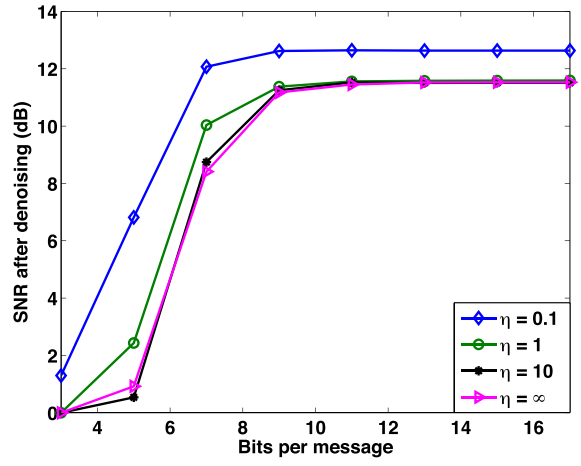
6.2 Application: denoising of sensor network signals

We now consider two real-world datasets, the one containing measurements of the evapotranspiration level and the second containing the daily bottlenecks in San Francisco county. The average monthly evapotranspiration (ETo) data are recorded at 121 measuring stations in California between January 2012 and December 2014[2]. We define a geographical graph, where the nodes of the graph consist of the sensors. Two nodes are connected if the distance between the sensors is smaller than 140 km. The weights of the graph are defined to be inversely proportional to the distance. We compute the average record per month for each station which results in 36 graph signals (i.e., one per month), each of dimension 121. The traffic data are part of the Caltrans Performance Measurement System (PeMS) dataset that provides traffic information throughout all major metropolitan areas of California [28][3]. It contains signals between January 2007 and August 2014 measured in 75 detector stations. The graph is designed by connecting stations when the distance between them is smaller than a threshold of $\theta = 0.04$. For two stations A and B, the distance d_{AB} is set to be the Euclidean distance of the GPS coordinates of the stations and the edge weights are computed using the exponential kernels such that $W_{AB} = e^{-d_{AB}}$. The signal on the graph is the duration in minutes of bottlenecks for each specific day. We remove the signal instances where no daily bottleneck was identified, and for computational issues, we normalize each signal to a unit norm.

For each of the two datasets, we use the clean signals to learn a polynomial dictionary with $S = 3$ and maximum polynomial degree of $K = 15$. In particular, for the evapotranspiration data, since the number of available signals is quite limited, we use all the 36 signals for training. The sparsity level is chosen to be $T_0 = 30$. Regarding the traffic data, we use one fourth of the signals for training and the rest for testing. Since these data are in general more sparse, we set the sparsity level to $T_0 = 5$. We run the learning algorithm for different values of the parameter

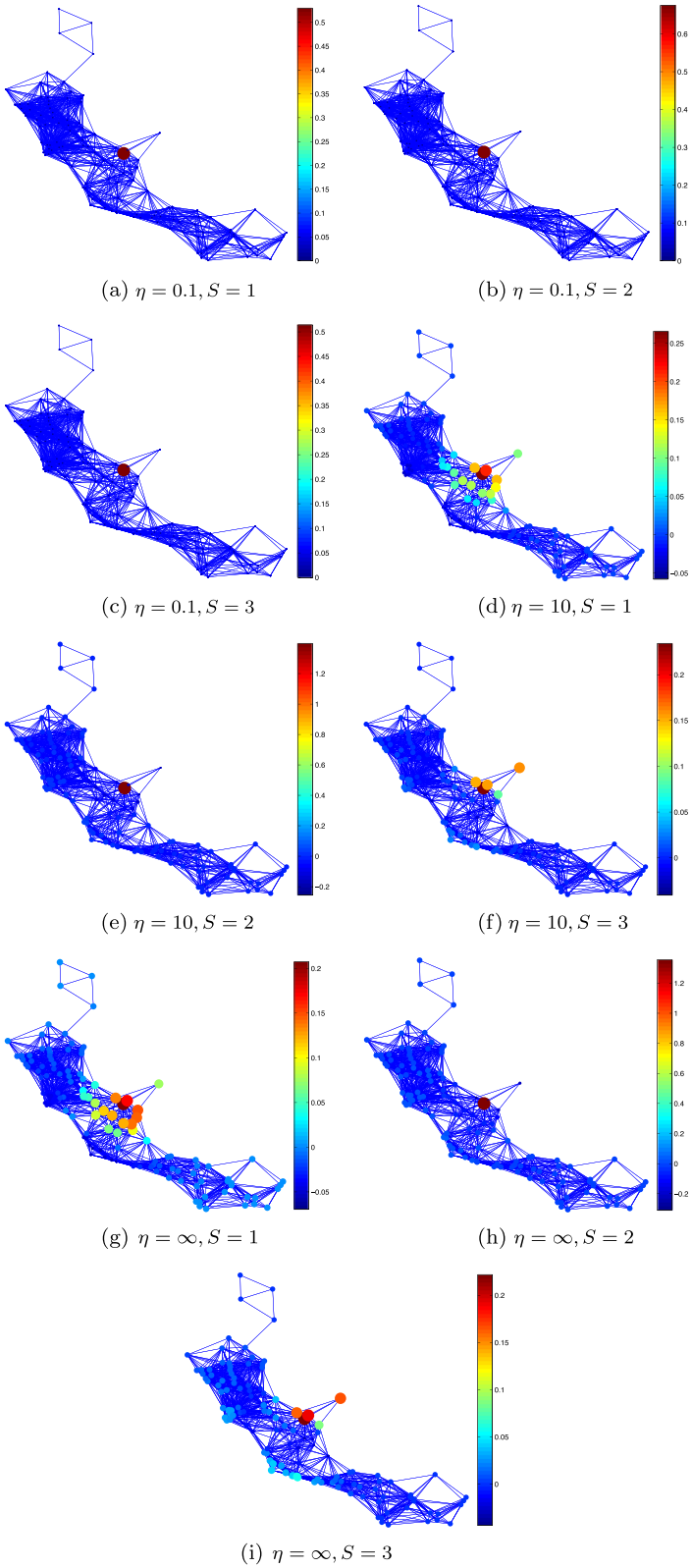


(a) Evapotranspiration data



(b) Traffic bottlenecks

Fig. 7 Denoising performance (in dB) versus bit rate per message for $l1$ regularization using dictionaries learned with different values of η. The SNR of the noisy signals is 10 dB. (**a**) Evapotranspiration data (**b**) Traffic bottlenecks



(a) $\eta = 0.1, S = 1$ (b) $\eta = 0.1, S = 2$

(c) $\eta = 0.1, S = 3$ (d) $\eta = 10, S = 1$

(e) $\eta = 10, S = 2$ (f) $\eta = 10, S = 3$

(g) $\eta = \infty, S = 1$ (h) $\eta = \infty, S = 2$

(i) $\eta = \infty, S = 3$

Fig. 8 Illustration of the atoms learned with different values of η, placed at node 10 of the cimis graph. The more we penalize the quantization error, the more localized are the atoms. **a** $\eta = 0.1$; S = 1. **b** $\eta = 0.1$; S = 2. **c** $\eta = 0.1$; S = 3. **d** $\eta = 10$; S = 1. **e** $\eta = 10$; S = 2. **f** $\eta = 10$; S = 3. **g** $\eta = \infty$; S = 1. **h** $\eta = \infty$; S = 2. **i** $\eta = \infty$; S = 3

$\eta = [0.1,\ 1,\ 10,\ \infty]$. The obtained dictionaries are then used for denoising a set of testing signals in distributed settings, at different bit rates. In the case of the ETo data, the testing signals are generated by adding Gaussian noise of zero mean and variance that depends on the desired SNR to the training signals. For the traffic data, we add Gaussian noise to the testing signals that were not used in the training. The results are illustrated in Figs. 7a, b.

First, we observe that under ideal communication, the denoising performance of the evapotranspiration signals is quite poor and the total gain is less than 1 dB. The reason for that is that these data are smoother on the graph, and our polynomial dictionary needs more atoms to approximate them. As a result, a sparse prior in the signals does not necessarily lead to big gains in terms of SNR. On the other hand, the gain observed in the traffic data is higher as such data follows our signal model and consists indeed of localized patterns on the graph. Similarly to the synthetic data, in both datasets, we observe that we achieve a significant gain when the parameter η is small at low bit rate. On the other hand, as we increase the bit budget, the performance obtained by the unconstrained dictionary ($\eta = \infty$) tends to approximate the one obtained at infinite bit rate.

Finally, we study the effect of the parameter η on the learned atoms. For simplicity, we focus only on one dataset. However, similar conclusions can be drawn from the second dataset as well. In Fig. 8, we plot the atoms that are obtained from the traffic dataset and placed at node 10 of the graph. We observe that the smaller the parameter η, i.e., the more we penalize the quantization error, the more localized are the learned atoms. In particular, we observe that all three subdictionaries tend to become similar, which means that the algorithm is learning a dictionary that can be well represented by only one subdictionary. On the other hand, the bigger the parameter η, i.e., the less we penalize the quantization error, the more the obtained atoms tend to approximate the ones with $\eta = \infty$. Thus, they are more spread on the graph. Moreover, each of the subdictionaries tends to capture distinct kernels. These results are quite intuitive and indicate that if we want to penalize the quantization noise, we should limit the communication between the nodes of the graph. The latter can be achieved by restricting both the maximum polynomial degree K, i.e., designing simpler and smoother graph kernels, and the number of distinct subdictionaries S, i.e., designing compact dictionaries with small S. This, however, comes at the cost of reduced sparsity or worse approximation performance. Since the atoms have a very local support, the number of atoms that is needed to approximate the signal is quite big. As a result, the signals are not very sparse in the obtained dictionary. This may also be a reason for a relatively poor denoising performance of the given dataset.

7 Conclusions

In this paper, we have studied the effect of quantization in distributed graph signal processing with polynomial dictionary operators. We have shown analytically that the overall quantization error depends on the communication pattern of the network captured by the graph Laplacian matrix and the graph structured dictionary. Following this observation, we have then proposed an algorithm that learns polynomial graph dictionaries to sparsely approximate graph signals while staying robust to quantization noise. We have shown that the quantization constraints penalize the magnitude of the polynomial coefficients, sacrificing on the recovery of the true dictionary atoms. Experimental results have illustrated the trade-offs between effective distributed signal representation in low bit rate communication settings and accuracy of the signal approximation in ideal settings.

Endnotes

[1] The main assumption of this paper is that the communication graph coincides with the representation graph that captures that structure of the signals.

[2] The data are publicly available at www.cimis.water.ca.gov.

[3] The data are publicly available at http://pems.dot.ca.gov.

Appendix

Proof For ease of notation, we ignore the iteration index and we bound the error norm as follows:

$$\begin{aligned}\|\boldsymbol{e}\| &= \left\| E\left(\boldsymbol{\mathcal{D}}^T\boldsymbol{y}\right) - \boldsymbol{\mathcal{D}}^T E\left(\boldsymbol{\mathcal{D}}\boldsymbol{x}\right) - E\left(\boldsymbol{\mathcal{D}}^T\boldsymbol{\mathcal{D}}\boldsymbol{x}\right)\right\| \\ &\leq \left\| E\left(\boldsymbol{\mathcal{D}}^T\boldsymbol{y}\right)\right\| + \left\|\boldsymbol{\mathcal{D}}^T E\left(\boldsymbol{\mathcal{D}}\boldsymbol{x}\right)\right\| + \left\| E\left(\boldsymbol{\mathcal{D}}^T\boldsymbol{\mathcal{D}}\boldsymbol{x}\right)\right\| \\ &\leq \sum_{s=1}^{S}\left\{\left\| E\left(\boldsymbol{\mathcal{D}}_s^T\boldsymbol{y}\right)\right\| + \left\|\boldsymbol{\mathcal{D}}_s^T E\left(\boldsymbol{\mathcal{D}}\boldsymbol{x}\right)\right\| + \left\| E\left(\boldsymbol{\mathcal{D}}_s^T\boldsymbol{\mathcal{D}}\boldsymbol{x}\right)\right\|\right\},\end{aligned} \tag{26}$$

where we have used the standard Cauchy-Schwarz inequality. For the sake of simplicity, we work with each term of the summation separately. After some basic operations with matrix norms, we can write the first term as follows:

$$\begin{aligned}\left\| E\left(\boldsymbol{\mathcal{D}}_s^T\boldsymbol{y}\right)\right\| &= \left\|\sum_{l=0}^{K-1}\left[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right]\boldsymbol{\epsilon}_l\right\| \\ &\leq \sum_{l=0}^{K-1}\left\|\left[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right]\boldsymbol{\epsilon}_l\right\| \\ &\leq \sum_{l=0}^{K-1}\left\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right\|\|\boldsymbol{\epsilon}_l\|.\end{aligned} \tag{27}$$

Similarly, the second and the third terms can be written as:

$$\left\|\boldsymbol{\mathcal{D}}_s^T E\left(\boldsymbol{\mathcal{D}}\boldsymbol{x}\right)\right\| = \left\|\sum_{l=0}^{K-1}\left[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right]\boldsymbol{\xi}_l\right\| \leq \sum_{l=0}^{K-1}\left\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathbf{L}^j\right\|\left\|\boldsymbol{\xi}_l\right\|, \tag{28}$$

and

$$\begin{aligned}\left\|E\left(\boldsymbol{\mathcal{D}}_s^T\boldsymbol{\mathcal{D}}\boldsymbol{x}\right)\right\| &= \left\|\sum_{k=0}^{K}\alpha_{sk}\mathbf{L}^k\sum_{s'=1}^{S}\sum_{l'=0}^{K-1}\left[\sum_{j'=1}^{K-l'}\alpha_{s'(l'+j')}\mathbf{L}^{j'}\right]\boldsymbol{\zeta}_{s',l'}\right\| \\ &\leq \left\|\sum_{k=0}^{K}\alpha_{sk}\mathbf{L}^k\right\|\sum_{s'=1}^{S}\sum_{l'=0}^{K-1}\left\|\sum_{j'=1}^{K-l'}\alpha_{s'(l'+j')}\mathbf{L}^{j'}\right\|\left\|\boldsymbol{\zeta}_{s',l'}\right\| \\ &\leq c\sum_{s'=1}^{S}\sum_{l'=0}^{K-1}\left\|\sum_{j'=1}^{K-l'}\alpha_{s'(l'+j')}\mathbf{L}^{j'}\right\|\left\|\boldsymbol{\zeta}_{s',l'}\right\|,\end{aligned} \tag{29}$$

where the last inequality comes from the assumption that $0\boldsymbol{I} \preceq \boldsymbol{\mathcal{D}}_s \preceq c\boldsymbol{I}$, which is a necessary assumption (see the optimization problem 20) for the class of spectral graph dictionaries that we are considering. Combining Eqs. (26)–(29), and using the assumption that the quantization noise is uniformly distributed with magnitude smaller than $\Delta/2$, we obtain the upper bound of (18). □

Acknowledgements
The authors would like to thank the anonymous reviewers for their constructive comments.

Authors' contributions
Both authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]Swiss Data Science Center, EPFL, Lausanne, Switzerland. [2]Signal Processing Laboratory (LTS4), EPFL, Lausanne, Switzerland.

Received: 1 February 2018 Accepted: 14 September 2018
Published online: 24 October 2018

References

1. D.I. Shuman, S.K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Process. Mag. **30**(3), 83–98 (2013)
2. X. Zhang, X. Dong, P. Frossard, in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process*. Learning of structured graph dictionaries, (Kyoto, 2012), pp. 3373–3376
3. D. Thanou, D.I. Shuman, P. Frossard, Learning parametric dictionaries for signals on graphs. IEEE Trans. Signal Process. **62**(15), 3849–3862 (2014)
4. D.I. Shuman, P. Vandergheynst, P. Frossard, in *Proc. IEEE Int. Conf. Distr. Comput. Sensor Sys*. Chebyshev polynomial approximation for distributed signal processing, (Barcelona, 2011), pp. 1–8
5. D.I. Shuman, P. Vandergheynst, P. Frossard, Distributed signal processing via Chebyshev polynomial approximation. arXiv:1111.5239 (2011)
6. X. Wang, M. Wang, Y. Gu, A distributed tracking algorithm for reconstruction of graph signals. IEEE J. Sel. Top. Signal Proc. **9**(4), 728–740 (2015)
7. S. Chen, A. Sandryhaila, J. Kovacevic, in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process*. Distributed algorithm for graph signal inpainting, (Brisbane, 2015), pp. 3731–3735
8. S. Safavi, U.A. Khan, Revisiting finite-time distributed algorithms via successive nulling of eigenvalues. IEEE Signal Proc. Lett. **22**(1), 54–57 (2015)
9. P.D. Lorenzo, M.P. Banelli, S. Barbarossa, S. Sardellitti, Distributed adaptive learning of graph signals. IEEE Trans. Signal Process. **65**(16), 4193–4208 (2017)
10. M.S. Segarra, A.G. Marques, A. Ribeiro, Optimal graph-filter design and applications to distributed linear network operators. IEEE Trans. Signal Proc. **65**(15), 4117–4131 (2017)
11. E. Isufi, A. Loukas, A. Simonetto, G. Leus, Autoregressive moving average graph filtering. IEEE Trans. Signal Proc. **65**(2), 274–288 (2017)
12. X. Shi, H. Feng, M. Zhai, T. Yang, B. Hu, Revisiting finite-time distributed algorithms via successive nulling of eigenvalues. IEEE Signal Proc. Lett. **22**(8), 1113–1117 (2015)
13. A. Sandryhaila, J.M.F. Moura, Discrete signal processing on graphs. IEEE Trans. Signal Proc. **61**(7), 1644–1656 (2013)
14. J. Liu, E. Isufi, G. Leus, Filter design for autoregressive moving average graph filters. arXiv:1711.09086 (2018)
15. D. Thanou, P. Frossard, in *Annual Allerton Conf. on Communic., Contr., and Comput*. Distributed signal processing with graph spectral dictionaries (IEEE, Monticello, 2015), pp. 1391–1398
16. L.F.O. Chamon, A. Ribeiro, in *Proc. IEEE Glob. Conf. Signal and Inform. Process*. Finite-precision effects on graph filters, (Monteral, 2017)
17. D. Hammond, P. Vandergheynst, R. Gribonval, Wavelets on graphs via spectral graph theory. Appl. Comput. Harmon. Anal. **30**(2), 129–150 (2010)
18. D.I. Shuman, M.J. Faraji, P. Vandergheynst, in *Proc. of the Int. Conf. on Sampling Theory and Applications*. Semi-supervised learning with spectral graph wavelets, (Singapore, 2011)
19. S. Segarra, A.G. Marques, G. Leus, A. Ribeiro, Reconstruction of graph signals through percolation from seeding nodes. IEEE Trans. Signal Proc. **64**(16), 4363–4378 (2016)
20. S. Chen, D. Donoho, M. Saunders, Atomic decomposition by basis pursuit. SIAM J. Sci. Comput. **20**(1), 33–61 (1999)
21. R. Tibshirani, Regression shrinkage and selection via the lasso. J. Royal Stat. Society Ser. B. **58**, 267–288 (1994)
22. S. Chen, D. Donoho, M. Saunders, Atomic decomposition by basis pursuit. SIAM Rev. **43**(1), 129–159 (2001)
23. A. Beck, M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems. Springer Optimization and Its Applications, vol 49.* (Springer, New York, 2009), pp. 183–202
24. P. Combettes, J.C. Pesquet, Proximal splitting methods in signal processing. Fixed-Point Algorithms for Inverse Problems in Science and Engineering, 185–212 (2011)
25. S. Sra, in *Adv. Neural Inf. Process. Syst*. Scalable nonconvex inexact proximal splitting, (Lake Tahoe, 2012), pp. 530–538
26. J.A. Tropp, Greed is good: algorithmic results for sparse approximation. IEEE Trans. Inform. Theory. **50**(10), 2231–2242 (2004)
27. S. Boyd, L. Vandenberghe, *Convex optimization*. (Cambridge University Press, New York, 2004)
28. T. Choe, A. Skabardonis, P.P. Varaiya, in *Annual Meeting of Transportation Research Board*. Freeway performance measurement system (PeMS): an operational analysis tool, (2002)