Graph Signal Processing: History, Development, Impact, and Outlook

Geert Leus, Antonio G. Marques, José M. F. Moura, Antonio Ortega, David I Shuman

Signal processing (SP) excels at analyzing, processing, and inferring information defined over *regular* (first continuous, later discrete) domains such as time or space. Indeed, the last 75 years have shown how SP has made an impact in areas such as communications, acoustics, sensing, image processing, and control, to name a few. With the digitalization of the modern world and the increasing pervasiveness of data-collection mechanisms, information of interest in current applications oftentimes arises in non-Euclidean, irregular domains. Graph signal processing (GSP) generalizes SP tasks to signals living on non-Euclidean domains whose structure can be captured by a weighted graph. Graphs are versatile, able to model irregular interactions, easy to interpret, and endowed with a corpus of mathematical results, rendering them natural candidates to serve as the basis for a theory of processing signals in more irregular domains.

The term "graph signal processing" was coined a decade ago in the seminal works [1], [2], [3], [4]. Since these papers were published, GSP-related problems have drawn significant attention, not only within the SP community [5] but also in machine learning venues, where research in

Due to the publication rules, the final number of references had to be cut down to 60. As a result, several relevant works had to be removed from the final list. The a version of this paper with an extended list of references is available at http://arxiv.org/abs/2303.12211.

Geert Leus is with the Fac. of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands, g.j.t.leus@tudelft.nl

Antonio G. Marques is with the Dept. of Signal Theory and Communications, King Juan Carlos University, Madrid, Spain, antonio.garcia.marques@urjc.es

José M. F. Moura is with the Dept. of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, moura@ece.cmu.edu

Antonio Ortega is with the Dept. of Electrical and Computer Engineering, University of Southern California, Los Angeles, California, USA, aortega@usc.edu

David I Shuman is with Franklin W. Olin College of Engineering, Needham, Massachusetts, USA, dshuman@olin.edu

graph-based learning has increased significantly [6]. Graph signals are well-suited to model measurements/information/data associated with (indexed by) a set where: (i) the elements of the set belong to the same class (regions of the cerebral cortex, members of a social network, weather stations across a continent), (ii) there exists a relation (physical or functional) of proximity, influence or association among the different elements of that set, and (iii) the strength of such a relation among the pairs of elements is not homogeneous. In some scenarios, the supporting graph is a physical, technological, social, information or biological network where the links can be explicitly observed. In many other cases, the graph is implicit, capturing some notion of dependence or similarity across nodes, and the links must be inferred from the data itself. As a result, GSP is a broad framework that encompasses and extends classical SP methods, tools, and algorithms to application domains of the modern technological world, including social, transportation, communication and brain networks; recommender systems; financial engineering; distributed control; and learning, just to name a few. While the theory and application domains of GSP continue to expand, GSP has become a technology with wide use. It is a research domain pursued by a broad community, the subject of not only many journal and conference articles, but also of textbooks [5], special issues of different journals, symposia, workshops, and special sessions at ICASSP and other SP conferences.

In this article, we provide an overview of the evolution of GSP, from its origins to the challenges ahead. The first half is devoted to reviewing the history of GSP and explaining how it gave rise to an encompassing framework that shares multiple similarities with SP, and especially digital signal processing (DSP). A key message is that GSP has been critical to develop novel and technically sound tools, theory, and algorithms that, by leveraging analogies with and the insights of DSP, provide new ways to analyze, process, and learn from graph signals. In the second half, we shift focus to review the impact of GSP on other disciplines. First, we look at the use of GSP in data science problems, including graph learning and graph-based deep learning. Second, we discuss the impact of GSP on applications, including neuroscience and image and video processing. We conclude with a brief discussion of the emerging and future directions of GSP.

I. The Early Roots

The roots of GSP can be traced to algebraic and spectral graph theory, harmonic analysis, numerical linear algebra, and specific applications of these ideas to areas such as data representations for high-dimensional data, pattern recognition, (fast) transforms, image processing, computer graphics, statistical physics, partial differential equations, semi-supervised learning, and neuroscience. Algebraic graph theory [7] dates back to the 1700s, and spectral graph theory [8] dates back to

the mid-1900s. They study mathematical properties of graphs and link the graph structure to the spectrum (eigenvalues and eigenvectors) of matrices related to the graph. However, they generally did not consider potential signals that could be living on the graph.

In the late 1990s and early 2000s, graph-based methods for analyzing and processing data became more popular – independently – in a number of disciplines, including computer graphics (e.g., [9]), image processing (e.g, [10]), graphical models in Bayesian statistics (e.g., [11], [12]), dimensionality reduction (e.g., [13]) and semi-supervised learning (e.g., [14]) within machine learning, and neuroscience (e.g., detailed history included in [15]). For example, in computer graphics, Taubin utilized graph Laplacian eigenvectors to perform surface smoothing by applying a low pass graph filter to functions defined on polyhedral surfaces [9], and later used similar ideas to compress polygonal meshes. In image processing, weighted graphs can be defined with edges being a function of pixel distance and intensity differences. Such semi-local and non-local graphs were exploited for denoising (bilateral filtering), image smoothing, and image segmentation (e.g., [16], [10]). Graphical models [12] – and in particular undirected graphical models, also referred to as Markov random fields – model data as a family of random variables (the vertices), with the graph edges capturing their probabilistic dependencies. Through the graph, these models sparsely encode complex probability distributions in high-dimensional spaces. Graphical models have been widely used in Bayesian statistics and Bayesian probabilistic approaches, kernel regression methods, statistical learning, and statistical mechanics [17]. We return to semi-supervised learning and neuroscience and their connections with GSP in Sec. III-C and Sec. IV-A, respectively.

Also in the late 1990s, two new models were introduced for random networks (graphs) to model the structure of complex engineered systems, going well beyond the classical Erdös-Rényi random graphs; real world large networked systems exhibit small world characteristics (the Watts-Strogatz model) and scale free degree distributions (the Barabási-Albert model). This led to a flurry of activity, usually referred to as *Network Science*, concerned with analyzing and designing complex systems like telecommunication, power grid, and large scale infrastructure networks [18]. While the central focus of network science was on properties of the network and its nodes (e.g., centralities, shortest paths, and clustering coefficients), network science researchers also leveraged graphs to explore the dynamics of processes such as percolation, traffic flows, synchronization, and epidemic spread [18, Part V], often adopting mean field approximations. For example, in the investigation of the susceptible-infected-susceptible (SIS) epidemiological model in scale-free graphs in [19], each vertex can be seen as having a 0/1 (susceptible/infected) signal residing on it. Advancements in network science have certainly informed the subsequent development of GSP.

In parallel, a stream of new methods for analyzing data on graphs were investigated. These methods tried specifically to combine (i) intuition and dictionary constructions for performing computational harmonic analysis on data on Euclidean domains with (ii) generalizable ways to incorporate the structure of the underlying graph into the data transforms. For example, one of the first general wavelet constructions for signals on graphs was the spatial wavelet transform of [20], which was defined directly in the vertex domain. In the seminal work [21], diffusion wavelets were constructed by (i) creating a multiresolution of approximation spaces, each spanned by graph signals generated by diffusing a unit of energy outwards from each vertex for a fixed amount amount of time; and (ii) computing orthogonal diffusion wavelets to serve as basis functions for the detail spaces that are the sequential orthogonal complements of the approximation spaces. Spectral graph wavelets [1] traded off the orthogonality of diffusion wavelets for a simpler generative method for each wavelet atom: define a pattern in the graph spectral domain and localize that pattern to be centered at each vertex of the graph. Meanwhile, the algebraic signal processing approach [22] showed that classical SP can be captured by a triplet defined by a shift operator. Different shifts lead to different SP models and different Fourier transforms. In particular, it showed that a shift based on Chebyshev polynomials, appropriate for lattice models like in images, leads to standard block transforms such as the discrete cosine and Karhunen–Loève transforms (DCT and KLT) that can be understood as Fourier transforms on certain graphs. Numerous other types of multiresolution transforms and dictionaries for data residing on graphs, trees, and compact manifolds were investigated in the subsequent few years. These included lifting and pyramid transforms, graph filter banks, tight spectral frames, vertex-frequency transforms that generalized the classical short-time Fourier transform, and learned dictionaries (see [23], [24] for a more complete literature review and list of references). GSP arose from these different fields, coalescing multiple perspectives into a common framework and set of ideas. In the last decade, this unifying framework has evolved into a full-fledged theory and technology.

II. The Theoretical Underpinnings

Ten years ago, [1], [2], [3], [4] introduced the field of GSP and established many of its foundations. Remarkably, these works approached the problem from two different perspectives. Inspired by graph theory and harmonic analysis, the authors of [1], [2] use the graph Laplacian as the core of their theory, naturally generalizing concepts such as frequencies and filter banks to the graph domain. Differently, the authors of [3], [4] follow an algebraic approach, under which the multiplication of a graph signal by the adjacency matrix of the supporting graph yields the most basic operation

of shift for a graph signal. Based on this simple operation, more advanced tools such as filtering, graph Fourier transform, graph frequency, or total variation can be generalized to the vertex and spectral graph domains. Rather than being considered as competing approaches, these works brought complementary views and tools and, jointly, contributed to increase the attention to the field. After introducing some common notation, this section reviews these two approaches, and then explains how they were merged into an integrated framework that facilitated drawing links with classical SP and propelled the growth of GSP.

A. Basic definitions and notational conventions

The goal in GSP is to leverage SP and graph theory tools to analyze and process signals defined over a network domain, with notable examples including technological, social, gene, brain, knowledge, financial, marketing, and blog networks. In these setups, graphs are used to both index the data and to represent relations/similarities/dependencies among the locations of the data. We denote the underlying weighted graph by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \omega)$, where $\mathcal{V} := \{1, ..., N\}$ denotes the set of N graph vertices, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ denotes the set of graph edges, and $\omega : \mathcal{E} \to \mathbb{R}$ is a weight function that assigns a real-valued weight to each edge, with a higher edge weight representing a stronger similarity or dependency between the two vertices connected by that edge. A graph with edge weights all equal to 1 is called *unweighted*. A *graph signal* contains information associated with each vertex of the graph. For simplicity, we focus our discussion on scalar, real-valued graph signals (each signal is a mapping from $\mathcal{V}$ to $\mathbb{R}$), but the values associated with each node could be discrete, complex, or even vectors (e.g., when multiple features per node are observed). Each scalar, real-valued graph signal can equivalently be represented as an N-dimensional vector $\mathbf{x} := [x_1, ..., x_N]^\top$, with x_i (also written sometimes as $[\mathbf{x}]_i$) representing the value of the signal at vertex i. An example of a graph signal is shown in Fig. 1.

To gain some intuition, consider the problem of studying Twitter patterns. Assume that we have N Twitter users, each vertex $i \in \mathcal{V}$ represents a user i, and each edge $e = (i, j) \in \mathcal{E}$ captures that two users i and j follow each other. The data, x_i, indexed by node i could, e.g., be the number of tweets that user i tweeted in a given time interval. In a second application, to understand traffic flow in cities, we can examine the number of pick-ups of for-hire-vehicles (FHV) (e.g., taxis, Uber or Lyft cars, etc) over a given time period. The graph $\mathcal{G}$ can be the city road map, with the vertices $i \in \mathcal{V}$ representing intersections and the edges $e \in \mathcal{E}$ representing road segments between intersections. The data x_i at each vertex i might, e.g., be the number of pick-ups close to that intersection over the

time period of interest. The graphs $\mathcal{G}$ in such real-world applications can be modeled as undirected (if $(i,j) \in \mathcal{E}$, then $(j,i) \in \mathcal{E}$), or directed (e.g., to capture one-way streets).

Classical SP signals such as audio and image signals that reside on Euclidean domains can also be viewed as graph signals. Consider for instance finite-length discrete-time one-dimensional signals, e.g., the N vertices of the graph are the time instances $i = 0, ..., N-1$, with N being the window length. Since the signal value x_{i+1} at time $i+1$ is usually closely related to the signal value x_i at the preceding time, there is a directed edge from vertex i to vertex $i+1$. At $i = N-1$, there are different options for the boundary conditions; here, we consider the periodic boundary condition, which means that the time instant "next" to the terminal instant $N-1$ is $i = 0$. The resulting "time graph" is then a directed cycle $\mathcal{G}_{dc}$ (see Fig. 2). By similar reasoning, vertices in the image graph represent the pixels, and because the image brightness or color x_{ij} at pixel (i,j) is usually highly related to the brightness or colors of its four neighboring pixels, there are undirected edges from (i,j) to its neighboring pixels. The corresponding graph is then an undirected $2D$-lattice.

At the core of GSP are $N \times N$ matrices that encode the graph's topology. The most prominent are (i) the adjacency matrix $\mathbf{A}$, whose (i,j)-entry is the edge weight $\omega((i,j))$ if $(i,j) \in \mathcal{E}$ and 0 otherwise; and (ii) the combinatorial (or non-normalized) graph Laplacian $\mathbf{L} := \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \text{diag}(\mathbf{A1})$ is the diagonal matrix of vertex degrees (sums of the weights of the edges adjacent to each vertex) and $\mathbf{1}$ is an $N \times 1$ vector of all ones; and (iii) the normalized graph Laplacian $\mathbf{L}_{\text{norm}} := \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$. We elaborate on the role of these matrices in the next section.

B. The spectral approach for GSP

Classical Fourier analysis of a one-dimensional signal decomposes the signal into a linear combination of complex exponential functions (continuous or discrete) at different frequencies, with increasing frequencies corresponding to higher rates of oscillation and basis functions that are less smooth. The spectral approach to GSP [1], [2] generalizes this classical Fourier analysis by writing graph signals as linear combinations of a basis of graph signals with the property that the basis vectors can be (roughly) ordered according to how fast they oscillate across the graph, or, related, how smooth they are with respect to the underlying graph structure. By "smooth" in this context, we mean that the values of the graph signal at each pair of neighboring vertices are similar.

The operator that captures this notion of smoothness with respect to the underlying (undirected) graph is the graph Laplacian $\mathbf{L}$. It is a discrete difference operator, since we have

$$[\mathbf{Lx}]_i = \sum_{j=1}^{N} A_{i,j}(x_i - x_j) = \sum_{j \in \mathcal{N}_i} A_{i,j}(x_i - x_j),$$

where $\mathcal{N}_i$ is the neighborhood of node i. Since $\mathbf{L}$ is a real symmetric matrix, it has a set of orthonormal eigenvectors $\{\mathbf{v}_\ell\}_{\ell=0}^{N-1}$ and a set of real non-negative eigenvalues $\{\lambda_\ell\}_{\ell=0}^{N-1}$. Assuming a connected graph, it can further be shown that there is only one eigenvalue zero, e.g., $\lambda_0 = 0$, with corresponding eigenvector $\mathbf{v}_0 = \mathbf{1}/\sqrt{N}$. In matrix form we obtain $\mathbf{L} = \mathbf{V}\text{diag}(\boldsymbol{\lambda})\mathbf{V}^\top$, with $\mathbf{V} = [\mathbf{v}_0, \ldots, \mathbf{v}_{N-1}]$ and $\boldsymbol{\lambda} = [\lambda_0, \ldots, \lambda_{N-1}]^T$.

Importantly, the graph Laplacian can also be viewed as a graph extension of the time-domain Laplacian operator $\frac{\partial^2}{\partial t^2}$. Just as the one-dimensional complex exponentials – the eigenfunctions of the time-domain Laplacian operator – capture a notion of frequency, we can interpret the graph Laplacian eigenvectors as graph frequency vectors, with the associated graph Laplacian eigenvalues capturing a notion of the rate of oscillation [2].

The Laplacian operator can also be used to introduce a measure of smoothness for a graph signal $\mathbf{x}$; namely, the graph Laplacian quadratic form

$$\mathbf{x}^\top \mathbf{L}\mathbf{x} = \sum_{(i,j)\in\mathcal{E}} A_{i,j}(x_i - x_j)^2, \tag{1}$$

which penalizes large differences between signal values at strongly connected vertices. Because $\mathbf{v}_\ell^\top \mathbf{L}\mathbf{v}_\ell = \lambda_\ell$, it is then clear from (1) that the larger the graph frequency λ_ℓ, the less smooth (or more variable) the graph Laplacian eigenvector $\mathbf{v}_\ell$. So with the indexing convention $0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{N-1}$, the graph frequency vectors $\{\mathbf{v}_\ell\}_{\ell=0}^{N-1}$ are ordered according to increasing variability (see Fig. 1). Using the Laplacian eigenvectors as the basis, we can now define a graph Fourier transform (GFT) as $\mathbf{V}^\top$. It transforms a graph signal $\mathbf{x}$ into its frequency components as $\hat{\mathbf{x}} = \mathbf{V}^\top \mathbf{x}$.

Graph filters can then be interpreted as operators that modify the different frequency components of a signal $\mathbf{x}$ individually. That is, the graph filter operation can be represented in the graph Fourier domain by $\mathcal{H} : \mathbb{R} \to \mathbb{R}$, so that $[\hat{\mathbf{y}}]_\ell = \mathcal{H}(\lambda_\ell)[\hat{\mathbf{x}}]_\ell$. In most cases, the spectral function $\mathcal{H}$ (oftentimes referred to as a kernel) is set to a pre-specified analytical form (typically parametric) that promotes certain properties in the output signals (e.g., rectangular kernels promote smoothness and remove noise; see Fig. 1). However, non-parametric approaches can also be used. Equally important, [2] also illustrates how graph filters can be used to interpolate missing values and to design signal dictionaries whose atoms concentrate their energy around a few frequencies or vertices, highlighting their relevance in a number of applications.

C. The algebraic approach for GSP

In classical SP, convolution is a key building block present in most algorithms, including filtering and sampling, and interpolation. In defining convolution and filtering, the time shift – the unit delay

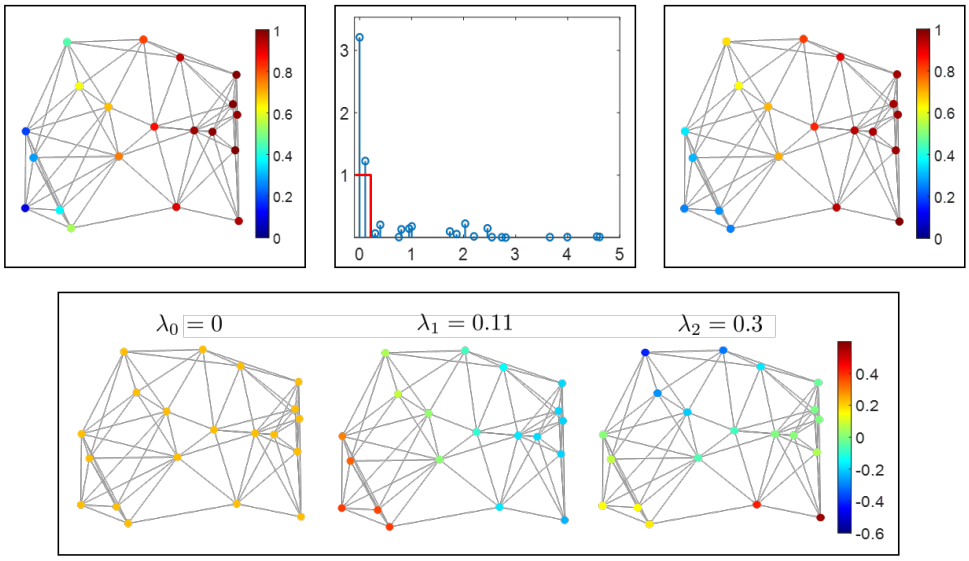

Fig. 1: Top left: An example of a graph with a color-coded graph signal on top; Top middle: The signal in the graph frequency domain and in red the frequency response of a potential low-pass graph filter; Top right: The filtered graph signal; Bottom: The first three eigenvectors of the graph Laplacian ordered with decreasing smoothness (increasing eigenvalue).

that transforms a signal into a delayed version of itself – plays a critical role. The outputs of linear time invariant filters are weighted linear combinations of delayed versions of the input. Similarly, the DFT can be understood as the transformation that diagonalizes every linear time invariant filter and provides an alternative description for signals and filters.

In extending these ideas to GSP, two key contributions of [3], [4] are: (i) highlighting the relevance of defining a "graph-aware" operator that plays the role of the "most basic operation" to be performed on a signal $\mathbf{x}$ defined over a graph $\mathcal{G}$; and (ii) setting this operation as $\mathbf{Ax}$, i.e., the multiplication of the graph signal $\mathbf{x}$ by the adjacency matrix $\mathbf{A}$ of $\mathcal{G}$. The motivation for the latter choice is twofold. First, $\mathbf{A}$ is a simple (parsimonious and linear) operator that combines the values of $\mathbf{x}$ in a manner that accounts for the local connectivity of $\mathcal{G}$. Second, when particularized to time-varying signals defined over the directed cycle $\mathcal{G}_{dc}$, using $\mathbf{A}_{dc}\mathbf{x}$ is equivalent to the classical unit delay; i.e., $[\mathbf{A}_{dc}\mathbf{x}]_{i+1} = [\mathbf{x}]_i$.

How can this basic graph-aware operator be leveraged to design (i) linear graph filters that are applied to a graph signal to generate another graph signal, and (ii) linear transforms that provide an alternative representation for a graph signal? In classical SP,[1] the basic non-trivial operation applied to a signal is the unit delay (time shift); in other words, the simplest filter is the time

[1]Because graphs are finite, we consider DSP with finite signals, and, for simplicity, with periodic signal extensions.

shift filter z^{-1}. Generic linear filters are then polynomials of this basic operator of the form $p(z) = p_0 + p_1 z^{-1} + \cdots + p_1 z^{-(N-1)}$, with z^{-l} being the consecutive application of the operator z^{-1} to a time signal l times. DSP polynomial filters are shift invariant in the sense that $z^{-1} \cdot p(z) = p(z) \cdot z^{-1}$.

Hence, to address the first question, [3] sets the simplest signal operation in GSP as the multiplication by the adjacency matrix $\mathbf{A}$ and, subsequently, defines graph filters as (matrix) polynomials of the form $p(\mathbf{A}) = p_0\mathbf{I}_N + p_1\mathbf{A} + \cdots + p_1\mathbf{A}^{(N-1)}$. It is easy to see that polynomial filters are $\mathbf{A}$ invariant, in the sense that $\mathbf{A} \cdot p(\mathbf{A}) = p(\mathbf{A}) \cdot \mathbf{A}$. Apart from the theoretical motivation, the polynomial definition exhibits a number of advantages. When applied to a graph signal $\mathbf{x}$, the operation $\mathbf{Ax}$ can be understood as a local linear combination of the signal values at one-hop neighbors. Similarly, $\mathbf{A}^2\mathbf{x}$ is a local linear combination of $\mathbf{Ax}$, reaching values that are in the 2-hop neighborhood. From this point of view, a graph filter $p(\mathbf{A})$ represented by a polynomial of order L is mixing values that are at most L hops away, with the polynomial coefficients $\{p_l\}_{l=0}^{L}$ representing the strength given to each of the neighborhoods. Another advantage is that, if $\mathbf{A}$ is set to the $\mathbf{A}_{dc}$ (the graph representing the support of classical time signals) the graph polynomial definition $p(\mathbf{A}_{dc})$ reduces to the classical time shift definition $p(z^{-1})$, so that graph filters become linear time invariant filters.

To address the second question, [3] defines the graph Fourier transform (GFT) as the linear transform that diagonalizes these graph filters of the form $p(\mathbf{A})$. Letting $\mathbf{A} = \mathbf{V}\text{diag}(\boldsymbol{\lambda})\mathbf{V}^{-1}$ be the eigendecomposition of the (possibly directed) adjacency matrix $\mathbf{A}$, then $p(\mathbf{A}) = p(\mathbf{V}\text{diag}(\boldsymbol{\lambda})\mathbf{V}^{-1}) = \mathbf{V}(p(\text{diag}(\boldsymbol{\lambda}))\mathbf{V}^{-1}$ (note that we use $\mathbf{V}^{-1}$ now instead of $\mathbf{V}^{\top}$ because the eigenvectors are not necessarily orthonormal as for the Laplacian). In other words, matrix polynomials can be understood as operators that transform the input by (i) multiplying it by the matrix $\mathbf{V}^{-1}$, (ii) applying an orthogonal operator $p(\text{diag}(\boldsymbol{\lambda}))$, and (iii) transforming the result back to the vertex domain with a multiplication by $\mathbf{V}$. The GFT of a graph signal and the signal spectral representation is then set as the multiplication by $\mathbf{V}^{-1}$ and the frequency response of a filter is found by calculating $p(\text{diag}(\boldsymbol{\lambda}))$, similarly as described in the previous section on the spectral approach. From the GFT of the signal, common SP concepts can now be defined in GSP [4] including ordering of graph frequencies, low and high graph frequencies, or design of low- and high-pass graph filters. Fig. 2 shows the generalization of the time domain to a more general graph domain. Applications in [3] to data prediction, graph signal compression, data classification, and customer behavior prediction for service providers and in [4] to filter design and malfunction detection in sensor networks show the breadth of application domains.

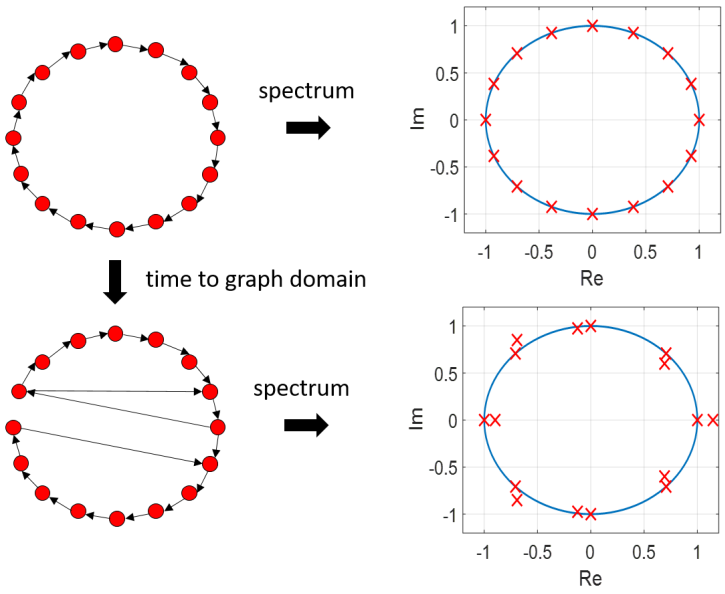

Fig. 2: Left: From the directed cycle representing the time domain to a general graph; Right: Eigenvalues (spectrum) of the related adjacency matrices.

D. The benefits of a joint framework

While having different origins, the approaches in [1], [2] and [3], [4] bring complementary perspectives. The work in [1], [2] relies on the graph Laplacian to capture the structure of $\mathcal{G}$, uses its eigendecomposition to characterize graph signals and define filtering operations, and draws clear links with existing graph-based techniques in a number of applications. In [3], [4], the focus is on the shift operation in the vertex domain, postulating the use of the adjacency matrix as the building block to design GSP algorithms, and unveiling a number of similarities with classical SP. Although some early works mixed features of [1], [2], [3], [4] (e.g., the use of polynomials based on the Laplacian matrix), the publication of these four papers and related works led to the emergence of works that combine both approaches under a common framework. One way to do so is to define a generic "graph-shift operator" (GSO) that plays a dual role: (i) it can be viewed as the most basic operation applied to a graph signal, and (ii) it codifies the structure of the graph in a more generic way than $\mathbf{L}$ or $\mathbf{A}$, so that it can be used to tackle a broader range of setups. Under this framework, the linear GSO $\mathbf{S} \in \mathbb{R}^{N \times N}$ has been set to different adjacency matrices (e.g., one-hop, two-hops), different graph Laplacians (e.g., combinatorial, normalized, random walk), the precision matrix of a Gaussian Markov random field, or even combinations of those. Based on the eigendecomposition of this operator, given by $\mathbf{S} = \mathbf{V}\text{diag}(\boldsymbol{\lambda})\mathbf{V}^{-1}$, linear graph filtering can be equivalently understood as an operator that is

linear and orthogonal (diagonal) in the frequency domain defined by $\mathbf{V}^{-1}$, or as the multiplication by a matrix that is a linear combination of successive applications (powers) of the GSO $\mathbf{S}$:

$$\mathbf{H}(\mathbf{S}) = \mathbf{V}\text{diag}(\widehat{\mathbf{h}})\mathbf{V}^{-1} \quad \text{or} \quad \mathbf{H}(\mathbf{S}) = \sum_{l=0}^{N-1} h_l \mathbf{S}^l, \tag{2}$$

where the notation $\mathbf{H}(\mathbf{S})$ is used to emphasize the dependence on the GSO $\mathbf{S}$. The first definition in (2) focuses on the frequency domain, with the filter parameters being the N-dimensional frequency response $\widehat{\mathbf{h}} = [\widehat{h}_0, ..., \widehat{h}_{N-1}]^\top$. The second definition in (2) focuses on the vertex domain, with the parameters of the filter being the N filter taps $\mathbf{h} = [h_0, ..., h_{N-1}]^\top$. Although we focus on degree $N-1$ polynomials, thanks to the Cayley–Hamilton Theorem, the definition in (2) can represent a matrix polynomial of any degree [3]. With these models at hand, the literature promptly addressed tasks such as prediction, classification, compression, filter identification, and filter design in graph/network contexts [3], [25], [26]. The particular solution obtained for any of these tasks depends on the GSO at hand, as well as the assumptions on the graph filter. For example, if the goal is to estimate the graph-based linear mapping from a set of input-output pairs collected in matrices $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_M]$ and $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_M]$, one requires $M = N$ input-output pairs if no structure is assumed for $\mathbf{H}$ and a single $M = 1$ pair if one assumes that $\mathbf{H}$ is a graph filter. Equally important, defining the counterparts of classical finite impulse response (FIR) and infinite impulse response (IIR) filters as $\mathbf{H}_{FIR}(\mathbf{S}) = \sum_{l=0}^{L-1} b_l \mathbf{S}^l$ and $\mathbf{H}_{IIR}(\mathbf{S}) = (\sum_{l=0}^{L-1} a_l \mathbf{S}^l)^{-1}$, identification from input-output observations is feasible even if only a subset (with cardinality larger than $2L$) of the signal values is observed [26]. Additionally, using the definitions in (2), it is not difficult to show that any cascade/parallel/feedback connection of graph filters can also be written as a graph filter, opening the door to make and exploit connections between graph-network processes and classical tools in control.

A natural next step is to use (2) to model certain properties of classes of graph signals of interest. To be more specific, consider that we model a graph signal $\mathbf{x} \in \mathbb{R}^N$ from a class of interest as $\mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{z}$ with $\mathbf{z}$ being a hidden seed signal and $\mathbf{H}(\mathbf{S})$ a generative graph filter that "transfers" some of the properties of $\mathbf{S}$ to $\mathbf{x}$. While mathematically simple, modelling graph signals as $\mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{z}$ has proven to be fruitful. A typical approach is to assume some parsimonious structure on either $\mathbf{z}$, the filter $\mathbf{H}(\mathbf{S})$, or both, and then analyze the impact of those assumptions on the properties of $\mathbf{x}$. Standard assumptions have included $\mathbf{H}(\mathbf{S})$ being a bandlimited graph filter so that $\mathbf{x}$ is graph bandlimited [27], $\mathbf{H}(\mathbf{S})$ being low pass so that $\mathbf{x}$ is smooth [2], [28], [29], $\mathbf{z}$ being a white signal so that $\mathbf{x}$ is graph stationary [30], [31], or $\mathbf{z}$ being sparse so that $\mathbf{x}$ is a diffused graph signal [32], as well as combinations of those. More importantly, the combination of the generative

model $\mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{z}$ and one or more of the previous structural assumptions have been leveraged to successfully generalize a number of estimation and learning tasks to the graph domain. Early examples investigated in the literature included signal denoising, sampling and interpolation, input identification, blind deconvolution, dictionary design, semi-supervised learning, classification, and the generalization of stationarity to graph domains (see, e.g, [23] for a detailed review). While covering all of these tasks goes beyond the scope of this paper, we discuss next three illustrative milestones: (i) sampling and interpolation, (ii) source identification and blind deconvolution, and (iii) statistical descriptions of random graph signals.

We start with a simple sampling and interpolation setup that, due to is practical relevance, received early attention from multiple research groups [33]. Consider the sampling set $\mathcal{M} \subseteq \mathcal{V}$ with cardinality $M \leq N$, and define the selection matrix $\mathbf{\Phi}_{\mathcal{M}} \in \{0,1\}^{M \times N}$ as the M rows of the $N \times N$ identity matrix indexed by the set $\mathcal{M}$. The sampled signal $\mathbf{x}_{\mathcal{M}} := \mathbf{\Phi}_{\mathcal{M}}\mathbf{x}$ collects the values of the graph signal $\mathbf{x}$ at the vertex set $\mathcal{M}$. The goal is to use $\mathbf{x}_{\mathcal{M}}$ – along with $\mathbf{S}$ – to recover $\mathbf{x}$, leveraging the structure of the graph. Since the problem is ill posed, we need to assume and enforce some structure on $\mathbf{x}$. Two widely adopted approaches are to (i) assume that $\mathbf{x}$ is K-bandlimited; i.e., it is in the span of the first K eigenvectors of $\mathbf{S}$, for some $K < N$, or (ii) assume that the signal $\mathbf{x}$ is smooth with respect to the underlying graph, which can be generically modelled as the norm of $\|\mathbf{x} - \mathbf{H}(\mathbf{S})\mathbf{x}\|$ being small, where $\mathbf{H}(\mathbf{S})$ is a low pass filter that is tuned to promote a particular notion of smoothness. We denote the subspace of K-bandlimited signals by $\mathcal{X}(\mathbf{V}_K) := \{\mathbf{V}_K\boldsymbol{\beta} \text{ for all } \boldsymbol{\beta} \in \mathbb{R}^K\}$. The statement that $\mathbf{x} \in \mathcal{X}(\mathbf{V}_K)$ is equivalent to saying that $\mathbf{x}$ is generated via a graph filter with $\widehat{\mathbf{h}} = [\mathbf{1}_K^\top, \mathbf{0}_{N-K}^\top]^\top$. These two alternative assumptions lead to the following optimization problems for interpolation, respectively:

$$\mathbf{x}^* = \text{argmin}_{\mathbf{x} \in \mathcal{X}(\mathbf{V}_K)} \ \|\mathbf{x}_{\mathcal{M}} - \mathbf{\Phi}_{\mathcal{M}}\mathbf{x}\|_2^2 \quad \text{or} \quad \mathbf{x}^* = \text{argmin}_{\mathbf{x}} \|\mathbf{x}_{\mathcal{M}} - \mathbf{\Phi}_{\mathcal{M}}\mathbf{x}\|_2^2 + \alpha\|(\mathbf{I} - \mathbf{H}(\mathbf{S}))\mathbf{x}\|_2^2, \quad (3)$$

with the weight α controlling the trade-off between minimizing the smoothness of $\mathbf{x}^*$ and how similar $\mathbf{x}^*$ and $\mathbf{x}$ are for the nodes in $\mathcal{M}$. For bandlimited signals, if $M \geq K$ and $(\mathbf{\Phi}_{\mathcal{M}}\mathbf{V}_K)$ is full rank, the signal $\mathbf{x}$ can be identified from its samples $\mathbf{x}_{\mathcal{M}}$ via $\mathbf{x} = \mathbf{V}_K(\mathbf{\Phi}_{\mathcal{M}}\mathbf{V}_K)^\dagger\mathbf{x}_{\mathcal{M}}$ [27]. While this is also true for time signals, other popular results in classical SP – such as ideal low-pass filters being the optimal interpolators or regularly spaced sampling being optimal – do not hold true for the graph domain, due to the lack of regularity in $\mathcal{G}$. Regarding the second optimization problem in (3), the solution is $\mathbf{x}^* = \left(\mathbf{\Phi}_{\mathcal{M}}^\top\mathbf{\Phi}_{\mathcal{M}} + \alpha(\mathbf{I} - \mathbf{H}(\mathbf{S})^2)\right)^{-1}\mathbf{\Phi}_{\mathcal{M}}^\top\mathbf{x}_{\mathcal{M}}$. In this case, we can interpret $\mathbf{\Phi}_{\mathcal{M}}^\top\mathbf{x}_{\mathcal{M}}$ as a zero-padded graph signal that is smoothly diffused through the graph by $\left(\mathbf{\Phi}_{\mathcal{M}}^\top\mathbf{\Phi}_{\mathcal{M}} + \alpha(\mathbf{I} - \mathbf{H}(\mathbf{S})^2)\right)^{-1}$.

Using the model $\mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{z}$, source identification and blind deconvolution have also been generalized to the graph setting [32]. In both, the signal $\mathbf{z}$ is assumed to be sparse. For source identification, given a sampled version of $\mathbf{x}$, the goal is to identify the locations and non-zero values of $\mathbf{z}$, which can be viewed as source nodes whose inputs are diffused throughout the network represented by $\mathbf{S}$. For blind deconvolution, the goal is to use $\mathbf{x}$ to identify both the sparse input $\mathbf{z}$ and the generating filter $\mathbf{H}(\mathbf{S})$, with a classical assumption being that the coefficients $\mathbf{h}$ are sparse or that the filter has a parsimonious FIR/IIR structure. Inspired by those works, generalizations were also investigated for demixing setups where the aggregation of multiple signals is observed (e.g., the sum of several network processes, each with different sources and dynamics).

Our last example to illustrate the benefits of a common GSP framework is the statistical description of random graph signals. Characterizing random processes is a challenging task even for regular time-varying signals, with stationarity models excelling at finding a sweet spot between practical relevance and analytical tractability. With this is mind, multiple efforts were carried out to generalize the definition of stationarity to graph signals [30], [31]. The key step was to say that a zero-mean random graph signal $\mathbf{x}$ is stationary in $\mathbf{S}$ if it can be modeled as $\mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{z}$, with $\mathbf{z}$ being white. This is equivalent to saying that the covariance matrix $\mathbf{C}_{\mathbf{x}} = \mathbb{E}\left[\mathbf{x}\mathbf{x}^{\top}\right]$ can be written as a polynomial of the GSO $\mathbf{S}$, illustrating the relation between the underlying graph and the statistical properties of the graph signal, and establishing meaningful links with Gaussian Markov random fields that assume $\mathbf{S} = \mathbf{C}_{\mathbf{x}}^{-1}$. With this definition, counterparts to concepts and tools such as the power spectral density, the periodogram, the Wiener filter, and autoregressive-moving-average (ARMA) models were developed [30]. These developments provide new ways to design graph-based covariance estimators and denoise graph signals, as well as a rigorous framework to better model, understand, and control random processes residing on a graph.

We close this section by highlighting that, while some instances of the problems discussed had been investigated well before the GSP framework was put forth (e.g., denoising based on smooth priors given by powers of the Laplacian, or source identification based on graph diffusion processes), those early works were mostly disconnected and focused on particular setups. The advent of GSP and the use of a common language and theoretical framework served a number of purposes: (i) facilitating the identification of connections between and differences among existing works; (ii) bringing different research communities together; (iii) enabling the design of more complex processing architectures that use early works as building blocks; (iv) providing a new set of tools for graph signals based on the generalization of classical SP schemes to the graph domain; and (v) aiding the development of novel, theoretically-grounded solutions to graph-based problems that had been solved in a heuristic

manner.

III. The Impact of GSP on Data Science

GSP has transformed how the SP community deals with irregular geometric data; however, it has also contributed to areas that go beyond SP, having a significant impact on data science related disciplines. To illustrate this, we next review several of the data science problems where GSP-based approaches have made significant contributions.

A. Graph learning

The field of GSP was originally conceived with a given graph ($\mathcal{G}$ or $\mathbf{S}$) in mind. Such a graph could originate from a physical network, such as transportation, communication, social or structural brain networks for instance. However, in many applications, the graph is an implicit object that describes relations or levels of association between the variables. In some cases, the links of those graphs can be based on expert domain knowledge (e.g., activation properties in protein-to-protein networks), but in many other cases the graph must be inferred from the data itself. Examples include graphs for image processing where the edges are defined based on both pixel distance and intensity differences, a k-nearest neighbor (kNN) graph for semi-supervised learning where edges connect data points with similar sets of features, or correlation graphs for functional brain networks. In those cases, the problem to solve can be formulated as "given a collection of M graph signals $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_M] \in \mathbb{R}^{N \times M}$, find an $N \times N$ sparse graph matrix $\mathbf{S}$ describing the relations among the nodes of the graph". Clearly, such a problem is severely ill posed, and models to relate the properties of the graph and the signals are key to address it in a meaningful way.

Learning a graph from data is a topic on its own, with roots in statistics, network science and machine learning (see [11] and references therein). Initial approaches focused on the information associated with each node separately, so that the existence of the link (i, j) in the graph was decided based only on the ith and jth row of $\mathbf{X}$. Contemporary (more advanced) approaches look at the problem as finding a mapping from $\mathbf{X}$ to $\mathbf{S}$, with graphical lasso (GL) being the most prominent example. GL is tailored for Gaussian Markov random fields and sets the graph to a sparsified version of the precision matrix, so that $\mathbf{S} \approx (\frac{1}{M}\mathbf{X}\mathbf{X}^\top)^{-1}$ [11]. The contribution of GSP to the problem of graph learning [29], [34] falls into this second class of approaches, where the more sophisticated (spectral and/or polynomial) relations between the signals and the graph can be fully leveraged. One cluster of early GSP works focused on learning a graph $\mathbf{S}$ that made the signals in $\mathbf{X}$ smooth with respect to the learnt graph [28]. If smoothness is promoted using a Laplacian-based total

variation regularizer $\sum_{m=1}^{M} \mathbf{x}_m^\top \mathbf{L}\mathbf{x}_m$, the formulation leads to a kernel-ridge regression problem with the pseudoinverse of $\mathbf{L}$ as the kernel and, meaningful links with GL can be established [34]. A second set of GSP-based topology inference methods model the data $\mathbf{X}$ as resulting from a diffusion process over the sought graph $\mathbf{S}$ through a graph filter. The key questions when modeling the observations as $\mathbf{x}_m = \mathbf{H}(\mathbf{S})\mathbf{z}_m$ are then the assumptions (if any) about the diffusing filter $\mathbf{H}(\mathbf{S})$ and the input signals $\mathbf{z}_m$. Assuming the inputs $\mathbf{z}_m$ to be white, which is tantamount to assuming that the signals $\mathbf{x}_m$ are stationary in $\mathbf{S}$, leads to a model where the covariance (precision) matrix of the observations is a polynomial of the sought GSO $\mathbf{S}$, all having the same eigenvectors. This not only provides a common umbrella to several existing graph-learning methods, but also a new (spectral and/or polynomial) way to address graph estimation [35], [36]. Indeed, the fact of GSP offering a well understood framework to model graph signals has propelled the investigation of multiple generalizations of the above methods, tackling, e.g., directed graphs, causal structure identification, presence of hidden nodes whose signals are never observed, dynamic networks, multi-layer graphs, and nonlinear models of interaction. We refer the interested reader to [29] and references therein for more details.

B. Network science

As discussed above, advancements in network science informed subsequent developments in GSP. It is now also the case that GSP techniques have been used to address network science problems such as clustering and community mining. We mention three examples here. First, in [37], spectral graph wavelets are utilized to develop a new fast, multiscale community mining protocol. Second, by graph spectral filtering random graph signals, feature vectors can be efficiently constructed for each vertex in a manner such that the distances between vertices based on these feature vectors resemble the distances based on standard spectral clustering feature vectors. In [38], a detailed account is provided of how that approach and other new sampling and interpolation methods developed for GSP can be used to accelerate spectral clustering by avoiding k-means. Third, [39] uses spectral graph wavelets to learn structural embeddings that help identify vertices that have similar structural roles in the network, even though they may be distant in the graph.

C. Semi-supervised learning

The goal of semi-supervised learning (SSL) is to utilize a combination of labeled and unlabeled data to predict the labels of the unlabeled data points. The labels may be discrete (semi-supervised classification) or continuous (semi-supervised regression). Many of the graph-based SSL methods (e.g., [14]) investigated by the machine learning community in the early 2000s constructed an

undirected, weighted similarity graph with each vertex representing one data point (either labeled or unlabeled), and then diffused the known labels across the graph to infer the labels at the unlabeled vertices. This approach can also be thought of as compelling the vector of labels to be smooth with respect to the underlying graph. Mathematically, this results in optimization problems with at least two terms: a fitting term that ensures the vector of labels exactly or approximately matches the known labels on the vertices corresponding to the labeled data points, and a regularization term of the form $\mathbf{x}^\top \mathbf{H}(\mathbf{S})\mathbf{x}$ for some GSO $\mathbf{S}$ and (low pass) graph filter $\mathbf{H}(\mathbf{S})$ that enforces global smoothness of the signal [40, Sec. III.D].

Rather than enforcing global smoothness of the labels with respect to the underlying graph, another GSP approach to SSL is to encourage the labels to be piecewise-smooth with respect to the graph by modeling them as a sparse linear combination of graph wavelet atoms [41]. Regularization problems resulting from this approach feature the same fitting term as above, but the additional term in the objective function captures the sparsity prior through the norm (or mixed norm) of the coefficients used to synthesize the labels as a linear combination of the graph wavelets. Finally, in GSP parlance, SSL is intimately related to graph signal interpolation, so that most of the results regarding the sampling and reconstruction of (bandlimited) graph signals, can (and have been) applied to SSL.

D. Graph neural networks

Neural networks (NN) are non-linear data processing architectures composed of multiple layers, each of which combines (mixes) the inputs linearly via matrix multiplication and then applies a scalar nonlinear function to each of the entries of the output. The values of the mixing matrices $\{\mathbf{\Theta}_\ell\}_{\ell=1}^L$ are considered as the parameters of the architecture. To avoid an excess of parameters, a standard approach is to impose some parsimonious structure on the mixing matrices (e.g., Toeplitz, low-rank, sparse), giving rise to different families of NN. Given the success of NN – and convolutional NN in particular – in processing regular data such as speech and images, a natural question is how to generalize these architectures to data defined over irregular graph domains. In this context, the ML learning community investigated graph NN (GNN) that incorporate the graph ($\mathcal{G}$ or $\mathbf{S}$) into NN architectures in different ways [6], [42]. GSP offers a principled way to address this question, postulating that the matrices $\{\mathbf{\Theta}_\ell\}_{\ell=1}^L$ have the form of a graph filter $\{\mathbf{H}(\mathbf{S})_\ell\}_{\ell=1}^L$. This both offers a flexible way to incorporate the graph (with the selection of the GSO $\mathbf{S}$ being application dependent) and also provides a range of options for parameterizing the graph filter (e.g., polynomial filters, rational filters, diffusion filters). Similarly, a number of generalizations and novel architectures that

leverage GSP have been proposed, including pooling schemes based on sampling over graphs, graph recurrent NN, architectures defined over product graphs, and NN based on graphon filters [43]. GSP has not only provided a common framework to better understand the contributions of and links between many of the existing works, but has also facilitated novel contributions on subjects such as transferability, robustness, or sensitivity with respect to the graph [44].

E. Graph-time processing

In many applications, a time series – as opposed to a scalar value – is observed at each node of the graph $\mathcal{G}$. If the length of each time series is T, the data at hand can be arranged in the form of a matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T] \in \mathbb{R}^{N \times T}$, which can be viewed as a collection of N times series (one per node of the graph), as a collection of T graph signals, or as a single signal $\text{vec}(\mathbf{X}) \in \mathbb{R}^{NT}$ that varies across both time and the nodes of the graph $\mathcal{G}$. The first approaches to handle time-varying graph signals were based on *product graphs* that combine a graph of the vertices with a graph for the time domain (e.g., a directed cycle graph $\mathcal{G}_{dc}$) to obtain a single larger graph $\mathcal{G} \times \mathcal{G}_{dc}$ with NT nodes [45], [46]. This interpretation allows for the use of standard GSP tools such as the GFT transform and graph filters, with the joint GFT being the Kronecker product of the original GFT $\mathbf{V}^{-1}$ and the DFT matrix $\mathbf{F}^H$, and the joint GSO some chosen product (e.g., Kronecker, Cartesian, strong) of the respective GSOs. Indeed, the joint spectrum of the time-varying graph signal $\text{vec}(\mathbf{X})$ can be analyzed this way, and joint graph-time filters can be adopted for its denoising or interpolation. In their most general form, those filters need not be separable over the graph and time domains, thereby increasing their modeling and processing potential.

Later, vector autoregressive (VAR) processes were considered for graph-time processing. A VAR models a vector process by expressing the current vector as a matrix-weighted version of past vectors plus some innovation, i.e., $\mathbf{x}_t = \sum_{p=1}^{P} \mathbf{A}_p \mathbf{x}_{t-p} + \mathbf{e}_t$. Considering that the vectors we are handling are graph signals, the underlying graph structure can be incorporated in such VAR models in different ways, leading to different GSP extensions. One direction is to replace the matrix weights by graph filters, i.e., $\mathbf{A}_p = \mathbf{H}_p(\mathbf{S})$, leading to *graph VAR processes* [47]. In such models, the graph filters can be implemented in the graph frequency domain or as a polynomial of the GSO in the vertex domain. Furthermore, causal models have been assumed where the polynomial order of the graph filter cannot be larger than the time delay on which the filter operates [48]. Another extension of VAR models also considers the interaction between the different nodes of the current vector, i.e., $\mathbf{x}_t = \mathbf{A}_0 \mathbf{x}_t + \sum_{p=1}^{P} \mathbf{A}_p \mathbf{x}_{t-p} + \mathbf{e}_t$, where $\mathbf{A}_0$ has a zero diagonal. It further enforces sparsity on all the matrix weights. In such *structural VAR processes* [49], the matrix weights can be viewed as graph

adjacency matrices that link the current data on a node with past data on the same node as well as with current and past data on neighboring nodes. Extensions to non-linear versions have also been considered.

IV. The Value of GSP in Science and Engineering Applications

Not surprisingly, GSP methods have been applied to engineering networks where a clear definition of the graph follows from a physical network. These include communication networks (e.g., developing distributed schemes to estimate the channels), smart grids, power networks, (e.g., designing distributed resource allocation algorithms for power flow), water networks, and transportation networks (e.g., developing graph-based architectures to predict traffic delay). Similarly, GSP has also contributed to applications where the network is not explicitly observable but can be inferred from additional information, such as social networks, meteorological prediction, genetics, and financial engineering. While all of the previous examples are meaningful and relevant, we briefly highlight here the two areas with the largest and most consistent GSP activity over the past decade: neuroscience and image and video processing.

A. Applications to neuroscience

Graphs have a long history in neuroscience, since they can be used to represent different relationships and pairwise connections between regions of the brain, taking each region to be a vertex [15]. An anatomical brain graph captures structural connections between the regions, as measured, e.g., via fiber tracts in white matter captured through diffusion magnetic resonance imaging. A functional brain graph, on the other hand, aims to capture pairwise interdependencies between activity that is measured in the different brain regions. Identifying the functional brain graph has been studied extensively for different reasons and with different modalities, the most common of which is functional magnetic resonance imaging (fMRI). Often, such studies also involve the estimation of dynamic graphs [50], [51]. During a sequence of task and rest periods, it has for instance been shown that the on-task and off-task functional brain graphs differ substantially [50]. Recent work also demonstrates that dynamics in the functional brain graph even exist during resting state fMRI, with meaningful correlations with electroencephalograph (EEG), demographic, and behavioral data [51].

Interestingly, most graph-based approaches in neuroscience consist of first identifying a brain graph and then using graph-theoretical and network-science tools to analyze its properties. From this point of view, GSP tools can be (and have been) leveraged for learning brain graphs [52]. However, GSP really shines when it comes to analyzing how the measured activity pattern – the brain signal –

behaves in relation to a brain graph (either anatomical or functional, related to one or multiple subjects). In other words, GSP provides a technology to merge the brain function, contained in the brain signal, with the brain graph (see [52] and references therein). Specifically, the GFT has been used to analyze cognitive behavior. For example, [53] shows that that there is a relation between the energy of the high frequency content of an fMRI signal and the attention switching ability of an individual. There is further research from the same group that states that, when learning a task, the correlations between the learning rate and the energies of the low/high frequency content of an fMRI signal change with the exposure time; i.e., they depend on how familiar we are with the task. In addition to the GFT, graph wavelets and Slepians have been used to reveal localized frequency content in the brain [52], and graph filters have been used as diffusion operators to model disease progression in dementia. While these results demonstrate the potential GSP has for neuroscience, we believe this pairing is still in its infancy and there is plenty of room for exploration.

B. Applications to image and video processing

As noted earlier, widely used techniques in image and video processing, including transforms such as the DCT and the KLT, segmentation methods, and image filtering can be interpreted from a GSP perspective [54]. In recent years, the emergence of a broader understanding of GSP has led to a further evolution of how graph-based approaches are used for image processing. As an example, while the DCT or the asymmetric discrete sine transform (ADST) are formed by the eigenvectors of path graphs with equal edge weigths, extensions have been proposed where graph edges with lower weights can be introduced in between pixels corresponding to image contours [55]. In these approaches, as in input-dependent image filtering [56], the image structure is first analyzed (e.g., contours detected) and then transforms adapted to the image characteristics are selected, with the choice of transform sent as side information.

A particularly promising application of GSP methods is to point cloud processing and compression. Each point in a point cloud is defined by its coordinates in 3D space and has associated with it an attribute (e.g., color or reflectance). While points are in a Euclidean domain, their positions, on the surfaces of the objects in the scene, are irregular and make it natural to develop a graph-based processing approach. Transforms have been proposed that leverage or are closely related to the GFT of a point cloud graph [57]. These methods are fundamental algorithms for geometry-based point cloud compression. Additionally, point cloud processing has become a major application domain for graph machine learning, with applications in areas such as denoising [58].

V. The Future Ahead

The focus of this article has been on reviewing the early results and growth of GSP, with an eye not only on the SP community but also on the applications and data science problems that have benefited from GSP. We close by discussing some of the emerging directions and open problems that we believe will shape the future of the field.

One emerging area in the field of GSP is **dynamic graphs**, more specially, how to estimate them and how to process time-varying graph signals residing on them. Graphs are rarely static; think for instance about social networks with new users or changing connections, or functional brain networks determined by a specific task that is carried out at a particular time instant. As a result, GSP tools, theory, and algorithms need to be extended to such scenarios. There is already quite some work on graph topology identification for dynamic graphs, but most of these methods link consecutive graphs in the cost function, making the problems computationally challenging [49], [29]. Adaptive methods (of the correction-only or prediction-correction type) try to tackle this issue, but tracking rates are still low. Processing signals residing on time-varying graphs has not been studied in depth, and this is clearly an area where many opportunities arise.

Extending GSP to **higher-order graphs** is another important future direction. Some applications are characterized by a graph domain where more than two nodes can interact; think for instance about a co-authorship network where groups of co-authors that collaborated on a paper are linked together, or about movie graphs in recommender systems, where movies starring the same actor form a group. Such graphs where an edge can join more than two nodes are called higher-order graphs. Popular abstractions of higher-order graphs are *simplicial complexes* and *cell complexes*. A simplicial/cell complex is a collection of subsets of the set of nodes satisfying certain properties. Whereas in a simplicial complex the subsets satisfy the subset inclusion property (e.g., there need to be links between each pair of the three co-authors of a paper), in a cell complex, they do not. However, both types of complexes share a similar recursive relationship between the higher-order Laplacians, leading to a hierarchical processing architecture that can process node signals over edges, edge signals over triangles/polygons (for a simplicial/cell complex), and so on. A less restrictive representation of a higher order graph is a *hypergraph* $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \omega)$, where ω is a function that assigns a weight to each hyperedge in $\mathcal{E}$. Hyperedges can connect more than two vertices in $\mathcal{V}$. Some recent overviews on higher-order networks, with focuses on GSP and network science, respectively, can be found in [59], [60]. There are still many open issues in higher order GSP, including the exploration of connections to adjacent fields such as topological data analysis and computational geometry.

Many other open problems – extending GSP to include uncertainty in the signals and graphs, design of exact and approximate Bayesian (recursive) estimators able to track variations across nodes and time, developing GSP models for categorical data, generalizing GSP results to continuous manifold (geometric) data, incorporating GSP tools into reinforcement learning and spatio-temporal control, etc. – are also expected to play important roles in the future of the discipline. If the first years of GSP combined theoretical developments with practical applications by placing a stronger focus on the former, we expect that the coming years will see an increased emphasis on applications, along with important efforts on learning and statistical schemes.

References

[1] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, pp. 129–150, Mar. 2011.

[2] 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, pp. 83–98, May 2013.

[3] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, pp. 1644–1656, Jan. 2013.

[4] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, pp. 3042–3054, June 2014.

[5] L. Stanković, D. Mandic, M. Daković, M. Brajović, B. Scalzo, S. Li, and A. G. Constantinides, "Data analytics on graphs," *Foundations and Trends® in Machine Learning*, vol. 13, 2020.

[6] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, pp. 18–42, Jul. 2017.

[7] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer, Berlin, 2001.

[8] D. M. Cvetković, M. Doob, and H. Sachs, *Spectra of Graphs: Theory and Application*. Academic Press, New York, 1980.

[9] G. Taubin, "A signal processing approach to fair surface design," in *Annu. Conf. Comp. Graphics Interactive Tech. (SIGGRAPH)*, pp. 351–358, 1995.

[10] A. Elmoataz, O. Lezoray, and S. Bougleux, "Nonlocal discrete regularization on weighted graphs: A framework for image and manifold processing," *IEEE Trans. Image Process.*, vol. 17, pp. 1047–1060, Jul. 2008.

[11] E. D. Kolaczyk, *Statistical Analysis of Network Data: Methods and Models*. Springer, 2009.

[12] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, no. 1–2, pp. 1–305, 2008.

[13] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comp.*, vol. 15, pp. 1373–1396, Jun. 2003.

[14] A. Smola and R. Kondor, "Kernels and regularization on graphs," in *Conf. Learning Theory (COLT)*, Aug. 2003.

[15] O. Sporns, *Networks of the Brain*. MIT press, 2010.

[16] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Intl. Conf. Comput. Vision (ICCV)*, Jan. 1998.

[17] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[18] M. Newman, *Networks*. Oxford Univ. Press, 2018.

[19] R. Pastor-Satorras and A. Vespignani, "Epidemic spreading in scale-free networks," *Phys. Rev. Lett.*, vol. 86, no. 14, p. 3200, 2001.

[20] M. Crovella and E. Kolaczyk, "Graph wavelets for spatial traffic analysis," in *IEEE Intl. Conf. Comput. Comms. (INFOCOM)*, Apr. 2003.

[21] R. R. Coifman and M. Maggioni, "Diffusion wavelets," *Appl. Comput. Harmon. Anal.*, vol. 21, pp. 53–94, Jun. 2006.

[22] M. Püschel and J. M. F. Moura, "Algebraic signal processing theory: Foundation and 1-D time," *IEEE Trans. Signal Proc.*, vol. 56, pp. 3572–3585, Aug. 2008. See also M. Püschel and J. M. F. Moura, "Algebraic signal processing theory: 1D space," *IEEE Trans. Signal Proc.*, vol. 56, pp. 3586–3599, Aug. 2008.

[23] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[24] D. I Shuman, "Localized spectral graph filter frames: A unifying framework, survey of design considerations, and numerical comparison," *IEEE Signal Process. Mag.*, vol. 37, pp. 43–63, Nov. 2020.

[25] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 4117–4131, 2017.

[26] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 274–288, 2016.

[27] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6510–6523, 2015.

[28] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, 2016.

[29] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Process. Mag.*, vol. 36, pp. 16–43, May 2019.

[30] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Trans. Signal Process.*, vol. 65, no. 22, pp. 5911–5926, 2017.

[31] N. Perraudin and P. Vandergheynst, "Stationary signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3462–3477, 2017.

[32] S. Segarra, G. Mateos, A. G. Marques, and A. Ribeiro, "Blind identification of graph filters," *IEEE Trans. Signal Process.*, vol. 65, no. 5, pp. 1146–1159, 2017.

[33] Y. Tanaka, Y. C. Eldar, A. Ortega, and G. Cheung, "Sampling signals on graphs: From theory to applications," *IEEE Signal Process. Mag.*, vol. 37, pp. 14–30, Nov. 2020.

[34] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 44–63, 2019.

[35] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 467–483, 2017.

[36] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat, "Characterization and inference of graph diffusion processes from observations of stationary signals," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 3, pp. 481–496, 2018.

[37] N. Tremblay and P. Borgnat, "Graph wavelets for multiscale community mining," *IEEE Trans. Signal Process.*, vol. 62, pp. 5227–5239, Oct. 2014.

[38] N. Tremblay and A. Loukas, "Approximating spectral clustering via sampling: a review," in *Sampling Techniques for Supervised or Unsupervised Tasks*, pp. 129–183, Springer, 2020.

[39] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining (KDD)*, pp. 1320–1329, 2018.

[40] D. I Shuman, P. Vandergheynst, D. Kressner, and P. Frossard, "Distributed signal processing via Chebyshev polynomial approximation," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, pp. 736–751, Dec. 2018.

[41] D. I Shuman, M. Faraji, and P. Vandergheynst, "Semi-supervised learning with spectral graph wavelets," in *Intl. Conf. Sampling Theory Appl. (SampTA)*, 2011.

[42] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Intl. Conf. Neural Inform. Process. Syst. (NeurIPS)*, p. 3844–3852, 2016.

[43] A. G. Marques, N. Kiyavash, J. M. F. Moura, D. V. D. Ville, and R. W. (Eds.), "Graph signal processing: Foundations and emerging directions," *IEEE Signal Process. Mag.*, vol. 37, Nov. 2020.

[44] L. Ruiz, F. Gama, and A. Ribeiro, "Graph neural networks: Architectures, stability, and transferability," *Proc. IEEE*, vol. 109, no. 5, pp. 660–682, 2021.

[45] A. Sandryhaila and J. M. F. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Process. Mag.*, vol. 31, pp. 80–90, Sep. 2014.

[46] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud, "A time-vertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs," *IEEE Trans. Signal Process.*, vol. 66, no. 3, pp. 817–829, 2018.

[47] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, "Forecasting time series with VARMA recursions on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 18, pp. 4870–4885, 2019.

[48] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 2077–2092, 2017.

[49] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proc. IEEE*, vol. 106, no. 5, pp. 787–807, 2018.

[50] R. P. Monti, P. Hellyer, D. Sharp, R. Leech, C. Anagnostopoulos, and G. Montana, "Estimating time-varying brain connectivity networks from functional MRI time series," *NeuroImage*, vol. 103, pp. 427–443, 2014.

[51] M. G. Preti, T. A. W. Bolton, and D. Van De Ville, "The dynamic functional connectome: State-of-the-art and perspectives," *NeuroImage*, vol. 160, pp. 41–54, 2017.

[52] W. Huang, T. A. W. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro, and D. Van De Ville, "A graph signal processing perspective on functional brain imaging," *Proc. IEEE*, vol. 106, no. 5, pp. 868–885, 2018.

[53] J. D. Medaglia, W. Huang, E. A. Karuza, A. Kelka, S. L. Thompson-Schill, A. Ribeiro, and D. S. Bassett, "Functional alignment with anatomical networks is associated with cognitive flexibility," *Nat. Hum. Behav.*, vol. 2, pp. 156–164, 2018.

[54] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng, "Graph spectral image processing," *Proc. IEEE*, vol. 106, no. 5, pp. 907–930, 2018.

[55] W. Hu, G. Cheung, A. Ortega, and O. C. Au, "Multiresolution graph Fourier transform for compression of piecewise smooth images," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 419–433, 2014.

[56] P. Milanfar, "A tour of modern image filtering: New insights and methods, both practical and theoretical," *IEEE Signal Process. Mag.*, vol. 30, pp. 106–128, Jan. 2013.

[57] R. L. De Queiroz and P. A. Chou, "Compression of 3D point clouds using a region-adaptive hierarchical transform," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3947–3956, 2016.

[58] D. Valsesia, G. Fracastoro, and E. Magli, "Deep graph-convolutional image denoising," *IEEE Trans. Image Process.*, vol. 29, pp. 8226–8237, 2020.

[59] M. T. Schaub, Y. Zhu, J.-B. Seby, T. M. Roddenberry, and S. Segarra, "Signal processing on higher-order networks: Livin' on the edge ... and beyond," *Signal Process.*, vol. 187, p. 108149, 2021.

[60] F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G. Young, and G. Petri, "Networks beyond pairwise interactions: Structure and dynamics," *Phys. Rep.*, vol. 874, pp. 1–92, 2020.