An Example-Driven Introduction to Data Analytics on Graphs

Ljubiša Stanković, Danilo Mandić, Miloš Daković, Ilya Kisil, Ervin Sejdić, Anthony G. Constantinides

I. Scope

Graphs are irregular structures which naturally account for data integrity, however, traditional approaches have been established outside Signal Processing, and largely focus on analyzing the underlying graphs rather than signals on graphs. Given the rapidly increasing availability of multisensor and multinode measurements, likely recorded on irregular or ad-hoc grids, it would be extremely advantageous to analyze such structured data as graph signals and thus benefit from the ability of graphs to incorporate spatial awareness of the sensing locations, sensor importance, and local versus global sensor association. The aim of this lecture note is therefore to establish a common language between graph signals, defined on irregular signal domains, and some of the most fundamental paradigms in DSP, such as spectral analysis of multichannel signals, system transfer function, digital filter design, parameter estimation, and optimal filter design.

This is achieved through a physically meaningful and intuitive real-world example of geographically distributed multisensor temperature estimation. A similar spatial multisensor arrangement is already widely used in Signal Processing curricula to introduce minimum variance estimators and Kalman filters [11], and by adopting this framework we facilitate a seamless integration of graph theory into the curriculum of existing DSP courses. By bridging the gap between standard approaches and graph signal processing, we also show that standard methods can be thought of as special cases of their graph counterparts, evaluated on linear graphs. It is hoped that our approach would not only help to demystify graph theoretic approaches in education and research but it would also empower practitioners to explore a whole host of otherwise prohibitive modern applications.

II. Relevance

While in classic graph signal theory, the graphs are typically given (e.g., in various computer, social, road, transportation, and power networks), the first step in graph signal processing is to employ background knowledge of signal generating mechanisms in order to define the graph as a signal domain. This poses a number of challenges, e.g., while the data sensing points (graph vertices) are usually well defined in advance however, their connectivity (graph edges) is not often available. In other words, the data domain within the graph signal paradigm represents a part of the problem itself, and has to be determined based on the properties of the sensing positions or features of the acquired set of data. All in all, the definition of an appropriate graph structure is a prerequisite for physically meaningful and computationally efficient graph signal processing applications.

Remark 1: In numerous practical settings, the signal domain is determined by equidistant time instants or by a set of spatial sensing points on a uniform grid. However, increasingly the actual data sensing domain may not even be related to the physical dimensions of time and/or space, and typically does not exhibit various forms of regularity. For example, in social or web-related networks, the sensing points and their connectivity pertain to specific objects/nodes and topology of their links. It should be noted that even for the data acquired in well defined time and space domains, the introduction of new relations between the sensing points, through graphs, may yield new insights into the analysis and provide enhanced data processing (e.g., based on local neighborhoods). The advantage of graphs over classical estimators in these settings is that graphs account naturally for irregular spatial sensor placement in the problem definition, together with the corresponding data connectivity in the analysis.

Indeed, Graph Signal and Information Processing is particularly well suited to making sense from data acquired over irregular data domains, which can be achieved, for example, by leveraging intuitions developed on Euclidean domains, or by employing analogies with other irregular domains such as polygon meshes and manifolds. In many emerging applications, e.g., Big Data, this also introduces a number of new challenges:

- Basic concepts must be revisited in order to accommodate structured but often incomplete information,
- New physically meaningful frameworks, specifically tailored for heterogeneous data sources, are required, and
- Trade-offs between performance and numerical requirements are a prerequisite when operating in real-time.

The common language and enhanced intuition between the graph and standard approaches, illuminated in this article through e.g., the relationships between the vertex and time domains, or graph clustering and vertex ordering, may be naturally generalized to address the above challenges and spur further developments in graph signal processing for Big Data.

III. Prerequisites

This Lecture Note assumes a basic knowledge of Linear Algebra and Digital Signal Processing.

IV. History of Graph Theoretic Application

Graph theory, as a branch of mathematics, has existed for almost three centuries. The beginning of graph theory applications in electrical engineering dates back to the mid-XIX century and the definition of Kirchoff’s laws. Owing to their inherent “spatial awareness”, graph models have since become a de facto standard for data analysis across the science and engineering areas, including chemistry, operational research, social networks, and computer sciences.
A systematic account of graph theory as an optimization tool can be attributed to the seminal book by Nicos Christofides of Imperial College London, published in 1975 [1]. Soon after gaining prominence in general optimization, it was very natural to explore the application of graph theory in signal processing and related areas [2]. Indeed, perhaps the first lecture course to teach graph theory to then emerging communication networks and channel coding student cohort was introduced by the author Anthony Constantinides in 1978. This helped to establish and formalize the connections between general optimization and the topology of a communication network, and has spurred further applications in image processing [3].

After a relative lull over the next two decades, current developments in graph theory owe their prominence to the emergence of modern data sources, such as large-scale sensor and social networks, which inherently provide rich underlying physical, social, and geographic structures that require new ways to establish statistical inference [4], [5], [6], [7], [8], [9], [10].

V. PROBLEM STATEMENT: AN ILLUSTRATIVE EXAMPLE

Consider a multi-sensor setup, shown in Fig. 1a) for measuring a temperature field in a known geographical region. The temperature sensing locations are chosen according to the significance of a particular geographic area to local users, with $N = 64$ sensing points in total, as shown in Fig. 1a). The temperature field is denoted by $\{x(n)\}$ and a snapshot of its values is given in Fig. 1b). Each measured sensor signal can then be mathematically expressed as

$$x(n) = s(n) + \epsilon(n),$$

(1)

where $s(n)$ is the true temperature that would have been obtained in ideal measuring conditions and $\epsilon(n)$ comprises the adverse effects of the local environment on sensor readings or faulty sensor activity, and is referred to as “noise” in the sequel. For illustrative purposes, in our study $\epsilon(n)$ was modeled as a realization of white, zero-mean, Gaussian process, with standard deviation $\sigma_\epsilon = 4$. It was added to the signal, $s(n)$, to yield the signal-to-noise ratio in $\{x(n)\}$ of $SNR_0 = 14.2$ dB.

Remark 2: Classical signal processing requires an arrangement of the quintessentially spatial temperature samples in Fig. 1a) into a linear structure shown in Fig. 1b). Obviously, such “lexicographic” ordering is not amenable to exploiting the spatial information related to the actual sensor arrangement, dictated by the terrain. This renders classical analyses of this temperature field inapplicable (or at best suboptimal), as the performance critically depends on the chosen sensor ordering scheme. This exemplifies that even a most routine temperature measurement setup requires a more complex estimation structure than the simple linear one corresponding to the classical signal processing framework, shown in Fig. 1b).

To introduce a “situation-aware” noise reduction scheme for the temperature field in Fig. 1a) we proceed to explore a graph-theoretic framework to this problem, starting from a local signal average operator. In classical Signal Processing this can be achieved through a moving average operator, e.g., through averaging across the neighboring data samples, or equivalently neighboring nodes, in the linear graph in Fig. 1b), and for each sensing point. Physically, such local neighborhood should indeed include close neighboring sensing points but which also exhibit similar meteorological properties defined by the distance, altitude difference, and other terrain properties.

Remark 2: Classical signal processing requires an arrangement of the quintessentially spatial temperature samples in Fig. 1a) into a linear structure shown in Fig. 1b). Obviously, such “lexicographic” ordering is not amenable to exploiting the spatial information related to the actual sensor arrangement, dictated by the terrain. This renders classical analyses of this temperature field inapplicable (or at best suboptimal), as the performance critically depends on the chosen sensor ordering scheme. This exemplifies that even a most routine temperature measurement setup requires a more complex estimation structure than the simple linear one corresponding to the classical signal processing framework, shown in Fig. 1b).

Consider the local neighborhoods for the sensing points $n = 20, 29, 37, and 41$, shown in Fig. 2a). The cumulative temperature for each sensing point is then given by

$$y(n) = \sum_{m = n} \text{at and around } n x(m),$$

so that the local average temperature for a sensing point $n$ may be easily obtained by dividing the cumulative temperature, $y(n)$, with the number of included sensing points. For example, for the sensing points $n = 20$ and $n = 37$, presented in Fig. 2a), the “domain knowledge aware” local estimation takes the form

$$y(20) = x(20) + x(19) + x(22) + x(23)$$

(2)

$$y(37) = x(37) + x(32) + x(33) + x(35) + x(61).$$

(3)
Fig. 2. Temperature sensing setup as a graph signal processing domain. a) Local neighborhood for the sensing points \( n = 20, 29, 37, \) and 41. These neighborhoods are chosen using “domain knowledge” dictated by local terrain and by taking into account the distance and altitude. Neighboring sensors for each of these sensing locations (vertices) are chosen in a physically meaningful way and their relation is indicated by the connectivity lines, called edges. b) Local neighborhoods for all sensing vertices, presented in a graph form.

For convenience, the full set of relations among the sensing points can now be arranged into the matrix form, to give

\[
y = x + Ax,
\]

where the matrix \( A \) indicates the connectivity structure of the neighboring sensing locations that should be involved in the calculation for each \( y(n) \). The matrix \( A \) is therefore referred to as the connectivity or adjacency matrix of a graph. Its elements are either 1 (if the corresponding vertices are related) or 0 (if they are not related). Fig. 2b) shows the sensing locations with the corresponding connectivity for the temperature estimation scenario in Fig. 2a). From (2) we can observe, for example, that the 20th row of the adjacency matrix \( A \) will have all zero elements, except for \( A_{20,19} = 1, A_{20,22} = 1, \) and \( A_{20,23} = 1 \) (for more detail see the electronic supplement).

This simple real-world example can be interpreted within the graph signal processing framework as follows:

- The sensing points where the signal is measured are designated as the graph vertices, see Fig. 1.
- The vertex-to-vertex lines indicating the connectivity among the sensing points are called the graph edges.
- The vertices and edges form a graph, as in Fig. 2b), a new and very structurally rich signal domain.
- The graph, rather than a standard vector of sensing points, is then used for analyzing and processing data, as it is equipped with spatial and physical awareness.
- The measured temperatures are now interpreted as signal samples on graph, as shown in Fig. 3.

Through relation (4), we have therefore introduced a simple graph system for physically and spatially aware signal averaging (a linear first-order graph system).

To emphasize our trust in a particular sensor (i.e., to model sensor relevance), a weighting scheme may be imposed on the edges (connectivity) between the sensing points, in the form

\[
y(n) = x(n) + \sum_{m \neq n} W_{nm}x(m).
\]

The weight \( W_{nm} \) indicates the strength of the coupling between signal values at the sensing points \( n \) and \( m \); its value is zero if the points \( n \) and \( m \) are not related and for \( n = m \). We have now arrived at a weighted graph, whereby each edge has an associated weight, \( W_{nm} \), which adds a “sensor relevance” information to the already established “spatial awareness” modeled by the edges. In our example, a matrix form of a weighted cumulative graph signal now becomes

\[
y = x + Wx.
\]

This equips graph signal models with additional flexibility, for example, in order to produce unbiased estimates, instead of
the cumulative sums in (4) and (5), the weighting coefficients within the estimate for each \(y(n)\) should sum up to 1. This can be achieved through a normalized form of (6), given by

\[
y = \frac{1}{2}(x + D^{-1}Wx),
\]

(7)

where the elements of the diagonal normalization matrix, \(D\), called the degree matrix, are \(D_{nn} = \sum_m W_{nm}\). When this simple normalized first-order system is employed to filter the original noisy signal from Fig. 3, an improvement of 6 dB over the original signal-to-noise ratio, \(SNR_0 = 14.2\ dB\), is achieved.

Another important operator for graph signal processing is the graph Laplacian, \(L\), which is defined as

\[
L = D - W.
\]

**Remark 3:** A graph is fully specified by the set of its vertices and their connectivity scheme (designated by edges). The edges may be defined by the adjacency matrix, \(A\), with \(A_{mn} \in \{0, 1\}\), for unweighted graphs or by the “connectivity strength” weighting matrix, \(W\), with \(W_{mn} \in \mathbb{R}^+\), for weighted graphs. The degree matrix, \(D\), and the Laplacian matrix, \(L\), with \(L_{nn} \in \mathbb{R}\), are defined using the adjacency/weighting matrix. When the relations between all pairs of vertices are mutually symmetric, then all the matrices involved are also symmetric, and such graphs are called undirected. If that is not the case, then the adjacency/weighting matrix is not symmetric and such graphs are called directed graphs.

The above-introduced graph framework is quite general and admits application to many different scenarios. For example, when performing an opinion poll within a social network, the members of that social network are treated as the vertices (data acquisition points). Their friendship relations are represented by the edges which model graph connectivity, see Sidebar 1, while the member answers play the role of graph signal values.

In the following, we shall demonstrate how this simple and intuitive concept provides a natural and straightforward platform to introduce the graph-counterparts of several fundamental signal processing algorithms.

**VI. System on a Graph**

The signal shift operator (unit time delay) is the lynchpin in discrete-time signal processing, but it is not so obvious to define on graphs due to the rich underlying connectivity structure. Topologically, the signal shift on a graph can be viewed as the movement of a signal sample from the considered vertex along all edges connected to this vertex. The signal shift operator can then be compactly defined using the graph adjacency matrix as \(x_{shifted} = Ax\).

To draw distinction between the standard shift and the graph shift operator, consider the line graph in Fig. 1b) (bottom) and the “spatial aware” graph in Fig. 2b), b), and assume that the input signal is a pulse that occurs only at the sensor \(n = 29\), that is, \(x(n) = \delta(n - 29)\). The shifted signal in classic signal processing (line graph in the bottom of Fig. 1b) (bottom)) will be \(x_{shifted}(n) = \delta(n - 28)\) and can be considered as a movement of the delta pulse along the line graph from vertex \(n\) to vertex \((n - 1)\). The same principle can be applied to the graph domain in Fig. 2a) whereby the delta pulse from vertex \(n = 29\) is moved to all its connected vertices, to obtain the shifted graph signal, \(x_{shifted}(n) = \delta(n - 27) + \delta(n - 28) + \delta(n - 51) + \delta(n - 59)\), as shown in Fig. 3.

If the shifted signal values are also scaled by the weighting coefficients of the corresponding edges, then the shifted signal is given by \(Wx\).

**Remark 4:** The standard shift operator, \(x(n) = x(n - 1)\), is a “one-to-one” mapping, while the graph shift operator, \(x_{shifted} = Ax\), is a “one-to-many” mapping which accounts for the underlying physics of the sensing process, not possible to achieve with standard DSP. Moreover, it also allows us to incorporate a contextual relation between the sensors within the grid through the weighting matrix \(W\). Notice that the graph shift operator does not satisfy the isometry property since the energy of the shifted signal is not the same as the energy of the original signal.

In analogy to the pivotal role of time shift in standard system theory, a system on a graph can be implemented as a linear combination of a graph signal and its graph shifted versions. The output from a system on a graph can then be written as

\[
y = h_0 W^0 x + h_1 W^1 x + \cdots + h_{M-1} W^{M-1} x = \sum_{m=0}^{M-1} h_m W^m x,
\]

(8)

where, by definition \(W^0 = I\), while \(h_0, h_1, \ldots, h_{M-1}\) are the system coefficients to be found (see Section X). Notice that for the directed and unweighted line graph in Fig. 1b) (bottom), the system on a graph in (8) reduces to the well known standard FIR filter, given by

\[
y(n) = h_0 x(n) + h_1 x(n-1) + \cdots + h_{M-1} x(n-M+1).
\]

**Remark 5:** The above established link between the classic transfer function of a physical system and its graph-theoretic counterpart may serve to promote new algorithmic approaches, which stem from signal processing, into many application scenarios that are directly considered as graphs.

Observe that the Laplacian operator applied on a signal, \(Lx\), can be considered as a combination of the scaled original signal, \(Dx\), and its weighted shifted version, \(Wx\), since \(Lx = Dx - Wx\). A system defined using the Laplacian

\[
y = L^0 x + h_1 L^1 x + \cdots + h_{M-1} L^{M-1} x
\]

(10)

therefore allows us to always produce an unbiased estimate of a constant \(c\), that is, if \(x = c\) then \(y = c\).

From (10), a simple first order system based on the graph Laplacian can be written as

\[
y = x + h_1 Lx
\]

(11)

and is amenable, with slight modifications, to being used for efficient low-pass graph filtering, see Sidebar 2.

**Remark 6:** A system on a graph is conveniently defined by the “graph transfer function”, \(H(W)\), as

\[
y = H(W)x.
\]

(12)

For an unweighted graph, the weighting matrix, \(W\), is equal to the adjacency matrix, \(A\). Instead of the weighting matrix, \(W\), for the definition of a system on a graph, we can also use the Laplacian matrix, \(L = D - W\).
Sidebar 1: Graph Topology (Edges and Weights)

There are three possible classes of problems which dictate the definition of graph edges:

- Geometry of the vertex positions: The distances between vertex positions play a crucial role in establishing relations between the sensed data. In many physical processes, the presence of edges and their associated connecting weights is defined based on the vertex distances. An exponential function of the Euclidean distance between vertices, $W_{mn} = e^{-r_{mn}/\alpha}$ or $W_{mn} = e^{-r_{mn}/\alpha}$ if $r_{mn} < \tau$ and $W_{mn} = 0$ for $r_{mn} \geq \tau$. This form has been used in the graph in Fig. 2, whereby the altitude difference, $h_{mn}$, was accounted for as $W_{mn} = e^{-h_{mn}/\alpha} e^{-h_{mn}/\beta}$.

- Physically well defined relations among the sensing positions: Examples include electric circuits, linear heat transfer systems, spring-mass systems, and various forms of networks like social, computer or power networks. In these cases, the edge weights are defined as a part of problem definition.

- Data similarity dictates the underlying graph topology: This scenario is the most common in image and biomedical signal processing (see Sidebar 5). Various approaches can be used to define data similarity, including the correlation matrix between the signals at various sensors or the corresponding inverse covariance (precision) matrix. Learning a graph (its edges) based on the set of the available data is an interesting and currently extensively studied research area.

Fig. 4. A single pulse graph signal $x$ at the vertex $n = 29$, that is, $x(n) = \delta(n - 29)$, and its graph shifted version $x_{shifted} = Ax$. The shift operator is demonstrated on the north-east part of the graph from Fig. 5 around the vertex $n = 29$, is presented.

**Properties of a system on a graph:** Following the above discussion, it is now possible to link the properties of linear systems with those of systems on a graph. From equations (8)-(12) the system on a graph is said to be:

- Linear, if $H(W)(a_1x_1 + a_2x_2) = a_1y_1 + a_2y_2$.

- Shift invariant, if $H(W)(Wx) = W(H(W)x)$.

**Remark 7:** A system on a graph, defined by $H(W) = h_0W^0 + h_1W^1 + \cdots + h_{M-1}W^{M-1}$ (13) is linear and shift invariant, since the matrix multiplication of the square weighting matrices is associative $(WW)W = (W^m)W$, that is $WW^m = W^mW$.

**VII. Graph Fourier Transform**

While classic spectral analysis is performed in the Fourier domain, spectral representations of graph signals employ either the adjacency/weighting matrix or the graph Laplacian eigenvalue decomposition. For the latter case we have

$L = U\Lambda U^{-1}$,

where $U$ is an orthonormal matrix of the eigenvectors, $u_k$, of the graph Laplacian matrix, $L$ (in its columns), and $\Lambda$ is a diagonal matrix of the corresponding eigenvalues, $\lambda_k$. These eigenvectors may then be used as a set of bases for spectral segmentation of graphs, see Sidebar 3.

The graph Fourier transform, $X$, of a graph signal, $x$, is then defined as

$X = U^{-1}x$. (14)

Physically, since $U^{-1} = U^T$, the element $X(k)$ of a graph Fourier transform, $X$, represents a projection of the graph signal, $x$, onto the $k$-th eigenvector, $u_k \in U$, that is

$X(k) = \sum_{n=1}^{N} x(n)u_k(n)$. (15)

The inverse graph Fourier transform is then straightforwardly obtained as

$x = U X$ (16)

or

$x(n) = \sum_{k=1}^{N} X(k)u_k(n)$. (17)

**Remark 8:** In analogy to the classic Fourier transform where the signal is projected onto a set of harmonic orthogonal bases, $X = U^{-1}x$, where $U$ is the matrix of harmonic bases $u_k = \left[ e^{i2\pi k/N}, \ldots, e^{i(2\pi k(N-1)/N)} \right] / \sqrt{N}$, the graph Fourier transform can be understood as a signal decomposition onto the set of eigenvectors of the graph Laplacian (or the adjacency matrix) that serve as orthonormal basis functions. In the case of a circular graph, the graph Fourier transform reduces to the standard discrete Fourier transform (DFT). For this reason, the transform in (13) is referred to as the graph discrete Fourier transform (GDFT).

Classic spectral analysis can thus be considered as a special case of graph signal spectral analysis, with the adjacency matrix defined on an unweighted circular directed graph (a line graph with the circularly connected last and first vertex), when $u_k = \left[ e^{i2\pi k/N}, \ldots, e^{i(2\pi k(N-1)/N)} \right] / \sqrt{N}$. This becomes obvious by considering that the eigenvalues of a directed unweighted circular graph, $\lambda_k = e^{-i2\pi k/N}$, are easily obtained as a solution of the eigenvalue/eigenvector relation $Au_k = \lambda_k u_k$. For a vertex $n$, this relation is of the form $u_k(n) = \lambda_k u_k(n)$. The previous vector elements $u_k(n)$ and eigenvalues $\lambda_k$ are the solutions of this
was filtered using Taubin’s algorithm, with a good and very simple approximation of a graph low-pass filter. Values of the squared local deviation, \((x(n) - x(m))^2\), correspond to a smooth, slow-varying signal. For a constant signal, \(x = c\), we therefore have \(E_x = 0\).

Physically, the minimum of \(xLx^T\) implies the smoothest possible signal and to arrive at this solution we may employ steepest descent. Then, the signal value at an iteration \(p\) is adjusted in the opposite direction of the gradient, toward the minimum of \(xLx^T\). The gradient of this quadratic form is \(\partial E_x / \partial x^T = 2Lx\), and yields the iterative procedure

\[
x_{p+1} = x_p - \alpha Lx_p = (I - \alpha L)x_p.
\]

Notice that the signal \(x_{p+1}\) can be considered as an output of the first order system in (11), with \(h_1 = -\alpha\), and this relation can be used for simple and efficient filtering of graph signals.

Since the minimum of the quadratic form \(xLx^T\) corresponds to a constant signal, in order to avoid obtaining only constant steady state (i.e., to account for the slow-varying part of the graph signal as well), the above iteration process can be used in alternation with \(x_{p+2} = (I + \beta L)x_{p+1}\). A compact form of these two iterative processes is known as Taubin’s \(\alpha - \beta\) algorithm and is given by

\[
x_{p+2} = (I + \beta L)(I - \alpha L)x_p.
\]

For appropriate values of \(\alpha\) and \(\beta\), this system can give a good and very simple approximation of a graph low-pass filter.

In our experiment, the original noisy signal from Fig. 3 was filtered using Taubin’s algorithm, with \(\alpha = 0.2\) and \(\beta = 0.1\). After 50 iterations, the signal-to-noise ratio improved from the original \(SNR_0 = 14.2\) dB to 26.8 dB.

Sidebar 2: Smoothness and Filtering on a Graph

The quadratic form of a graph signal is given by

\[
E_x = xLx^T = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} W_{nm}(x(n) - x(m))^2
\]

and can be used to define signal smoothness since small values of the squared local deviation, \((x(n) - x(m))^2\), correspond to a smooth, slow-varying signal. For a constant signal, \(x = c\), we therefore have \(E_x = 0\).

For appropriate values of \(\alpha\) and \(\beta\), this system can give a good and very simple approximation of a graph low-pass filter.

From (19), \(U^{-1}y = H(\Lambda)U^{-1}x\), or in terms of the graph Fourier transform of the input and output signal

\[
Y = H(\Lambda)X.
\]

The classic spectral transfer function for (6) is then obtained by using the adjacency matrix of an unweighted directed circular graph whose eigenvalues are \(\lambda_k = e^{-j2\pi k/N}\).

VIII. SPECTRAL DOMAIN OF SYSTEM ON GRAPHS

Consider a system on a graph, as in (10), defined by its Laplacian matrix, given by

\[
y = \sum_{m=0}^{M-1} h_m L^m x.
\]

Upon employing the eigen-domain (graph spectral) representation of the Laplacian matrix, \(L = U\Lambda U^{-1}\), we have

\[
y = \sum_{m=0}^{M-1} h_m U\Lambda^m U^{-1} x = U H(\Lambda) U^{-1} x,
\]

where

\[
H(\Lambda) = \sum_{m=0}^{M-1} h_m \Lambda^m
\]

is the transfer function of the graph system.

IX. SPECTRAL DOMAIN FILTER DESIGN

Consider a desired graph transfer function, \(G(\Lambda)\). Like in classic signal processing, a system with this transfer function can be implemented either in the spectral domain or in the vertex domain.

The spectral domain implementation is straightforward and can be performed in the following three steps:

1. Calculate the GDFT of the input graph signal \(X = U^{-1}x\).
2. Multiply the GDFT of the input graph signal with transfer function \(G(\Lambda)\) to obtain \(Y = G(\Lambda)X\), and...
The eigenvector corresponding to \( \lambda_1 = 0 \) is a constant (maximally smooth for any vertex ordering) and thus not appropriate for vertex ordering and segmentation.

Spectral similarity of vertices can also be defined using eigenvectors, if the eigenvector elements \( u_k(n) \), \( k = 1, 2, \ldots, P \) are assigned to the vertex \( n \). If \( u_k \) is omitted, then a \((P - 1)\)-dimensional spectral vector becomes \( q_n = [u_2(n), \ldots, u_P(n)]^T \).

The spectral similarity between vertices \( n \) and \( m \) is then defined as the two-norm \( \|q_n - q_m\|_2 \). Commonly, we use a few lower-order (smooth) spectral coefficients to define spectral similarity.

From here, we can now proceed in two ways:

- Keep the original vertex positions and color them according to the spectral vectors \( q_n \). For the graph from Fig. 2 such coloring is performed using the eigenvector elements \( u_2(n) \), \( u_3(n) \), and \( u_4(n) \) as color coordinates for the vertex \( n \). Similar colors indicate high spectral similarity.

Graph coloring using eigenvector values \( u_2(n) \), \( u_3(n) \), and \( u_4(n) \) as color coordinates for vertex \( n \)

3) Calculate the output graph signal as the inverse graph Fourier transform of \( Y \) to yield \( y = UY \).

Notice that this procedure may be computationally very demanding for large graphs where it may be easier to implement the desired filter (or its close approximation) in the vertex domain, in analogy to the time domain in the classic approach. This means that we have to find the coefficients, \( h_0, h_1, \ldots, h_{M-1} \) in (8), such that its spectral representation, \( H(A) \), is equal (or at least as close as possible) to the desired \( G(A) \).

In other words, the transfer function of the vertex domain system in (8), given by \( H(\lambda_k) = h_0 + h_1 \lambda_k + \ldots h_{M-1} \lambda_k^{M-1} \), should be equal to the desired transfer function, \( G(\lambda_k) \), for each \( k \). This condition leads to a system of linear equations

\[
\begin{align*}
    h_0 + h_1 \lambda_1 + \ldots h_{M-1} \lambda_1^{M-1} &= G(\lambda_1) \\
    h_0 + h_1 \lambda_2 + \ldots h_{M-1} \lambda_2^{M-1} &= G(\lambda_2) \\
    &\vdots \\
    h_0 + h_1 \lambda_N + \ldots h_{M-1} \lambda_N^{M-1} &= G(\lambda_N).
\end{align*}
\]

The matrix form of this system is given by

\[
V_\lambda h = g,
\]

where \( V_\lambda \) is a Vandermonde matrix formed of the eigenvalues, \( \lambda_k \), while \( h = [h_0, h_1, \ldots, h_{M-1}]^T \) is the vector of system coefficients that we wish to estimate, and

\[
g = [G(\lambda_1), G(\lambda_2), \ldots, G(\lambda_N)]^T = \text{diag}(G(A)).
\]

The system order \( M \) is typically significantly lower than the number of equations, \( N \). For such an overdetermined case, the least-squares approximation of \( h \) is obtained by minimizing the squared error, \( e^2 = \|V_\lambda h - g\|_2^2 \). Like in standard least-squares, the solution is obtained by direct minimization, \( \partial e^2/\partial h^T = 0 \), to yield

\[
\hat{h} = (V_\lambda^T V_\lambda)^{-1}V_\lambda^T g = \text{pinv}(V_\lambda)g. \tag{24}
\]

The so obtained solution, \( \hat{h} \), therefore represents the mean square error minimizer for \( V_\lambda h = g \). Notice that this solution may not satisfy \( V_\lambda \hat{h} = g \), in which case the coefficients \( \hat{g} \) (its spectrum \( \hat{G}(A) \)) may be used, that is

\[
V_\lambda \hat{h} = \hat{g}.
\]

Such a solution, in general, differs from the desired system coefficients \( g \) (its spectrum \( G(A) \)).

Example: Consider the graph signal from Fig. 3. The task is to design a graph filter whose frequency response is \( g(\lambda_k) = \exp(-\lambda_k) \) and to filter the graph signal using this

Graph coloring using eigenvector values \( u_2(n) \), \( u_3(n) \), and \( u_4(n) \) as color coordinates for vertex \( n \)
SIDEBAR 4: COMMENTS ON THE FILTER DESIGN IN (22)

1) Consider the case when all the eigenvalues of the graph Laplacian, \( \mathbf{L} \), are distinct:
   a) If the filter order is such that \( M = N \), then the solution of (22) is unique, since the Vandermonde determinant is always nonzero,
   b) If the filter order is such that \( M < N \) (overdetermined system in (22)), the solution is obtained in the mean square sense as in (24).

2) If some of the eigenvalues are of a degree higher than one, the system in (22) reduces to a system of \( N_m < N \) linear equations (by removing multiple equations for the repeated eigenvalues).
   a) If the filter order is such that \( N_m < M \leq N \) (underdetermined system (22)) then \( (M - N_m) \) filter coefficients are free variables and the system has an infinite number of solutions, while all obtained filters are equivalent,
   b) If the filter order is such that \( M = N_m \), the solution to the system in (22) is unique,
   c) If the filter order is such that \( M < N_m \) (overdetermined system in (22)), the solution is obtained in the mean square sense.

3) Any filter of an order \( M > N_m \) has a unique equivalent filter whose order is at most \( N_m \). Such equivalence can be obtained by setting the free variables to zero, \( h_i = 0 \) for \( i = N_m, N_m + 1, \ldots, N - 1 \).

spectral domain graph filter. For \( M = 4 \), the corresponding system coefficients can be found to be \( h_0 = 0.9606, h_1 = -0.7453, h_2 = 0.1936 \), and \( h_3 = -0.0162 \). Upon signal filtering using the so defined graph transfer function, the output signal-to-noise ratio was \( SNR = 21.74 \text{ dB} \), that is a 7.54 dB improvement over the original signal-to-noise ratio \( SNR_0 = 14.2 \text{ dB} \).

More detail on the solution of the system in (22) and (23) is provided in Sidebar 4.

X. OPTIMAL DENOISING

Consider a measurement, as in the temperature measurement scenario in Fig. [1] which is composed of a slowly-varying desired signal, \( s \), and a superimposed fast changing disturbance, \( \varepsilon \), to give

\[ x = s + \varepsilon. \]

The aim is to design a graph filter for disturbance suppression (denoising), the output of which is denoted by \( y \).

The optimal denoising task can then be defined through a minimization of the cost function

\[ J = \frac{1}{2} \| y - x \|^2 + \alpha y^T \mathbf{L} y. \]  \hspace{1cm} (25)

The minimization of the first term, \( \frac{1}{2} \| y - x \|^2 \), enforces the output signal, \( y \), to be as close as possible, in terms of the minimum residual disturbance power, to the available observations, \( x \). As mentioned before, the second term, \( y^T \mathbf{L} y \), represents a measure of smoothness of the graph filter output, \( y \). For more detail on promoting smoothness of a graph signal, see Sidebar 2. The parameter \( \alpha \) models a balance between the closeness of the output, \( y \), and the observed data, \( x \), and the smoothness of output estimate \( y \). While the problem in (25) could be expressed as a Lagrangian constrained optimization, at this juncture we choose to focus more on the graph theoretic issues and hence we adopt a simpler option selecting externally the mixing parameter \( \alpha \).

The solution to this minimization problem follows from

\[ \frac{\partial J}{\partial y} = y - x + 2\alpha \mathbf{L} y = 0 \]

and results in a smoothing optimal denoiser in the form

\[ y = (\mathbf{I} + 2\alpha \mathbf{L})^{-1} x. \]

The Laplacian spectral domain form of this relation is

\[ \mathbf{Y} = (\mathbf{I} + 2\alpha \mathbf{L})^{-1} \mathbf{X}, \]

with the corresponding graph filter transfer function

\[ H(\lambda_k) = \frac{1}{1 + 2\alpha \lambda_k}. \]

For a small \( \alpha \), \( H(\lambda_k) \approx 1 + y \approx x \), while for a large \( \alpha \), \( H(\lambda_k) \approx \delta(k) \) and \( y \approx const., \) which enforces \( y \) to be maximally smooth (a constant, without any variation).

Using \( \alpha = 4 \), the obtained output signal-to-noise ratio for the graph signal from Fig. [3] was \( SNR = 26 \text{ dB} \), a 11.8 dB improvement over the original \( SNR_0 = 14.2 \text{ dB} \).

Remark 9: There are many cases when the graph topology is unknown, so that the graph structure, i.e., the Laplacian (graph edges and their weights) is also unknown. To this end, we may employ a group of methods for graph topology learning, based on the minimization of the cost function in (25) with respect to both the Laplacian, \( \mathbf{L} \), and the output signal, \( y \), with additional constraints imposed on the Laplacian values.

XI. CURRENT RESEARCH

Current research is mainly focused on graphs themselves, for example, on reducing the complexity of calculation in very large graphs, including downsampling, multirate analysis, compressive sensing, graph segmentation, and vertex-varying and vertex-frequency analysis.

XII. WHAT WE HAVE LEARNED

Natural signals (speech, biomedical, video) reside over irregular domains and are, unlike the signals in communications, not adequately processed using, e.g., standard harmonic analyses. While Data Analytics are heavily dependent on advances in DSP, neither the EE graduates worldwide nor practical data analysts are yet best prepared to employ graph algorithms in their future jobs. Our aim has been to fill this void by providing an example-driven platform to introduce graphs and their properties through the well understood notions of transfer functions, Fourier transform, and digital filtering.

While both a graph with \( N \) vertices and a classical discrete time signal with \( N \) samples can be viewed as \( N \)-dimensional vectors, structured graphs are irregular domains which convey information about both the signal generation and propagation mechanisms. We can then employ intuition and our know-how from Euclidean domains to revisit basic dimensionality reduction operations, such as coarse graining of graphs (cf. standard downsampling). In addition, in the vertex domain a number of
different distances (shortest-path, resistance, diffusion) have useful properties which can be employed to maintain data integrity throughout the processing, storage, communication and analysis stages, as the connectivities and edge weights are either dictated by the physics of the problem at hand or are inferred from the data. This particularly facilitates maintaining control and intuition over distributed operations throughout the processing chain.

It is our hope that this lecture note has helped to demystify graph signal processing for students and educators, together with empowering practitioners with enhanced intuition in graph-theoretic design and optimization. This material may also serve as a vehicle to seamlessly merge curricula in Electrical Engineering and Computing. The generic and physically meaningful nature of this example-driven Lecture Note is also likely to promote intellectual curiosity and serve as a platform to explore the numerous opportunities in manifold applications in our ever-growing interconnected world, facilitated by the Internet of Things.

AUTHORS

Ilija Kisil (i.kisil15@imperial.ac.uk) is a Ph.D. candidate at Imperial College London. His research interests include tensor decompositions, big data, efficient software for large scale problems, and graph signal processing.

Ervin Sejdić, SMIEEE, (esejdic@ieee.org) is an assistant professor at the University of Pittsburgh, USA. His research interests include biomedical signal processing, rehabilitation engineering, and neuroscience. He received the USA Presidential Early Career Award for Scientists and Engineers in 2016.

Anthony G. Constantinides, LFIEEE, (a.constantinides@imperial.ac.uk) is emeritus professor of signal processing in the Department of Electrical and Electronic Engineering at Imperial College London, United Kingdom.

REFERENCES

[1] N. Christofides, “Graph theory: An algorithmic approach”, Academic Press, 1975.
[2] F. Afrati and A. G. Constantinides, “The use of graph theory in binary block code construction”, in Proceedings of the International Conference on Digital Signal Processing, pp. 228-233, Florence, Italy, 310 August - 2 September, 1978
[3] O. J. Morris, M. de J. Lee, and A. G. Constantinides, “Graph theory for image analysis: An approach based on the shortest spanning tree”, IEE Proceedings F-Communications, Radar and Signal Processing, vol. 133, no. 2, pp. 146-152, 1986.
[4] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, “Discrete signal processing on graphs: Sampling theory,” IEEE Trans. on Signal Processing, vol. 63, no. 24, pp. 6510-6523, Dec. 15, 2015.
[5] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs,” IEEE Trans. on Signal Processing, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.
[6] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs: Frequency analysis,” IEEE Trans. on Signal Processing, vol. 62, no. 12, pp. 3042–3054, Jun. 2014.
[7] 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 Mag., vol. 30, no. 3, pp. 83–98, May 2013.
[8] L. Stanković, M. Daković, and E. Sejdić, “Vertex-frequency analysis: A way to localize graph spectral components,” IEEE Signal Processing Mag., Vol.34, July 2017.
[9] S. Saito, H. Suzuki, and D. P. Mandic, “Hypergraph p-Laplacian: A differential geometry view”, Proceedings of the the Thirty Second AAAI Conference on Artificial Intelligence (AAAI-18), pp. 3984-3991, 2018.
[10] L. Stanković and E. Sejdić, “Vertex-frequency analysis of graph signals,” Springer Nature, 2019.
[11] M.H. Hayes, “Statistical digital signal processing and modeling,” John Wiley & Sons, 2009.