Distributed Signal Processing with Graph Spectral Dictionaries

Dorina Thanou and Pascal Frossard

***Abstract*— We study the distributed processing of graph signals that are well represented by graph spectral dictionaries. We first analyze the impact of quantization noise in the distributed computation of polynomial dictionary operators that are commonly used in various signal processing tasks. We show that the impact of quantization depends on the graph geometry and on the structure of the spectral dictionaries. Then, we focus on the problem of distributed sparse signal representation that can be solved with an iterative soft thresholding algorithm. We define conditions on the dictionary structure to ensure the convergence of the distributed algorithm and finally propose a dictionary learning solution that permits to control the robustness to quantization noise. Experimental results for reconstruction and denoising of both synthetic and practical signals illustrate the tradeoffs that exist between accurate signal representation and robustness to quantization error in the design of dictionaries operators in distributed graph signal processing.**

***Index Terms*— sparse approximation, graph signal processing, quantization, polynomial dictionaries**

I. Introduction

Wireless sensor networks have been widely used for applications such as surveillance, weather monitoring, or automatic control that are often supported by distributed signal processing methods. In such 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 representations are certainly powerful and promising tools for representing signals in the irregular structured domain defined by sensor networks [1]. In particular, when graph signals mostly capture the effect of a few processes on a graph topology, they can be modelled as the linear combinations of a small number of constitutive components in a dictionary, namely the graph atoms. Spectral graph dictionaries [2], [3], [4], [5], which can also be considered as a set of well-defined 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. Due to their polynomial structure [4], [5], or their effective approximation with Chebyshev polynomials [6], [7], the spectral graph dictionaries can be implemented distributively and therefore represent ideal operators for graph signal processing in sensor networks.

*D. Thanou and P. Frossard are with Ecole Polytechnique Fédérale de Lausanne (EPFL), Signal Processing Laboratory-LTS4, Lausanne, Switzerland, Emails:{`dorina.thanou`, `pascal.frossard`}`@epfl.ch`.

In wireless sensor networks, each sensor node typically communicates only with a small number of neighbor nodes due to energy constraints and limited communication range. In addition, the information exchanged by the network nodes is quantized prior to transmission because of limitations in communication bandwidth and computational power. The quantization process induces some noise that impacts the performance of sensor network applications and requires careful consideration to ensure the proper convergence of the distributed signal processing algorithms.

In this work, 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. 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 governed by the characteristics of the dictionary. We then propose an algorithm for learning polynomial graph dictionaries that permit to control the robustness of distributed algorithms to quantization noise. Experimental results illustrate the dictionary design tradeoffs between accurate signal representation and robustness to quantization errors, and show in particular that it is important to sacrifice on signal approximation performance for ensuring proper convergence of distributed algorithms in low bit rate settings.

The paper is organized as follows. In Section II, we model the sensor network with a graph, and we recall the use of polynomial graph dictionaries for sparse representation of graph signals. We study the quantization error that appears in the distributed computations with polynomial graph dictionaries in Section III, and we propose an algorithm for learning polynomial graph dictionaries that are robust to the quantization noise in Section V. Finally, in Section VI we evaluate the performance of our algorithm in both synthetic and real world signals.

II. Polynomial graph dictionaries for distributed signal representation

In this section, we review some of the basic concepts from the literature and introduce notations that are needed for the rest of the paper.

A. Distributed sensor network topology

We consider a sensor network topology that is modeled as a weighted, undirected graph $\mathcal{G} = (V, \mathcal{E})$, where $V \in$

$\{1, \dots, N\}$ represents the set of sensor nodes and $N = |V|$ 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 $W(i, j) > 0$ is assigned to each edge if $\{i, j\} \in \mathcal{E}$, whose value is dependent on the distance between the nodes i and j. D is a diagonal degree matrix that contains as elements the sum of each row of the matrix 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 defined as $\mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, and it is a real symmetric matrix that has a complete set of orthonormal eigenvectors with corresponding nonnegative eigenvalues. We denote its eigenvectors by $\chi = [\chi^1, \chi^2, ..., \chi^N]$, and the spectrum of eigenvalues by $\Lambda := \left\{ 0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{(N-1)} \leq 2 \right\}$. The eigenvalues of the graph Laplacian provide a notion of frequency on the graph and the corresponding eigenvectors define the graph Fourier transform [1].

B. Sparse graph signal model

We consider a general class of graph signals that are sparse linear combinations of (overlapping) graph patterns $\widehat{g}(\cdot)$, defined in the spectral domain of the graph, and positioned at different vertices on the graph [4]. Each pattern captures the local form of the 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 a pattern across the vertices of the graph [2], [8] through a graph operator defined as

$$\widehat{g}(\mathcal{L}) = \chi \widehat{g}(\Lambda) \chi^T, \tag{1}$$

where χ, Λ are the eigenvectors and eigenvalues of the graph Laplacian matrix. The generating kernel $\widehat{g}(\cdot)$, that is a function of the eigenvalues of the Laplacian, characterizes the graph pattern in the spectral domain. In particular, if the generating kernel is smooth, $\widehat{g}(\mathcal{L})$ consists of a set of N columns, each representing a localized atom, generated by the same kernel, and positioned on different nodes on the graph [2], [1]. One can thus 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 [2], [4]:

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

The corresponding graph operator is then defined as

$$\widehat{g}(\mathcal{L}) = \chi \left(\sum_{k=0}^{K} \alpha_k \Lambda^k \right) \chi^T = \sum_{k=0}^{K} \alpha_k \mathcal{L}^k. \tag{3}$$

Note that the atom given by the column n is equal to the polynomial $\widehat{g}(\cdot)$ of order K translated to the vertex n. The polynomial structure of the kernel $\widehat{g}(\cdot)$ ensures that the resulting atom has its support contained in a K-hop neighborhood of vertex n [2]. A graph dictionary is then defined as a concatenation of subdictionaries in the form $\mathcal{D} = [\widehat{g_1}(\mathcal{L}), \widehat{g_2}(\mathcal{L}), ..., \widehat{g_S}(\mathcal{L})]$, where each subdictionary captures a different graph pattern in the spectral domain that is translated across all the vertices of the graph. Finally, a graph signal y can be expressed as a linear combination of a set of atoms generated from different graph kernels $\{\widehat{g_s}(\cdot)\}_{s=1,2,...,S}$,

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

where x_s are the coefficients in the linear combination. An example of the dictionary $\mathcal{D}$ is the spectral graph wavelet dictionary [2], or more generally the union of graph Fourier multipliers that can be efficiently approximated with Chebyshev polynomials [7]. In applications where sparsity is required, one can typically learn the polynomial coefficients numerically from a set of training signals that live on the graph as shown in [4].

C. Distributed computation of the graph operators

An important benefit of the polynomial graph dictionaries is the fact that they can be efficiently stored and implemented in distributed signal processing tasks. Each polynomial dictionary can be implemented locally, i.e., by exchanging only information between nodes on a graph that are connected by an edge. These dictionaries can be used in solving in a distributed fashion classical signal processing algorithms including regression problems involving learning or regularizing a scalar function defined on the graph, such as denoising [6], semi-supervised learning [9], signal reconstruction [4], interpolation and reconstruction of band limited graph signals [5]. These type of applications typically require the computation of quantities such as the forward application of the dictionary and its adjoint to be computed by only local exchange of information. As discussed in [2], [6], [7], [4], the polynomial structure of the dictionary enables such quantities to be efficiently computed in a distributed fashion.

We consider, as an illustration, the distributed processing scenario where each node n of the graph wants to compute the sparse decomposition in a polynomial dictionary by solving a sparse regularization problem of the form

$$x^* = \min_x ||y - \mathcal{D}x||_2^2 + \kappa ||x||_1, \tag{4}$$

where κ is a parameter that controls the sparsity level. The main assumption is that the node knows its own component of a signal $y \in \mathbb{R}^N$, the n^{th} row of the corresponding Laplacian matrix $\mathcal{L}$, and the polynomial coefficients used in the dictionary. Solving Eq. (4) requires the computation of the operations $\mathcal{D}x$, $\mathcal{D}^T y$, $\mathcal{D}^T \mathcal{D} x$ in a distributed way. As described in [6], these operations can be eventually computed after some local linear filtering steps.

III. Distributed quantized operations with polynomial dictionaries

We now study the effect of quantization in distributed signal processing with polynomial dictionaries by modelling the propagation of the quantization error in the different dictionary-based operators. Given a graph signal y, and the representation of the signal in a polynomial graph dictionary $\mathcal{D}$, i.e., $y = \mathcal{D}x$, as described in Section II, we study the computation of three basic operators i.e., the forward $\mathcal{D}x$, the adjoint operator $\mathcal{D}^T y$, and the $\mathcal{D}^T\mathcal{D}x$ operator in distributed settings.

1) Distributed computation of $\mathcal{D}^T y$: The distributed computation of $\mathcal{D}^T y$ requires first the computation of the different powers of the Laplacian matrix, i.e., $\{\mathcal{L}^0 y, \mathcal{L}y, \mathcal{L}^2 y, ..., L^K y\}$, in a distributed way. The latter can be done efficiently by successive multiplications of the matrix $\mathcal{L}$ with the signal y over $K+1$ iterations. We introduce a new variable z_k that captures the transmitted value during the k iteration, with $z_0 = y$, in the computation of $\mathcal{D}^T y$. Before the sensors exchange information, the value of sensor n at iteration k, i.e., $z_k(n)$, is quantized such that

$$\tilde{z}_k(n) = z_k(n) + \epsilon_k(n), \tag{5}$$

where $\epsilon_k(n)$ is the quantization error in k, $z_k(n)$ is the value of the sensor before quantization and $\tilde{z}_k(n)$ is the quantized value that the sensor n sends to its neighbors. Then, each node updates its value as a linear combination of its own quantized value and the quantized values received from its neighbors $z_k(i)$ with $i \in \mathcal{N}_n$, based on the recursive update

$$z_{k+1} = \mathcal{L}(z_k + \epsilon_k), \forall k = [0, 1, ..., K-1] \tag{6}$$

where $\epsilon_k = (\epsilon_k(1), \epsilon_k(2), ..., \epsilon_k(N))$. Taking into consideration the quantization error from the previous iterations, Eq. (6) can be re-written as

$$z_{k+1} = \mathcal{L}^{k+1} z_0 + \sum_{l=0}^{k} \mathcal{L}^{k+1-l}\epsilon_l. \tag{7}$$

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).

Next, we compute $\widetilde{\mathcal{D}_s y}$, the quantized vector corresponding to $\mathcal{D}_s y$, by applying the polynomial coefficients on the values generated by the sequence $\{z_0, z_1, ..., z_K\}$ as follows

$$\begin{aligned}\widetilde{\mathcal{D}_s^T y} &= \sum_{k=0}^{K} \alpha_{sk} z_k = \sum_{k=0}^{K} \alpha_{sk}\mathcal{L}^k y + \sum_{l=1}^{K}\Big[\sum_{j=1}^{l-1} \alpha_{sl}\mathcal{L}^{l-j}\epsilon_j\Big] \\ &= \sum_{k=0}^{K} \alpha_{sk}\mathcal{L}^k y + \sum_{l=0}^{K-1}\Big[\sum_{j=1}^{K-l} \alpha_{s(l+j)}\mathcal{L}^j\Big]\epsilon_l \\ &= \mathcal{D}_s^T y + E(\mathcal{D}_s^T y),\end{aligned} \tag{8}$$

where Eq. (8) is obtained after some simple matrix manipulations, and we have set

$$E(\mathcal{D}_s^T y) = \sum_{l=0}^{K-1}\Big[\sum_{j=1}^{K-l} \alpha_{s(l+j)}\mathcal{L}^j\Big]\epsilon_l.$$

Finally, the operation $\widetilde{\mathcal{D}^T y} = \{\widetilde{\mathcal{D}_s^T y}\}_{s=1}^S$ can be written as

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

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

```
 1: Inputs at node $n$: $y(n), \mathcal{L}_{n,:}$, $\alpha = [\alpha_1; ...; \alpha_S]$, quantization stepsize $\Delta$
 2: Output at node $n$: $\{(\widetilde{\mathcal{D}^T y})_{(s-1)N+n}\}_{s=1,..,S}$
 3: Quantize and transmit $\tilde{y}(n) = y + \epsilon_0(n)$ to all $m \in \mathcal{N}_n$
 4: Receive $\tilde{y}(m)$ from neighbors in $\mathcal{N}_n$
 5: Set $z_0(n) = y(n)$,   $\tilde{z}_0(n) = \tilde{y}(n)$.
 6: for $k = 2, ..., K$ do:
 7:     Compute $z_{k-1}(n) = (\mathcal{L}^T\tilde{z}_{k-2})_n$
 8:     Quantize and transmit $\tilde{z}_{k-1}(n) = (\mathcal{L}^T\tilde{z}_{k-2})_n + \epsilon_{k-1}(n)$ to all the neighbors
 9:     Receive $\tilde{z}_{k-1}(m)$ from all the neighbors $m \in \mathcal{N}_n$.
10: end for
11: for $s = 1, .., S$ do
12:     Compute $(\widetilde{\mathcal{D}^T y})_{(s-1)N+n} = \sum_{k=0}^{K} \alpha_{ks} z_k(n)$
13: end for
```

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

2) Distributed computation of $\mathcal{D}x$: We recall that $\mathcal{D}x = \sum_{s=1}^S \mathcal{D}_s x_s$, where $x = (x_1,\ x_2,\ \dots\ , x_S)$ is a vector containing as entries the sparse codes 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 above. Following the same reasoning as in the computation of $\mathcal{D}_s^T y$, we obtain

$$\begin{aligned}\widetilde{\mathcal{D}x} &= \sum_{s=1}^{S}\Big\{\sum_{k=0}^{K} \alpha_{sk}\mathcal{L}^k x_s + \sum_{l=0}^{K-1}\Big[\sum_{j=1}^{K-l} \alpha_{s(l+j)}\mathcal{L}^j\Big]\zeta_{s,l}\Big\} \\ &= \sum_{s=1}^{S}[\mathcal{D}_s x_s + E(\mathcal{D}_s x_s)],\end{aligned} \tag{9}$$

where $\zeta_{s,l}$ is the quantization error vector that occurs while transmiting the l^{th} power of the Laplacian, multiplied by x_s. The main steps of the distributed algorithm for the computation of $\mathcal{D}x$ are shown in Alg. 2.

3) Distributed computation of $\mathcal{D}^T\mathcal{D}x$: By combining Eqs. (8), (9), we can compute $\mathcal{D}^T\mathcal{D}x$ as follows,

$$\begin{aligned}\widetilde{\mathcal{D}_s^T\mathcal{D}x} &= \sum_{k=0}^{K} \alpha_{sk}\mathcal{L}^k\widetilde{\mathcal{D}x} + \sum_{l=0}^{K-1}\Big[\sum_{j=1}^{K-l} \alpha_{s(l+j)}\mathcal{L}^j\Big]\xi_l \\ &= \sum_{k=0}^{K} \alpha_{sk}\mathcal{L}^k \sum_{s'=1}^{S}\Big\{\sum_{k'=0}^{K} \alpha_{s'k'}\mathcal{L}^{k'} x_{s'} \\ &+ \sum_{l'=0}^{K-1}\Big[\sum_{j'=1}^{K-l'} \alpha_{s'(l'+j')}\mathcal{L}^{j'}\Big]\zeta_{s',l'}\Big\} + \sum_{l=0}^{K-1}\Big[\sum_{j=1}^{K-l} \alpha_{s(l+j)}\mathcal{L}^j\Big]\xi_l \\ &= \mathcal{D}_s^T\mathcal{D}x + \sum_{k=0}^{K} \alpha_{sk}\mathcal{L}^k E(\mathcal{D}x) + E(\mathcal{D}_s^T\mathcal{D}x),\end{aligned} \tag{10}$$

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

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

where ξ_l is the quantization noise that occurs while transmitting l^{th} power of the Laplacian multiplied by $\widetilde{\mathcal{D}x}$. Note that we have set

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

$$E(\mathcal{D}_s^T\mathcal{D}x) = \sum_{l=0}^{K-1}\Big[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^{j}\Big]\xi_l.$$

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

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

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

IV. Distributed graph signal regularization with sparse prior

In the following, we use the computation of the above operators for the sparse distributed representation of a signal 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 [10] of the form of Eq. (4) that we recall here

$$x^* = \min_x ||y - \mathcal{D}x||_2^2 + \kappa||x||_1.$$

The above problem can be solved by iterative soft thresholding (ISTA) [11], in which the update of the estimated coefficients is given by

$$x^t = \mathcal{S}_{\kappa\tau}\big(x^{(t-1)} + 2\tau\mathcal{D}^T\big(y - \mathcal{D}x^{(t-1)}\big)\big), \quad t = 1, 2, ... \tag{11}$$

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

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

Due to the polynomial structure of the dictionary, ISTA can be solved in a distributed way as shown in [6]. The estimate of the signal is then given by $\hat{y} = Dx^*$.

The first step of ISTA requires the computation of the gradient of the term $\|y - \mathcal{D}x\|^2$, which implies the computation of the operations $\mathcal{D}^Ty$, $\mathcal{D}^T\mathcal{D}x$ in each iteration of the algorithm. When the communication is quantized, the quantization noise introduces an error in the gradient. Eq. (11) can then be written as

$$\tilde{x}^t = \mathcal{S}_{\kappa\tau}\big(x^{(t-1)} + 2\tau\big(\mathcal{D}^Ty - \mathcal{D}^T\mathcal{D}x^{(t-1)} + e^{(t-1)}\big)\big), \tag{12}$$

where $e^{(t-1)}$ is the total error induced by the distributed computation of $\widetilde{\mathcal{D}^T\mathcal{D}x}$, $\widetilde{\mathcal{D}^Ty}$. According to Eqs. (8), (10) the total error can be expressed as

$$e^{(t-1)} = E^{(t-1)}(\mathcal{D}^Ty) + \mathcal{D}^T E^{(t-1)}(\mathcal{D}x) + E^{(t-1)}(\mathcal{D}^T\mathcal{D}x).$$

The convergence of the ISTA algorithm then depends on the sequence of error and is characterized by the following result from [12].

Theorem 1 ([12]). *Let f a gradient Lipschitz 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:*

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

Then, the sequence generated by the iterates

$$x^{(t)} = prox_{\tau_{t-1}g}(x^{(t-1)} - \tau_{t-1}\nabla f(x^{(t-1)}) + \tau_{t-1}e^{(t-1)})$$

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

$$\forall\tau \le \bar{\tau}, \quad \tau\|e^{(t-1)}\| \le \bar{\epsilon}, \quad \text{for some } \bar{\epsilon} \ge 0, \forall t.$$

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 the gradient error norm converges towards 0, then the sequence limit point is the exact solution of the problem.

For ISTA to converge in our distributed settings, we have to make sure that the error in the gradient is bounded. We use a uniform quantizer with a quantization stepsize Δ for the magnitude, and we send one more bit for the sign. Under this assumption, an upper-bound on the norm of the error is given by the following lemma.

Lemma 1. *Let $e^{(t-1)}$ be the error due to quantization in the computation of the gradient at iteration $t-1$ as defined* *in Eq. (12) and Δ the quantization stepsize of a uniform quantizer. Then,*

$$\|e^{(t-1)}\| \leq \sqrt{N}\frac{\Delta}{2}\sum_{s=1}^{S}\Big\{2\sum_{l=0}^{K-1}\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j\| + c\sum_{s'=1}^{S}\sum_{l'=0}^{K-1}\|\sum_{j'=1}^{K-l'}\alpha_{s'(l'+j')}\mathcal{L}^{j'}\|\Big\}. \qquad (13)$$

The proof of Lemma 1 is provided in the appendix. The above inequality shows that the error $\|e^{(t-1)}\|$ in the gradient is upper-bounded by functions of the quantization errors $\|\epsilon_l\|, \|\xi_l\|, \|\zeta_{s',l'}\|$, multiplied by a matrix polynomial of $\mathcal{L}$. The quantization errors depend on the number of bits, i.e., the rate constraints, and for a uniform quantizer, they are upper-bounded by the quantization stepsize. In particular, if the norm $\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j\|$, for $l \in \{1,...,K-1\}$, $s \in \{1,...,S\}$ is bounded by a constant $\eta > 0$, Eq. (13) becomes:

$$\|e^{(t-1)}\| \leq \sqrt{N}\frac{\Delta}{2}SK(2+c)\eta. \qquad (14)$$

Thus, under the condition that $\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j\| < \eta$, the error of the gradient at each iteration is bounded, which implies that ISTA converges to a local minimum. Moreover, as $\Delta \to 0$, i.e., the bit rate tends to infinity, then $\|e^{(t-1)}\| \to 0$ independently of η. On the other hand, when the number of bits is limited and fixed, the quantization noise depends on the characteristics of the dictionary. In particular, as $\eta \to 0$, $\|e^{(t-1)}\| \to 0$. Finally, the upper-bound indicates that the higher the degree K of the polynomial, the more the error is accumulated over the iterations. The latter is quite intuitive as higher polynomial degree requires more information to be exchanged between the sensors, at the cost of a bigger propagation of the quantization noise. However, a big K guarantees that the polynomial functions can better approximate the underlying spectral kernels and as a consequence the graph signals. The latter indicates that there is a tradeoff between the representation performance of the polynomial dictionary and the propagation of the quantization noise.

V. Polynomial dictionary learning for quantized communication

In this section, 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. When the quantization bit budget and the graph are given, the total error can be controlled by choosing proper values for the polynomial coefficients $\{\alpha_{sk}\}_{s=1,k=0}^{S,K}$. From Eq. (13), the polynomial coefficients need to be computed in such a way that the spectral norm $\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j\|$, for $l \in \{1,...,K-1\}$, compensates for the limited number of bits. We recall that the spectral norm is defined as $\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j\| = \lambda_{max}\Big(\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j\Big)$, which due to the fact that the matrix is symmetric, is simply the largest eigenvalue of $\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j$. Thus constraining the spectral norm is equivalent to constraining the eigenvalues of the corresponding matrix.

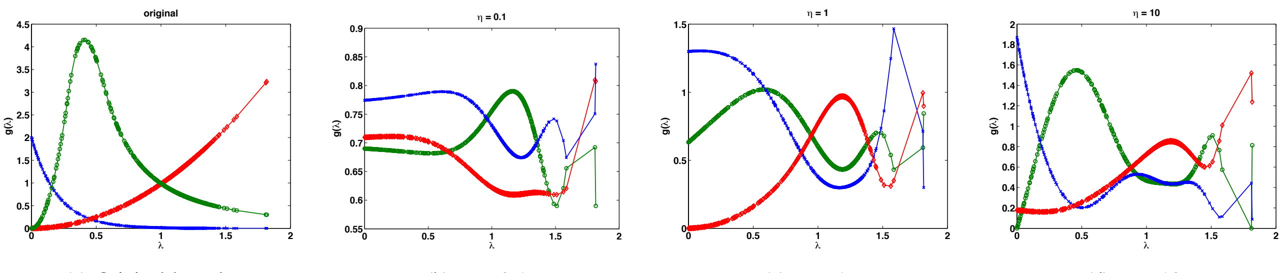
We use the condition on the maximum eigenvalue to construct dictionaries that are robust to the quantization noise. Given a set of training signals $Y = [y_1, y_2, ..., y_M] \in \mathbb{R}^{N\times M}$, all living on the weighted graph $\mathcal{G}$, our objective is to learn a polynomial graph dictionary $\mathcal{D} \in \mathbb{R}^{N\times NS}$, that can efficiently represent all of the signals in 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 $\mathcal{D}$ has the form (3), this is equivalent to learning the parameters $\{\alpha_{sk}\}_{s=1,2,...,S;\ k=1,2,...,K}$ that characterize the set of generating kernels, $\{\widehat{g_s}(\cdot)\}_{s=1,2,...,S}$. We denote these parameters in vector form as $\alpha = [\alpha_1; ...; \alpha_S]$, where α_s is a column vector with $(K+1)$ entries. To penalize the effect of the quantization noise, we impose an additional constraint to the original problem [4], which bounds the eigenvalues of the matrix $\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j$, for $l \in \{1,...,K-1\}$, $s \in \{1,...,S\}$.

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

$$\begin{aligned}
\underset{\alpha\in\mathbb{R}^{(K+1)S},\ X\in\mathbb{R}^{SN\times M}}{\text{argmin}} \quad & \big\{||Y-\mathcal{D}X||_F^2 + \mu\|\alpha\|_2^2\big\} \\
\text{subject to} \quad & \|x_m\|_0 \leq T_0, \quad \forall m \in \{1,...,M\}, \\
& \mathcal{D}_s = \sum_{k=0}^{K}\alpha_{sk}\mathcal{L}^k, \forall s \in \{1,2,...,S\} \\
& 0I \preceq \mathcal{D}_s \preceq cI, \quad \forall s \in \{1,2,...,S\} \\
& (c-\epsilon_1)I \preceq \sum_{s=1}^{S}\mathcal{D}_s \preceq (c+\epsilon_2)I, \\
& -\eta I \preceq \sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j \preceq \eta I, \quad \forall l,\ \forall s,
\end{aligned} \qquad (15)$$

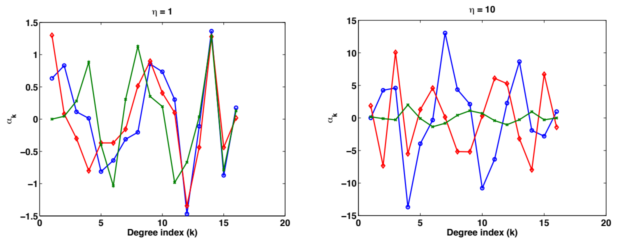
where $\mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_S]$, x_m corresponds to column m of the coefficient matrix X, T_0 is the sparsity level of the coefficients of each signal, and I is the identity matrix. The additional constraints control the spectral characteristics of the kernels as discussed in [4]. The optimization problem (15) is not convex, but it can be approximately solved in a computationally efficient manner by alternating between the sparse coding and dictionary update steps. The sparse coding step can be solved using orthogonal matching pursuit (OMP) [13]. The polynomial coefficient update step is a quadratic program that can be solved using interior point methods [14].

We notice that the parameter η is a design parameter that, from Eq. (14), should be chosen inversely proportional to the quantization stepsize. In particular, a small η penalizes the propagation of the quantization noise when the dictionary is applied in distributed settings with rate constraints, at the cost however 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. On the other hand, a large η gives more flexibility to the algorithm



(a) Original kernels (b) $\eta = 0.1$ (c) $\eta = 1$ (d) $\eta = 10$

Fig. 1. Illustration of the kernels recovered for different values of η.



(a) $\eta = 1$ (b) $\eta = 10$

Fig. 2. Polynomial coefficient values, for each of the kernels, for different values of η.

to learn a set of coefficients that are good for approximating the kernels and thus the signals at infinite bit rate, without penalizing the propagation of the quantization error. The effect of this tradeoff is studied in the experimental section.

VI. Experimental results

We first study the performance of our algorithm for the approximation of synthetic signals. 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 $W(i,j) = e^{-\frac{[dist(i,j)]^2}{2\theta^2}}$ if the physical distance $dist$ between vertices i and j is less than or equal to κ, and zero otherwise. We fix $\theta = 0.04$ and $\kappa = 0.09$ in our experiments, and ensure that the graph is connected. In our first set of experiments, to construct a set of synthetic training signals consisting of localized patterns on the graph, we use a generating 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 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 the training signals for different values of the parameter η.

First, we study the effect of the parameter η in the learned kernels. In Fig. 1(a), we illustrate the original kernels of the underlying dictionary, and in Figs. 1(b)-1(d), we plot the ones recovered by solving the dictionary learning algorithm for different values of η. As expected, we observe that as we increase the value of η, the recovered kernels become closer to the original ones. The effect of the parameter η is more obvious in Fig. 2, where we plot the values of the polynomial coefficients. We observe that, when η is small, the polynomial coefficients are relatively small in magnitude. As a result, the algorithm is not able to capture relatively complicated kernels.

In the next set of experiments, we study the approximation performance of the graph signals with the obtained kernels. We focus first in centralized settings, without rate constraints. We approximate 1000 testing signals, generated in the same way as the training, by computing the sparse approximation in the learned dictionaries with OMP, for different sparsity levels. For comparison, we compute also the approximation performance achieved by sparsely approximating the signals in the spectral graph wavelet dictionary [2]. The obtained results are illustrated in Fig. 3. 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 performance over all the testing signals. As we reduce the values of the parameter η, the approximation performance in the ideal scenario of infinite bit rate deteriorates significantly. It becomes closer and in some cases even worse than the one achieved with the spectral graph wavelet dictionary [2], that is not adapted to the training signals. This behavior is consistent with the conclusion drawn from Figs. 1, 2. 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.

Next, we move to the distributed approximation of the testing signals using Eq. (12). We assume that the messages exchanged by the sensors are uniformly quantized before transmission. In particular, for each message we send one bit for the sign and quantize the magnitude. For each signal y, the quantization range of the transmitted messages is defined to be $[0, \|y\|_\infty]$, and 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. Each value of κ corresponds to different sparsity levels. In Fig. 4, 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 κ. Interestingly, we observe that the best representation performance is obtained when η is very small. As we increase η, the effect of the quantization noise becomes significantly high, which leads to a dramatically

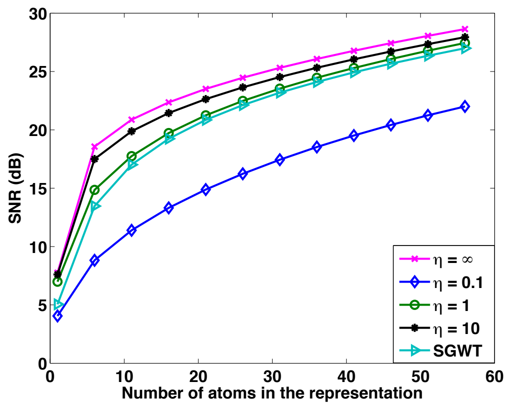


Fig. 3. Approximation performance (SNR) versus sparsity level, achieved with the polynomial graph dictionary for different values of η in centralized settings.

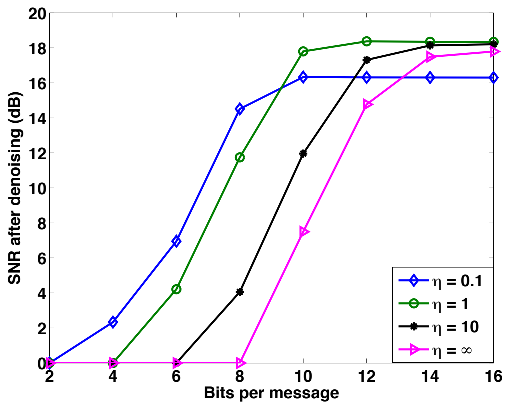


Fig. 5. Denoising performance (in dB) versus bits per message using dictionaries learned with different values of η. The initial SNR is 10 dB.

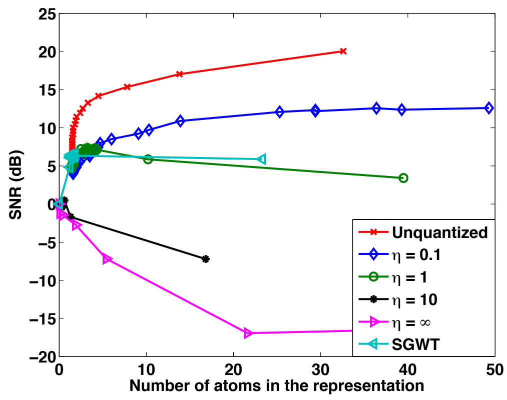


Fig. 4. Approximation performance (SNR) versus sparsity level, achieved with the polynomial graph dictionary for different values of η in distributed settings, with a bit rate of 6 bits per message.

low SNR. The worst performance is obtained when $\eta = \infty$ and the robustness constraint is ignored. This confirms that the imposed constraint can indeed reduce the effect of the quantization noise.

Furthermore, we apply our dictionaries in 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. In Fig. 5, we illustrate the SNR obtained after denoising with different numbers of bits per messages, for each of the learned dictionaries. For comparison, we add also the result achieved by denoising the signals under ideal communication constraints, with a polynomial graph dictionary learned with [4] (i.e., $\eta = \infty$). The obtained results indicate that a small η at low bit rate can bring significant gain in terms of denosing performance. Of course, when the bit rate is high, the denoising performance obtained with the dictionary corresponding to a small η ($\eta = 0.1$) saturates to a small SNR value. These results are consistent with the one obtained in the previous experiments.

Finally, we study the application of the proposed framework to the denoising of real world signals. We consider the daily bottlenecks in San Francisco county. These are parts of the Caltrans Performance Measurement System (PeMS) dataset that provides traffic information throughout all major metropolitan areas of California [15].[1] It contains measurements of 75 detector stations between January 2007 and August 2014. 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, 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, which is normalized for computational issues.

We use half of the signals to learn a polynomial dictionary with $S = 3$, and a maximum polynomial degree of $K = 15$. The sparsity level in the learning phase is set to $T_0 = 5$. We run the learning algorithm for different values of the parameter $\eta = [0.1,\ 1,\ 10,\ \infty]$. The obtained dictionaries are then used for denoising the rest of the signals in distributed settings, at different bit rates. The results are illustrated in Fig. 6. As in the case of the synthetic data, we observe at low bit rate a significant gain when the parameter η is small.If we reduce the value of η even further though, we expect the performance at high bit rate to degrade significantly and saturate to an SNR value that is smaller than the one obtained for $\eta \to \infty$, similarly to Fig. 5. Finally, as in the synthetic experiments, when η becomes large denoising of the signals by 2 dB requires approximately 8 bits per message, which is much higher than the case of small η.

VII. Conclusions

We have studied the effect of quantization in distributed graph signal processing with polynomial dictionary operators. We have shown that the performance is dependent on the graph geometry and on the dictionary structure. We have

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

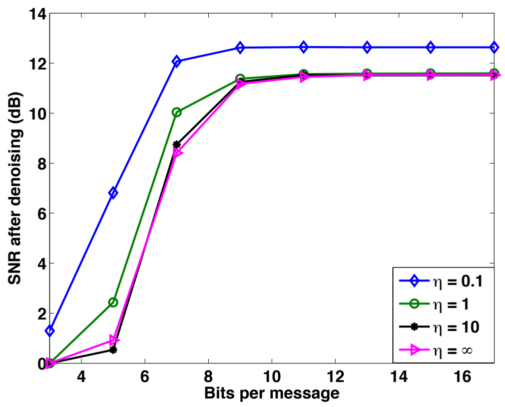


Fig. 6. Denoising performance (in dB) versus bit rate per message for the traffic dataset using dictionaries learned with different values of η. The initial SNR is 10 dB.

then proposed an algorithm that learns graph dictionaries to sparsely approximate graph signals while staying robust to quantization noise. Experimental results have illustrated the tradeoffs between effective distributed signal representation in low bit rate communication settings and the accuracy of the signal approximation in ideal settings. Our future work will focus on the analysis of the distributed processing performance with loss of information as it appears in the practical networks, as well as on the design of adaptive quantization algorithms to yet improve the distributed graph signal processing performance.

Appendix

Proof of Lemma 1: We bound the norm of the error as follows. For ease of notation, we ignore the iteration index.

$$\begin{aligned}\|e\| &= \|E(\mathcal{D}^T y) + \mathcal{D}^T E(\mathcal{D}x) + error(\mathcal{D}^T\mathcal{D}x)\|\\ &\le \|E(\mathcal{D}^T y)\| + \|\mathcal{D}^T E(\mathcal{D}x)\| + \|E(\mathcal{D}^T\mathcal{D}x)\|\\ &\le \sum_{s=1}^{S} \|E(\mathcal{D}_s^T y)\| + \|\mathcal{D}_s^T E(\mathcal{D}x)\| + \|E(\mathcal{D}_s^T\mathcal{D}x)\|, \quad (16)\end{aligned}$$

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

$$\|E(\mathcal{D}_s^T y)\| = \|\sum_{l=0}^{K-1}\Big[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j\Big]\epsilon_l\|$$

$$\le \sum_{l=0}^{K-1}\|\Big[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j\Big]\epsilon_l\| \le \sum_{l=0}^{K-1}\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j\|\|\epsilon_l\|. \quad (17)$$

Similarly, the second and the third term can be written as

$$\|\mathcal{D}_s^T E(\mathcal{D}x)\| = \|\sum_{l=0}^{K-1}\Big[\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j\Big]\xi_l\|$$

$$\le \sum_{l=0}^{K-1}\|\sum_{j=1}^{K-l}\alpha_{s(l+j)}\mathcal{L}^j\|\|\xi_l\|, \quad (18)$$

and

$$\|E(\mathcal{D}_s^T\mathcal{D}x)\| = \|\sum_{k=0}^{K}\alpha_{sk}\mathcal{L}^k\sum_{s'=1}^{S}\sum_{l'=0}^{K-1}\Big[\sum_{j'=1}^{K-l'}\alpha_{s'(l'+j')}\mathcal{L}^{j'}\Big]\zeta_{s',l'}\|$$

$$\le \sum_{s'=1}^{S}\sum_{l'=0}^{K-1}\|\sum_{k=0}^{K}\alpha_{sk}\mathcal{L}^k\|\|\sum_{j'=1}^{K-l'}\alpha_{s'(l'+j')}\mathcal{L}^{j'}\|\|\zeta_{s',l'}\|$$

$$\le c\sum_{s'=1}^{S}\sum_{l'=0}^{K-1}\|\sum_{j'=1}^{K-l'}\alpha_{s'(l'+j')}\mathcal{L}^{j'}\|\|\zeta_{s',l'}\|, \quad (19)$$

where the last inequality comes from the constraint $0I \preceq \mathcal{D}_s \preceq cI$ in the optimization problem (15). Combining Eqs. (16), (17), (18), (19), and using the assumption that the quantization noise is uniformly distributed with magnitude smaller than $\Delta/2$, we obtain the upper bound of (13). $\square$

References

[1] D. I Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[2] D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2010.

[3] X. Zhang, X. Dong, and P. Frossard, "Learning of structured graph dictionaries," in *Proc. IEEE Int. Conf. Acc., Speech, and Signal Process.*, Mar. 2012, pp. 3373 – 3376.

[4] D. Thanou, D. I Shuman, and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3849–3862, Aug. 2014.

[5] S. Segarra, A. G. Marques, G. Leus, and A. Ribeiro, "Reconstruction of graph signals through percolation from seeding nodes," *Submitted to IEEE Trans. Signal Process.*, 2015.

[6] D. I Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev polynomial approximation for distributed signal processing," in *Proc. Int. Conf. Distr. Comput. Sensor Sys.*, June 2011.

[7] D. I Shuman, P. Vandergheynst, and P. Frossard, "Distributed signal processing via Chebyshev polynomial approximation," *ArXiv: http://arxiv.org/pdf/1111.5239.pdf*, Nov. 2011.

[8] D. I Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *arXiv: http://arxiv.org/abs/1307.5708*, 2013.

[9] D. I Shuman, M. J. Faraji, and P. Vandergheynst, "Semi-supervised learning with spectral graph wavelets," in *Proc. of the Int. Conf. on Sampling Theory and Applications*, May 2011.

[10] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, Aug. 1999.

[11] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Img. Sci.*, vol. 2, no. 1, pp. 183–202, Mar. 2009.

[12] S. Sra, "Scalable nonconvex inexact proximal splitting," in *Adv. Neural Inf. Process. Syst.*, 2012, pp. 530–538.

[13] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inform. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[14] S. Boyd and L. Vandenberghe, *Convex Optimization*, New York: Cambridge University Press, 2004.

[15] T. Choe, A. Skabardonis, and P. P. Varaiya, "Freeway performance measurement system (PeMS): an operational analysis tool," in *Annual Meeting of Transportation Research Board*, Jan. 2002.