Autoregressive Moving Average Graph Filter Design

Liu, Jiani; Isufi, E.; Leus, G.

Publication date
2016

Document Version
Accepted author manuscript

Published in
Proceedings of the 37th WIC Symposium on Information Theory in the Benelux and The 6th Joint WIC/IEEE Symposium on Information Theory and Signal Processing in the Benelux

Citation (APA)
Liu, J., Isufi, E., & Leus, G. (2016). Autoregressive Moving Average Graph Filter Design. In F. Glineur, & J. Louveaux (Eds.), *Proceedings of the 37th WIC Symposium on Information Theory in the Benelux and The 6th Joint WIC/IEEE Symposium on Information Theory and Signal Processing in the Benelux* (pp. 219-226). Université Catholique de Louvain, Belgium.

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Autoregressive Moving Average Graph Filter Design

Jiani Liu Elvin Isufi Geert Leus
Delft University of Technology
Faculty of EEMCS
2826 CD Delft, The Netherlands
j.liu-1@tudelft.nl e.isufi-1@tudelft.nl g.j.t.leus@tudelft.nl

Abstract

To accurately match a finite-impulse response (FIR) graph filter to a desired response, high filter orders are generally required leading to a high implementation cost. Autoregressive moving average (ARMA) graph filters can alleviate this problem but their design is more challenging. In this paper, we focus on ARMA graph filter design for a known graph. The fundamental aim of our ARMA design is to create a good match to the desired response but with less coefficients than a FIR filter. Our design methods are inspired by Prony's method but using proper modifications to fit the design to the graph context. Compared with FIR graph filters, our ARMA graph filters show better results for the same number of coefficients.

1 Introduction

Graph signal processing (GSP) extends classical digital signal processing to signals connected with the topology of a graph [1], [2]. More and more concepts and tools from classical signal processing are transferred to the field of GSP, including the uncertainty principle [3], graph wavelets [4], graph signal classification [5], graph signal recovery [6],[7], and graph signal sampling [8].

Depending on the definition of the graph frequency and the graph Fourier transform (GFT) [1], [2], many different kinds of graph filters have been designed as linear operators acting upon a graph signal. Similar to traditional digital signal processing, graph filters amplify or attenuate the graph signal at different graph frequencies. Graph filters have been used for many signal processing applications [9]. Traditionally, graph filters are expressed as a polynomial in the so-called graph shift matrix (e.g., the adjacency matrix, the graph Laplacian, or any of their modifications), resulting in a so-called finite-impulse response (FIR) graph filter [10], [12]. However, to accurately match some given filter specifications, FIR filters require high filter orders leading to a high implementation cost.

An alternative filter approach is the so-called infinite impulse response (IIR) graph filter [13]. Compared with FIR graph filters, characterized by a polynomial frequency response, IIR graph filters have a rational polynomial response, which brings more flexibility to their design. As such, IIR graph filters have the potential to fit complicated filter specifications with small degrees. As a particular instance of IIR graph filters, autoregressive moving average (ARMA) graph filters have been introduced in [14] [15]. ARMA graph filters were introduced as universal filters that do not depend on the particular topology of the graph, although in this paper we will design them for a particular graph in mind. Note that ARMA graph filters are capable of not only shaping the graph signal in the graph frequency domain, but also in the regular temporal frequency domain, in case the graph signal is time varying [15]. And although our

proposed designs can be used in that context, we will only focus on the graph domain in this paper.

Although ARMA graph filters show great promises, their design is challenging. This paper tries to tackle this issue for known graphs, which means that we only have to match the ARMA graph filter to the desired response in a set of known frequencies. We will present two design procedures, both inspired by Prony's method. The first design transforms the problem from the graph frequency domain to the coefficient domain (as done in the classical Prony's method). However, in contrast to the regular time domain, in the graph domain, this transformation is generally not optimal and does not lead to a good solution. Our second design shows that staying in the graph frequency domain preserves the optimality of the solution and also simplifies the problem. We conclude the paper by showing some simulation results for our two ARMA graph filter design procedures based on a known graph. They illustrate that in some cases, the FIR filter can be outperformed for the same number of filter coefficients.

2 FIR Graph Filter

Consider an undirected graph $G = (V, E)$, where V is a set of N nodes and E is the set of edges. We indicate with $\mathbf{M}$ the symmetric graph shift matrix, which could be the adjacency matrix, the graph Laplacian or any of their modifications. An eigenvalue decomposition of $\mathbf{M}$ leads to $\mathbf{M} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{U}^T$ is the graph Fourier transform (GFT) matrix and $\mathbf{\Lambda}$ is a diagonal matrix with on the diagonal, the graph frequencies λ_n.

A graph filter $\mathbf{G}$ is a linear operator that acts upon a graph signal $\mathbf{x}$, leading to the output $\mathbf{y} = \mathbf{G}\mathbf{x}$. A finite impulse response (FIR) graph filter of order K, or FIR(K) in short, can be expressed as a K-th order polynomial in $\mathbf{M}$:

$$\mathbf{G} = g(\mathbf{M}) = \sum\nolimits_{k=0}^{K} g_k \mathbf{M}^k, \tag{1}$$

where g_k are the FIR filter coefficients. This means that $\mathbf{G}$ is diagonizable by the GFT matrix $\mathbf{U}^T$, i.e., $\mathbf{U}^T\mathbf{G}\mathbf{U} = g(\mathbf{\Lambda})$, and the filter indeed reshapes the spectrum of the input:

$$\hat{\mathbf{y}} = \mathbf{U}^T\mathbf{y} = \mathbf{U}^T\mathbf{G}\mathbf{U}\mathbf{U}^T\mathbf{x} = g(\mathbf{\Lambda})\hat{\mathbf{x}},$$

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ represent that GFT of the input and output signal, respectively. The frequency response of the filter at frequency λ_n, $n = 1, 2, \ldots, N$, is thus given by

$$\hat{g}_n = g(\lambda_n) = \sum_{k=0}^{K} g_k \lambda_n^k. \tag{2}$$

Define now the $N \times (K+1)$ Vandermonde matrix $\mathbf{\Psi}_{K+1}$ as the $N \times (K+1)$ matrix with entries $[\mathbf{\Psi}]_{n,k} = \lambda_n^{k-1}$. Then stacking the filter coefficients g_k in the $(K+1) \times 1$ vector $\mathbf{g}$ and the frequency response $\hat{g}_n$ in the $N \times 1$ vector $\hat{\mathbf{g}}$, we have

$$\hat{\mathbf{g}} = \mathbf{\Psi}_{K+1}\mathbf{g}. \tag{3}$$

From [11] we know that if the graph is known and if the graph frequencies are distinct, the filter coefficients can be computed as

$$\mathbf{g} = \mathbf{\Psi}_{K+1}^{\dagger}\hat{\mathbf{g}}, \tag{4}$$

where $\mathbf{A}^\dagger$ is the pseudo-inverse of the matrix $\mathbf{A}$.

It is well known (and clear from (3) and (4)) that for a known graph of N nodes with N distinct graph frequencies, a finite impulse response (FIR) graph filter of order $K = N - 1$ can exactly represent any desired response. In that case, the Vandermonde matrix $\mathbf{\Psi}_{K+1} = \mathbf{\Psi}_N$ is a square invertible matrix. However, for large (i.e., more than 100 nodes) graphs, which are regularly encountered in real applications (e.g., a temperature prediction system containing hundreds of cities), the computation of the FIR filter coefficients could be numerically infeasible because of the ill-conditioning of the related system matrix $\mathbf{\Psi}_N$. This issue is usually resolved by reducing the order of the FIR filter below the number of nodes of the graph ($K < N - 1$), at the cost of a reduced accuracy. But for large graphs, this still leads to large FIR filter orders which are costly to implement. In this work, we introduce another approach to resolve these problems of FIR filters.

3 ARMA Graph Filter

In order to achieve a better accuracy with a smaller order filter, we apply an autoregressive moving average (ARMA) filter to the signal living on the known graph G. Note that for an ARMA filter, it has been shown [14] that working with a translated version of the normalized graph Laplacian leads to the best stability conditions, but we will make abstraction of this here and simply work with a general shift matrix $\mathbf{M}$. From now on, we also assume that all graph frequencies λ_n are distinct.

For an ARMA(P,Q) graph filter, the graph frequency response at frequency λ_n, $n = 1, 2, \ldots, N$, can be written as

$$\hat{g}_n = g(\lambda_n) = \frac{\sum_{q=0}^{Q} b_q \lambda_n^q}{1 + \sum_{p=1}^{P} a_p \lambda_n^p}. \tag{5}$$

For future use, we also define $\mathbf{a} = [1, a_1, \ldots, a_p]^T$ and $\mathbf{b} = [b_0, b_1, \ldots, b_q]^T$ as the ARMA filter coefficients. Note that because the graph and thus the graph frequencies are known, we can always write $\hat{g}_n$ for all $n = 1, 2, \ldots, N$ as an $(N-1)$-th order polynomial in λ_n with fixed coefficients:

$$\hat{g}_n = \sum_{k=0}^{N-1} g_k \lambda_n^k, \tag{6}$$

or in other words, the ARMA(P,Q) graph filter can always be written as an FIR($N-1$) graph filter if the graph is known. As before, the FIR filter coefficients g_k can be stacked in the $N \times 1$ vector $\mathbf{g}$ and the frequency response $\hat{g}_n$ in the $N \times 1$ vector $\hat{\mathbf{g}}$. The relation between $\hat{\mathbf{g}}$ and $\mathbf{g}$ is then given by $\hat{\mathbf{g}} = \mathbf{\Psi}_N \mathbf{g}$, where $\mathbf{\Psi}_N$ is defined as before but with K replaced by $N - 1$.

Now assume that we are given a prescribed frequency response $\hat{h}_n$, which can again be related to a set of N FIR filter coefficients h_k through

$$\hat{h}_n = h(\lambda_n) = \sum_{k=0}^{N-1} h_k \lambda_n^k. \tag{7}$$

As before, we respectively stack h_k and $\hat{h}_n$ into $\mathbf{h}$ and $\hat{\mathbf{h}}$, which are related by $\hat{\mathbf{h}} = \mathbf{\Psi}_N \mathbf{h}$. The problem statement in this work is then to find the ARMA coefficients $\mathbf{a}$ and $\mathbf{b}$, resulting in a frequency response $\hat{\mathbf{g}}$ that best represents the desired frequency response $\hat{\mathbf{h}}$.

4 ARMA Filter Design

Optimally, we would try to minimize the error between $\hat{\mathbf{g}}$, which is parameterized by $\mathbf{a}$ and $\mathbf{b}$, and $\hat{\mathbf{h}}$, or in other words, we would try to solve

$$\min_{\mathbf{a},\mathbf{b}} \sum\nolimits_{n=1}^{N} \left| \hat{h}_n - \frac{\sum_{q=0}^{Q} b_q \lambda_n^q}{1 + \sum_{p=1}^{P} a_p \lambda_n^p} \right|^2 . \tag{8}$$

But since that is a difficult problem to solve, as in Prony's method, we choose to solve the related (yet not equivalent) problem

$$\min_{\mathbf{a},\mathbf{b}} \sum\nolimits_{n=1}^{N} \left| \hat{h}_n \left(1 + \sum\nolimits_{p=1}^{P} a_p \lambda_n^p\right) - \sum\nolimits_{q=0}^{Q} b_q \lambda_n^q \right|^2 . \tag{9}$$

Following a similar idea as in Prony's method, we use (7) to expand $\hat{h}_n$ in (9), resulting in

$$\min_{\mathbf{a},\mathbf{b}} \sum\nolimits_{n=1}^{N} \left| \left(\sum\nolimits_{k=0}^{N-1} h_k \lambda_n^k\right) \left(1 + \sum\nolimits_{p=1}^{P} a_p \lambda_n^p\right) - \sum\nolimits_{q=0}^{Q} b_q \lambda_n^q \right|^2 .$$

Using some simple algebra, this can be transformed into the following least squares (LS) problem, written in matrix form as

$$\min_{\mathbf{a},\mathbf{b}} \|\mathbf{\Psi}_{N+P}\mathbf{H}\mathbf{a} - \mathbf{\Psi}_{Q+1}\mathbf{b}\|^2, \tag{10}$$

where $\mathbf{H}$ is the $(N+P) \times (P+1)$ Toeplitz matrix expressed by

$$\mathbf{H} = \begin{bmatrix} h_0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ h_{N-1} & \dots & h_0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & h_{N-1} \end{bmatrix} .$$

Note that the two terms in this LS problem are nothing more than the frequency response of the concatenation of the filter $\mathbf{h}$ and $\mathbf{a}$ on the left and the frequency response of the filter $\mathbf{b}$ on the right. More specifically, defining $\hat{\mathbf{a}} = \mathbf{\Psi}_{P+1}\mathbf{a}$ and $\hat{\mathbf{b}} = \mathbf{\Psi}_{Q+1}\mathbf{b}$ as the frequency responses of the graph filters $\mathbf{a}$ and $\mathbf{b}$, we can also write (10) as

$$\min_{\mathbf{a},\mathbf{b}} \|\hat{\mathbf{h}} \circ \hat{\mathbf{a}} - \hat{\mathbf{b}}\|^2, \tag{11}$$

where $\circ$ represents the element-wise Hadamard product.

Again, following Prony's approach, we now try to transform the problem (10) from the frequency domain to the filter coefficient domain. This can be done by observing that the matrices $\mathbf{\Psi}_{N+P}$ and $\mathbf{\Psi}_{Q+1}$ can both be written as a function of the frequency transformation matrix $\mathbf{\Psi}_N$ as

$$\mathbf{\Psi}_{N+P} = \mathbf{\Psi}_N[\mathbf{I}_N, \mathbf{T}], \quad \mathbf{\Psi}_{Q+1} = \mathbf{\Psi}_N \begin{bmatrix} \mathbf{I}_{Q+1} \\ \mathbf{0}_{(N-Q-1)\times(Q+1)} \end{bmatrix},$$

where $\mathbf{T}$ is some $N \times P$ transformation matrix that can be computed from the first equation since both $\mathbf{\Psi}_N$ and $\mathbf{\Psi}_{N+P}$ are known. So defining

$$\bar{\mathbf{H}} = [\mathbf{I}_N, \mathbf{T}]\mathbf{H}, \quad \bar{\mathbf{b}} = \begin{bmatrix} \mathbf{I}_{Q+1} \\ \mathbf{0}_{(N-Q-1)\times(Q+1)} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0}_{(N-Q-1)\times 1} \end{bmatrix},$$

we can rewrite (10) as

$$\min_{\mathbf{a},\mathbf{b}} \|\mathbf{\Psi}_N \bar{\mathbf{H}}\mathbf{a} - \mathbf{\Psi}_N \bar{\mathbf{b}}\|^2, \tag{12}$$

Multiplying both terms with $\mathbf{\Psi}_N^{-1}$ (note that this corresponds to a loss of optimality), we finally transform the problem from the frequency domain to the filter coefficient domain:

$$\min_{\mathbf{a},\mathbf{b}} \|\bar{\mathbf{H}}\mathbf{a} - \bar{\mathbf{b}}\|^2. \tag{13}$$

Since $\bar{\mathbf{b}}$ has $N-Q-1$ zeros at the bottom, we can use the bottom $N-Q-1$ equations of (13) to solve for $\mathbf{a}$. The vector $\mathbf{b}$ can then be estimated using the top $Q+1$ equations of (13). Alternatively, we can use (12) (or (10)) after plugging in the estimate for $\mathbf{a}$.

In the time domain, this works well since $\mathbf{\Psi}_N$ then simply is the discrete Fourier transform (DFT) matrix, which is well-conditioned and even unitary up to a scale. In the general graph domain, however, $\mathbf{\Psi}_N$ can be very badly conditioned, especially for large graphs. So computing the desired filter coefficients h_k (and thus the matrix $\mathbf{H}$) as well as the transformation matrix $\mathbf{T}$, is numerically infeasible. Moreover, since $\mathbf{\Psi}_N$ is generally far from a scaled unitary matrix, the problems (12) and (13) are far from equivalent. Hence, we need to tackle this problem in a different way.

The basic idea is not to expand $\hat{h}_n$ in (9), and to rewrite it in matrix form as

$$\min_{\mathbf{a},\mathbf{b}} \|[\mathbf{\Psi}_{P+1} \circ (\hat{\mathbf{h}} \otimes \mathbf{1}_{1\times(P+1)})]\mathbf{a} - \mathbf{\Psi}_{Q+1}\mathbf{b}\|^2, \tag{14}$$

where $\otimes$ represents the Kronecker product. Then, instead of trying to move from the frequency domain to the filter coefficient domain, in order to exploit the finite order of $\mathbf{b}$, we simply project out this term using the orthogonal projection matrix

$$\mathbf{P}^{\perp}_{\mathbf{\Psi}_{Q+1}} = \mathbf{I}_N - \mathbf{\Psi}_{Q+1}\mathbf{\Psi}^{\dagger}_{Q+1}, \tag{15}$$

which is generally well-conditioned. Staying in the frequency domain not only preserves the optimality of the LS problem, but also simplifies the solution procedure. By multiplying the objective function in (14) with $\mathbf{P}^{\perp}_{\mathbf{\Psi}_{Q+1}}$ we obtain the *equivalent* problem

$$\min_{\mathbf{a}} \|\mathbf{P}^{\perp}_{\mathbf{\Psi}_{Q+1}}[\mathbf{\Psi}_{P+1} \circ (\hat{\mathbf{h}} \otimes \mathbf{1}_{1\times(P+1)})]\mathbf{a}\|^2, \tag{16}$$

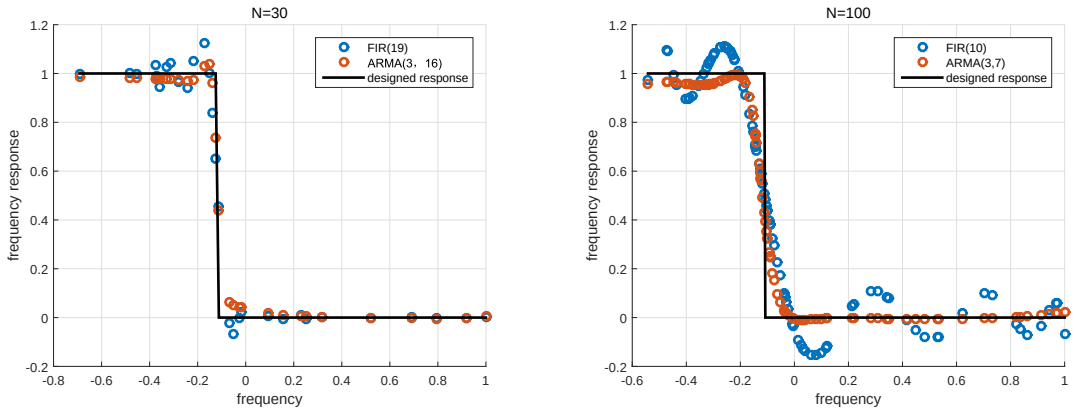
which can be used to solve for $\mathbf{a}$. The vector $\mathbf{b}$ can be estimated using (14) after plugging in the estimate for $\mathbf{a}$.

5 Numerical evaluation

Similar to [15], for our numerical simulations, we will adopt the translated normalized Laplacian as the symmetric graph shift matrix $\mathbf{M} = \mathbf{I} - \mathbf{L}_n$, where $\mathbf{L}_n$ is the normalized graph Laplacian. Due to Sylvester's matrix theorem, the eigenvalues of $\mathbf{M}$ which can be seen as shifted graph frequencies are within (-1,1]. As shown in [15], such a choice

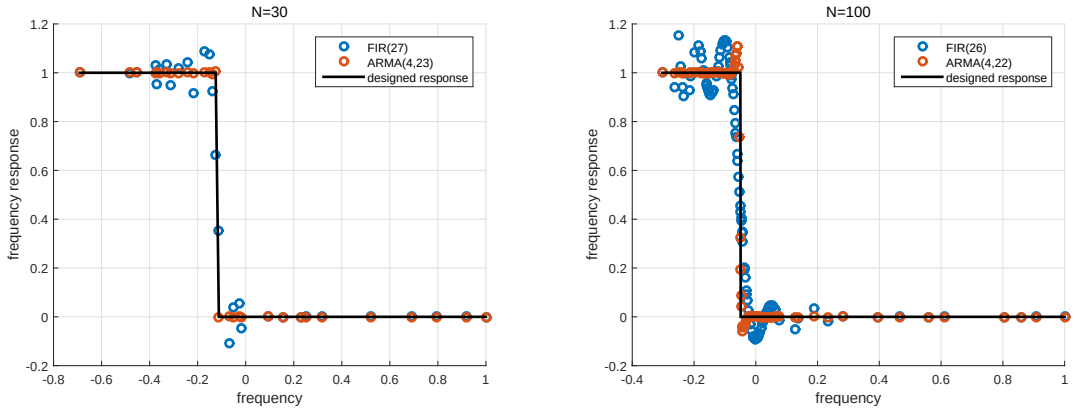
does not change the filtering and improves the stability region of the ARMA filters in case they are implemented as in [14], [15].

The fundamental aim of our ARMA filter design is to create a good match to the desired response, but with less coefficients than an FIR filter with a similar match or to obtain a better match than an FIR filter with the same number of coefficients. For small graphs (N=20), the Vandermonde matrix is well-conditioned. Thence, an FIR filter with order $N-1$ can perfectly match any desired response. And also FIR filters with smaller orders work well in that case. In general, we observed that for small graphs, an FIR filter generally outperforms an ARMA filter with the same number of coefficients. However, for larger graphs, this is not always true. To test our ARMA filter design methods, we generate two scenarios with connected graphs by randomly placing (i) 30 nodes and (ii) 100 nodes in a squared area, where two nodes are neighbors if and only if they are close enough to each other (we choose 20% of the maximum distance as the condition for selecting neighbors). We test both design methods. The first one is based on (13), whereas the second one is based on (16).



(a) 30 nodes (b) 100 nodes

Figure 1: ARMA filter design method (13) for a 30-node and 100-node graph.



(a) 30 nodes (b) 100 nodes

Figure 2: ARMA filter design (16) for a 30-node and 100-node graph.

In Figure 1, we adopt the first method and show a comparison between the ARMA filter and FIR filter for the same number of coefficients. For a graph with 30 nodes,

an FIR filter (28) is applied for the first step. We choose the order of the FIR filter close to the number of nodes which gives a good performance for this graph. The ARMA filter design starts by fixing a specific order $P \ll N$ and then searching for the Q that leads to the best fit. Only the number of filter orders Q smaller than $K - P$ need to be investigated. The MSE between the desired response and ARMA filter response is used as evaluation index. Given the initial condition $P = 3$, the best performance comes from ARMA (3, 16). For this graph with 30 nodes, the ARMA filter clearly outperforms the FIR filter. The performance comparison of the ARMA filter with the FIR filter for larger graphs (100 nodes) depends on the selection of P and Q. Even though the ARMA filter also seems to outperform the FIR filter for 100 nodes, the ARMA filter shows errors. This is due to the fact that the first method loses optimality when we go from the frequency domain to the filter coefficient domain.

In Figure 2, we focus on the second design method (i.e., (16)), which is appropriate for graphs of any size. We apply our second method on the same graphs as before. As for the first method, we pick an order $P \ll N$ and search for Q. Then we compare the resulting ARMA design with a FIR filter of order $P + Q$. For the graph with 30 nodes, the ARMA filter almost matches perfectly. For the larger graph of 100 nodes, the performance of the ARMA filter only shows a mismatch around the cutoff frequency. Clearly, this ARMA filter design shows a better performance than the FIR filter design with the same number of coefficients.

The two methods only seem to make sense here if Q is larger than P. However, the stability of the ARMA filter (for an implementation according to [14], [15]) depends on the initial P and the construction of the graph. For the same graph, when we change the order P, the best performing ARMA filter may lose stability. Also, for different graphs and with the same denominator order P the stability is not guaranteed.

6 Conclusion

In this work, we have considered the design of ARMA graph filters when the graph structure is known. We have seen that Prony's method cannot be directly applied but needs some proper modifications in the graph context. The proposed ARMA filter design outperforms a same-order FIR filter in large graphs with a relatively small number of filter coefficients. Even though in different scenarios the designed filter leads to stable implementations according to [14], [15], stability is not guaranteed. Future work will be based on stable ARMA filter design.

References

[1] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83 - 98, May. 2013.

[2] A. Sandryhaila and J. M. Moura, "Discrete Signal Processing on Graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644-1656, Jan. 2013.

[3] Agaskar and Y. M. Lu, "A Spectral Graph Uncertainty Principle, " *IEEE Transactions Information Theory*, vol.59, no. 7,pp. 4338-4356, July 2013.

[4] D. K. Hammond, and P. Vandergheynst, "Wavelets on graphs via spectral graph theory", *Applied and Computational Harmonic Analysis* , vol. 30, no. 2, pp. 129-150, Mar. 2011.

[5] A. Sandryhaila, and J. M. F. Moura, "Classification via regularization on graphs", in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Austin, TX, pp. 495 - 498, Dec. 2013.

[6] S. Chen, A. Sandryhaila, and J. M. F. Moura, "Signal recovery on graph: variation minimization ," *IEEE Transactions on Signal Processing*, vol. 63, no. 17, pp. 4609 - 4624, Jun. 2015.

[7] S. Chen, R. Varma, and A. Singh, "Signal recovery on graphs: Random versus experimentally designed sampling", in *IEEE, Sampling Theory and Applications (SampTA), 2015 International Conference*, Washington, DC, pp. 337 - 341, May 2015.

[8] A. Anis, A. Gadde, and A. Ortega, "Toward a sampling theory for signals on arbitrary graphs", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, pp. 3864 - 3868, May. 2014.

[9] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovacevic, "Signal Denoising on Graphs via Graph Filtering", in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Atlanta, GA, USA, pp. 872-876, Dec. 2014.

[10] A. Sandryhaila, S. Kar, and J. M. F. Moura, "Finite-time distributed consensus through graph filters," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, pp. 1080-1084, May. 2014.

[11] A Sandryhaila, JMF Moura, "Discrete signal processing on graphs: Graph filters", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, BC, pp. 6163 - 6166, May. 2013.

[12] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis", *IEEE Transactions Signal Processing*, vol. 62, no.12, pp. 3042-3054, May. 2014.

[13] X. Shi, H. Feng, M. Zhai, T. Yang, and B. Hu, "Infinite impulse response graph filters in wireless sensor networks", *IEEE Signal Processing Letters*, vol. 22, no. 8, pp. 1113 - 1117, Jan. 2015.

[14] A. Loukas, A. Simonetto, and G. Leus, "Distributed Autoregressive Moving Average Graph Filters", *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1931 - 1935, Jun. 2015.

[15] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Distributed Time-Varying Graph Filtering", arXiv preprint, arXiv:1602.04436v1, 2016.

[16] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*, Georgia Institute of Technology, sec. 4.3, pp. 133-144, 1996.