TU Delft

Delft University of Technology

Finite Impulse Response Filters for Simplicial Complexes

Yang, Maosheng; Isufi, Elvin; Schaub, Michael T. ; Leus, Geert

DOI
10.23919/EUSIPCO54536.2021.9616185

Publication date
2021

Document Version
Final published version

Published in
2021 29th European Signal Processing Conference (EUSIPCO)

Citation (APA)
Yang, M., Isufi, E., Schaub, M. T., & Leus, G. (2021). Finite Impulse Response Filters for Simplicial Complexes. In *2021 29th European Signal Processing Conference (EUSIPCO): Proceedings* (pp. 2005-2009). Article 9616185 IEEE. https://doi.org/10.23919/EUSIPCO54536.2021.9616185

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

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

Finite Impulse Response Filters for Simplicial Complexes

Maosheng Yang
Dept. of Intelligent Systems
Delft University of Technology
Delft, The Netherlands
m.yang-2@tudelft.nl

Elvin Isufi
Dept. of Intelligent Systems
Delft University of Technology
Delft, The Netherlands
e.isufi-1@tudelft.nl

Michael T. Schaub
Dept. of Computer Science
RWTH Aachen University
Aachen, Germany
schaub@cs.rwth-aachen.de

Geert Leus
Dept. of Microelectronics
Delft University of Technology
Delft, The Netherlands
g.j.t.leus@tudelft.nl

***Abstract*—In this paper, we study linear filters to process signals defined on simplicial complexes, i.e., signals defined on nodes, edges, triangles, etc. of a simplicial complex, thereby generalizing filtering operations for graph signals. We propose a finite impulse response filter based on the Hodge Laplacian, and demonstrate how this filter can be designed to amplify or attenuate certain spectral components of simplicial signals. Specifically, we discuss how, unlike in the case of node signals, the Fourier transform in the context of edge signals can be understood in terms of two orthogonal subspaces corresponding to the gradient-flow signals and curl-flow signals arising from the Hodge decomposition. By assigning different filter coefficients to the associated terms of the Hodge Laplacian, we develop a subspace-varying filter which enables more nuanced control over these signal types. Numerical experiments are conducted to show the potential of simplicial filters for sub-component extraction, denoising and model approximation.**

***Index Terms*—Hodge decomposition, Hodge Laplacian, simplicial complexes, simplicial filters, Topological signal processing.**

I. Introduction

Graph signal processing (GSP) has emerged over the last years as a powerful tool to deal with high-dimensional signals defined on non-Euclidean domains [1], [2]. Using the perspective of GSP, notions like the Fourier transform, filters, and signal sampling have been extended to the domain of graphs [3], [4], [5]. These definitions can also be leveraged to understand graph neural networks [6]. GSP not only provides a generalization of signal processing for time series and images, but specifies an explicit model for the underlying signal dependencies in terms of a graph.

By construction, graphs model dependencies between node data via edges, which encode pairwise relations between nodes. In certain scenarios, however, we may be interested in specifying multi-way relationships between subsets of nodes. For instance, in a coauthorship network, we want to describe collaborations between more than two authors [7]. Similarly, in a social network, we want to capture the connections between groups of friends rather than two people [8]. To deal with such settings, researchers have introduced *hypergraph signal processing* as one paradigm, in which we model group relationships through hyperedges [9], [10], [11]. Other works [12], [13] use the Volterra model to describe higher-order relations.

In parallel, *topological signal processing* (TSP) has been proposed to analyze signals defined over topological spaces, especially in the form of simplicial complexes composed of nodes, edges, and triangles, etc. [11], [14], [15]. Important examples of such signals defined on simplices include flow signals over edges, like traffic flows in a transportation network or data flows in a communication network. Edge flows have also been considered in the context of statistical ranking or to process currency exchange market rates [16]. To process such flow signals, a number of procedures have been developed. For instance, [11], [17] consider the problem of flow denoising by solving a regularized optimization problem, which is equivalent to a low-pass filter in the edge space. Similarly, flow interpolation and edge sampling have been studied in [18], and random walks have been extended from the node space to the edge space in [19]. There exist even certain neural network architectures for data defined on the edge space of a simplicial complex [20]. However, more explicit treatments of linear filtering and filter design on simplicial complexes, have so far not been provided in the literature. In this paper, we thus define, design and analyze finite impulse response (FIR) filters for simplicial complexes, as fundamental building blocks for signal processing on simplicial complexes.

M. Yang is supported by the TU Delft AI Labs Programme. M. T. Schaub received funding from the Ministry of Culture and Science (MKW) of the German State of North Rhine-Westphalia ("NRW Rückkehrprogramm").

Contribution and paper outline. We re-examine the Fourier transform for simplicial complexes (SCs) [11], [14], [17], and define two types of simplicial frequencies, measuring different properties of signals. Using this perspective we propose an FIR filter for SCs, which employs the Hodge Laplacian associated to the SC as a shift operator, in lieu of the graph Laplacian used analogously for graph signals. Focusing on the processing of edge flows, we study this local shift operator, which enables us to implement convolutions of flows on SCs locally with small computational complexity. Using the Hodge decomposition associated to the space of edge flows, we show the capability of such filters to modulate signals at different frequencies. Further, by assigning individual filter coefficients to the lower and upper adjacent coupling terms of the Hodge Laplacian, we create a more expressive linear filter that can modulate gradient and curl flows more precisely, leading to an improved filter performance. We illustrate our results by several numerical experiments.

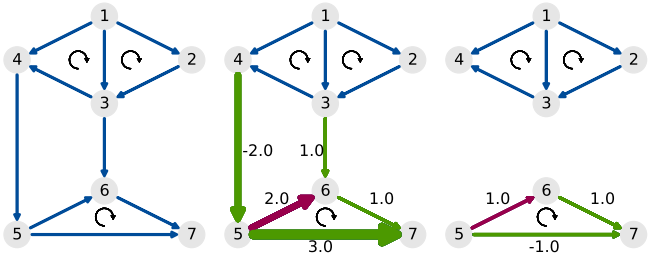
The remainder of the paper is structured as follows. In Section II we review simplicial complexes, simplicial signals, the Hodge Laplacian and the Hodge decomposition. In Section III, we recall the Fourier transform of simplicial signals and define two types of simplicial frequencies. The FIR filter on SCs is proposed in Section IV where simplicial signal shifting, spectral analysis and filter design are investigated. Then the subspace-varying filter is proposed with improved performance. We conduct experiments in synthetic and real world networks in Section V before concluding the paper.

II. Background

Simplicial complexes and signals. Given a finite set of vertices $\mathcal{V}$, a k-simplex $\mathcal{S}^k$ is a subset of $\mathcal{V}$ with cardinality $k+1$. A subset of a k-simplex with cardinality k is called a *face*. The number of faces of a k-simplex is $k+1$. A *coface* of a k-simplex is a $(k+1)$-simplex that includes this k-simplex. A node is a 0-simplex, an edge is a 1-simplex, and a triangle is a 2-simplex. For an edge, its incident nodes are faces. The edges connecting a node with its neighbors, are the cofaces of that node [14], [15].

A simplicial complex $\mathcal{X}$ is a set of simplices such that for any k-simplex $\mathcal{S}^k$ in $\mathcal{X}$, any subset of $\mathcal{S}^k$ must also be in $\mathcal{X}$. We denote the

Authorized licensed use limited to: TU Delft Library. Downloaded on December 09,2021 at 13:34:32 UTC from IEEE Xplore. Restrictions apply.



(a) An all-one flow (b) Shifting by $\mathbf{L}_{1,\ell}$ (c) Shifting by $\mathbf{L}_{1,u}$

Fig. 1. (a) An all-one flow defined on the edges of a simplicial complex (SC) with three 2-simplices $\{1,2,3\},\{1,3,4\},\{5,6,7\}$. The flow magnitude is displayed by the edge width. (b-c) One-step $\mathbf{L}_{1,\ell}$ and $\mathbf{L}_{1,u}$ shifted results shown on edges only in red and green. A negative value indicates an opposite relative direction and no link means zero flow on that edge. (b) In the shifting by $\mathbf{L}_{1,\ell}$, edge $(5,6)$ (in red) locally collects the information from its one-hop lower neighbors (in green) and combines with its own state. (c) In the shifting by $\mathbf{L}_{1,u}$, edge $(5,6)$ (in red) locally collects the information from its one-hop upper neighbors (in green) and combines with its own state. Thus, by summing (b) and (c), the one-step $\mathbf{L}_1$ shifted result on edge $(5,6)$ is 3.

number of k-simplices in $\mathcal{X}$ by N_k. A graph is an SC with simplices of maximal cardinality 2. Another example is shown in Fig. 1a, where nodes are the 0-simplices, edges are the 1-simplices, and triangles are the 2-simplices [11]. Note that every simplex is equipped with a reference orientation, as indicated by arrows in Fig. 1a.

In a simplicial complex, we define a k-simplicial signal as a mapping from the k-simplices to the real space $\mathbb{R}^{N_k}$. For example, $\mathbb{R}^{N_0}$ is the graph signal space in GSP, and $\mathbb{R}^{N_1}$ is the space of edge flows. For an edge flow $\mathbf{f} = [f_1,\dots,f_{N_1}]^\top \in \mathbb{R}^{N_1}$, the sign of its entry denotes the direction of the flow [11], [15], relative to a chosen (arbitrary) reference orientation. For example, the direction of a random flow is shown by the edge arrows in Fig. 1b.

Incidence matrices for simplicial complexes. The relationships between $(k-1)$- and k-simplices can be described via the incidence matrix $\mathbf{B}_k$. The rows of $\mathbf{B}_k$ are indexed by $(k-1)$-simplices and the columns by k-simplices. The matrix $\mathbf{B}_1$ is simply the node-edge incidence matrix, and $\mathbf{B}_2$ is the edge-triangle incidence matrix. We construct $\mathbf{B}_2$ as follows: if an edge is oriented along with its coface, then the corresponding entry in $\mathbf{B}_2$ is 1; if it is anti-aligned, the entry is -1; otherwise it is zero. For example, in Fig. 1a, edge $\{1,2\}$ is aligned with triangle $\{1,2,3\}$, while edge $\{1,3\}$ is not.

The Hodge Laplacian and the Hodge decomposition. An algebraic tool to analyze simplicial signals is the *k-th Hodge Laplacian*, given by $\mathbf{L}_k = \mathbf{B}_k^\top\mathbf{B}_k + \mathbf{B}_{k+1}\mathbf{B}_{k+1}^\top$, where we define the lower Laplacian $\mathbf{L}_{k,\mathrm{l}} \triangleq \mathbf{B}_k^\top\mathbf{B}_k$, and the upper Laplacian $\mathbf{L}_{k,\mathrm{u}} \triangleq \mathbf{B}_{k+1}\mathbf{B}_{k+1}^\top$. The 0-th Hodge Laplacian corresponds to the graph Laplacian $\mathbf{L}_0 = \mathbf{B}_1\mathbf{B}_1^\top$ used in GSP. While we consider unweighted Hodge Laplacians in this paper, weighted variants also exist [19].

Hodge Laplacians admit a *Hodge decomposition*, by which the simplicial signal space can be decomposed into three orthogonal subspaces $\mathbb{R}^{N_k} = \mathrm{im}(\mathbf{B}_k^\top) \oplus \mathrm{im}(\mathbf{B}_{k+1}) \oplus \ker(\mathbf{L}_k)$, where $\oplus$ is the direct sum of vector spaces and $\mathrm{im}(\cdot)$ and $\ker(\cdot)$ are the *image* and *kernel* spaces of a matrix. For $k=1$, these subspaces carry the following interpretations [14], [19].

Gradient space. The incidence matrix $\mathbf{B}_1$ acts as the divergence operator mapping an edge flow $\mathbf{f}$ into a node signal $\mathbf{B}_1\mathbf{f}$, which computes the net flow of each node. Its adjoint $\mathbf{B}_1^\top$ can differentiate a node signal $\mathbf{v}$ along the edges to induce an edge flow $\mathbf{B}_1^\top\mathbf{v}$, i.e., the gradient operation. Thus, any flow $\mathbf{f}_G \in \mathrm{im}(\mathbf{B}_1^\top)$ can be written as the gradient of a node signal $\mathbf{v}$, i.e., $\mathbf{f}_G = \mathbf{B}_1^\top\mathbf{v}$. We call $\mathbf{f}_G$ a *gradient flow*. The subspace $\mathrm{im}(\mathbf{B}_1^\top)$ is defined as the *gradient space*.

Curl space. The incidence matrix $\mathbf{B}_2$ can induce an edge flow from a triangle signal $\mathbf{t}$ by $\mathbf{B}_2\mathbf{t}$. Its adjoint $\mathbf{B}_2^\top$ is known as the *curl operator*. By applying it to an edge flow $\mathbf{f}$ as $\mathbf{B}_2^\top\mathbf{f}$, we can compute the flow circulating along the triangles. Thus, any flow $\mathbf{f}_C \in \mathrm{im}(\mathbf{B}_2)$ can be induced by a triangle flow $\mathbf{t}$ as $\mathbf{f}_C = \mathbf{B}_2\mathbf{t}$. We call $\mathbf{f}_C$ a *curl flow*. The subspace $\mathrm{im}(\mathbf{B}_2)$ is defined as the *curl space*.

Harmonic space. The remaining space $\ker(\mathbf{L}_1)$ is known as the *harmonic space*. Any flow $\mathbf{f}_H \in \ker(\mathbf{L}_1)$ is divergence and curl free, i.e., it is flow conservative. That is, the net flow at every node is zero and the flow circulating along the triangles is also zero.

Since $\mathbf{B}_1\mathbf{B}_2 = \mathbf{0}$, any gradient flow $\mathbf{f}_G$ satisfies $\mathbf{B}_2^\top\mathbf{f}_G = \mathbf{0}$, i.e., it does not include any circular flows along triangles and is accordingly called *curl-free*. The space orthogonal to the gradient space, i.e., $\ker(\mathbf{B}_1) = \mathrm{im}(\mathbf{B}_2) \oplus \ker(\mathbf{L}_1)$, is called the *cycle space*. Any flow $\mathbf{f}$ in this space fulfills $\mathbf{B}_1\mathbf{f} = \mathbf{0}$, and is thus called *divergence-free*. Note that the cycle space consists of both the curl space and the harmonic space discussed above.

III. Simplicial Fourier Transform

In this section, we recall the Fourier transform of simplicial signals [11], [14] and define two types of simplicial frequencies.

Simplicial Fourier transform. The k-th Hodge Laplacian has the spectral decomposition $\mathbf{L}_k = \mathbf{U}_k\mathbf{\Lambda}_k\mathbf{U}_k^\top$, where the matrix $\mathbf{U}_k = [\mathbf{u}_{k,1},\dots,\mathbf{u}_{k,N_k}]$ collects the eigenvectors and the diagonal matrix $\mathbf{\Lambda}_k = \mathrm{diag}(\lambda_{k,1},\dots,\lambda_{k,N_k})$ the corresponding eigenvalues. Given a signal $\mathbf{x}^k \in \mathbb{R}^{N_k}$ defined on the k-simplices, its embedding by the simplicial Fourier Transform (SFT) is $\tilde{\mathbf{x}}^k = \mathbf{U}_k^\top\mathbf{x}^k$, i.e., the projection of $\mathbf{x}^k$ on $\mathbf{U}_k$. The inverse SFT is $\mathbf{x}^k = \mathbf{U}_k\tilde{\mathbf{x}}^k$ [14]. The STF of $\mathbf{L}_0$ corresponds to the graph Fourier transform [3].

The eigenvectors of $\mathbf{L}_1$ span the three subspaces that appeared in the Hodge decomposition [11]: (i) the gradient space $\mathrm{im}(\mathbf{B}_1^\top)$ is spanned by a set of eigenvectors $\mathbf{U}_G$ of $\mathbf{L}_{1,\mathrm{l}}$ with positive eigenvalue; (ii) the curl space $\mathrm{im}(\mathbf{B}_2)$ is spanned by a set of eigenvectors $\mathbf{U}_C$ of $\mathbf{L}_{1,\mathrm{u}}$ with positive eigenvalue; and (iii) the harmonic space $\ker(\mathbf{L}_1) = \ker(\mathbf{L}_{1,\mathrm{l}}) \cap \ker(\mathbf{L}_{1,\mathrm{u}})$ is spanned by the eigenvectors $\mathbf{U}_H$ of $\mathbf{L}_1$ with zero eigenvalue. Moreover, the gradient and curl space span the image of $\mathbf{L}_1$, i.e., $\mathbf{U}_G \oplus \mathbf{U}_C = \mathrm{im}(\mathbf{L}_1)$. This correspondence between eigenvectors of the Hodge Laplacian and the Hodge decomposition exists in general. However, we focus on the edge space in this paper.

Finally, since the eigenvalues of $\mathbf{L}_k$ are all non-negative they can be naturally interpreted in terms of a frequency. However, unlike in the case of graphs [3] there are two types of eigenvectors for $\mathbf{L}_k$. For $k=1$, we thus distinguish the following types of frequencies:

Gradient frequencies. For an eigenvector $\mathbf{u}_G$ of $\mathbf{L}_1$ in the gradient space $\mathrm{span}(\mathbf{B}_1^\top)$, the corresponding eigenvalue is $\lambda_G = \mathbf{u}_G^\top\mathbf{L}_1\mathbf{u}_G = \|\mathbf{B}_1\mathbf{u}_G\|_2^2$, which is an ℓ_2-norm divergence measure for $\mathbf{u}_G$. Thus, the eigenvectors in $\mathbf{U}_G$ corresponding to a large eigenvalue λ_G have a large quadratic measure of the divergence. If the SFT of an edge flow has a large projection on such eigenvectors, we say it has a high gradient frequency. We call the eigenvalues λ_G associated to the gradient space $\mathbf{U}_G$ *gradient frequencies*.

Curl frequencies. For an eigenvector $\mathbf{u}_C$ in the curl space $\mathrm{span}(\mathbf{B}_2)$, its corresponding eigenvalue can be written as $\lambda_C = \mathbf{u}_C^\top\mathbf{L}_1\mathbf{u}_C = \|\mathbf{B}_2^\top\mathbf{u}_C\|_2^2$, which is an ℓ_2-norm curl measure for $\mathbf{u}_C$. Thus, eigenvectors corresponding to a large λ_C have a large curl quadratic measure. We call the eigenvalues λ_C associated to the curl space *curl frequencies*.

Harmonic frequencies. Finally, we call the zero eigenvalues *harmonic frequencies*. The multiplicity of zero frequencies is equal to the number of 1-dim holes (cycles) in a SC [15]. If an edge flow is contained in the harmonic space, we say it is a harmonic flow.

Authorized licensed use limited to: TU Delft Library. Downloaded on December 09,2021 at 13:34:32 UTC from IEEE Xplore. Restrictions apply.

For convenience, we collect gradient frequencies in the set $\mathcal{Q}_G = \{\lambda_{G,1}, \ldots, \lambda_{G,N_G}\}$, curl frequencies in the set $\mathcal{Q}_C = \{\lambda_{C,1}, \ldots, \lambda_{C,N_C}\}$ and harmonic frequencies (which are just zeros) in the set $\mathcal{Q}_H$. Thus, given any edge flow $\mathbf{f} \in \mathbb{R}^{N_1}$, we can use the SFT to define three embeddings by projecting $\mathbf{f}$ onto each of the subspaces defined by the Hodge decomposition: i) the gradient embedding: $\tilde{\mathbf{f}}_G = \mathbf{U}_G^\top \mathbf{f}$; ii) the curl embedding: $\tilde{\mathbf{f}}_C = \mathbf{U}_C^\top \mathbf{f}$; iii) the harmonic embedding: $\tilde{\mathbf{f}}_H = \mathbf{U}_H^\top \mathbf{f}$. If we can control these embeddings for any edge flow, we can achieve the desired *filtering*.

Example 1. Consider an edge flow containing mainly gradient embeddings with high frequency, i.e., induced by a potential on the nodes (Fig. 2a). If this edge flow is corrupted by white noise, the filtering proposed in [17], will not provide a good estimate (Fig. 2d) of the true flow as it penalizes both high gradient and curl frequencies. Similar to the filter (Fig. 2c) regularized by $\mathbf{f}^\top \mathbf{L}_1 \mathbf{f}$ proposed in [11]. We will see that our simplicial filters can solve such problem.

Remark. In GSP, graph frequencies measure the signal smoothness over the graph, while simplicial frequencies measure two types of signal properties, which make filtering in the spectral domain more involved. For $k = 1$, the gradient frequency measures the net flow passing through the nodes, while the curl frequency measures the flow circulating along the triangles. The harmonic (zero) frequencies indicate the solenoidal and irrotational properties, i.e., the flow conservation [14], [11]. Specifically, observe that a gradient frequency can be larger or smaller than a curl frequency as both frequencies correspond to different, orthogonal features of an edge flow. Thus, a low- or high-pass filtering without reference to these type of frequencies thus mix different features of an edge-flow signal. Filters in simplicial signal spaces thus generally would be expected to tune gradient, curl and harmonic components as required by the user.

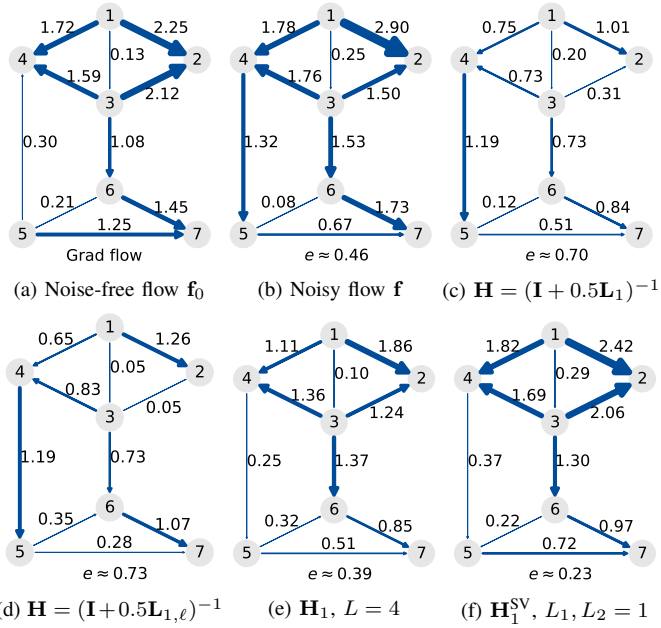
IV. Simplicial FIR Filter

As alluded to above, a linear filter of a flow signal amounts to a map that modulates the gradient, curl and harmonic embeddings of any flow. Such a filter may, e.g., extract the gradient component, or remove the harmonic component of a flow. While the filter will be dependent on the simplicial complex, it should be independent of the simplicial signals it is applied to. To achieve this, we propose the following linear simplicial FIR filter:

$$\mathbf{H}_1 = \sum_{l=0}^{L-1} h_l \mathbf{L}_1^l = \sum_{l=0}^{L-1} h_l (\mathbf{B}_1^\top \mathbf{B}_1 + \mathbf{B}_2 \mathbf{B}_2^\top)^l, \tag{1}$$

where L is the *filter length*, and $\mathbf{h} = [h_0, \ldots, h_{L-1}]^\top$ stacks the *filter coefficients*. Note that the filter has an analogous form to graph filters [3], [4], and can be extended to the k-simplicial space by using $\mathbf{L}_k$. To understand the effect of applying the filter (1) to an input signal, we first define the neighborhood set for k-simplices.

Simplicial neighborhood. For the i-th k-simplex $\mathcal{S}_i^k$ in a SC, we define its *lower neighborhood* $\mathcal{N}_{\mathrm{l},i}$ as the set of k-simplices sharing a common face with it. Similarly, the *upper neighborhood* $\mathcal{N}_{\mathrm{u},i}$ of $\mathcal{S}_i^k$ collects the k-simplices sharing a common coface with $\mathcal{S}_i^k$. The lower and upper neighbors of $\mathcal{S}_i^k$ are encoded in the nonzero off-diagonal elements of $\mathbf{L}_{1,\mathrm{l}}$ and $\mathbf{L}_{1,\mathrm{u}}$, respectively. More specifically, the cardinality of $\mathcal{N}_{\mathrm{l},i}$ (called *lower degree*) and the cardinality of $\mathcal{N}_{\mathrm{u},i}$ (called *upper degree*), are equal to the numbers of nonzero off-diagonal elements of the i-th row/column of $\mathbf{L}_{1,\mathrm{l}}$ and $\mathbf{L}_{1,\mathrm{u}}$, respectively. For example, for edge $(5,6)$ in Fig. 1a, we can see that $\mathcal{N}_{\mathrm{l}} = \{(4,5),(3,6),(5,7),(6,7)\}$, while the upper neighborhood is $\mathcal{N}_{\mathrm{u}} = \{(5,7),(6,7)\}$, as they share a triangle with edge $(5,6)$. The corresponding row/column in $\mathbf{L}_{1,\mathrm{l}}$ and $\mathbf{L}_{1,\mathrm{u}}$ have respectively 4 and 2 nonzero off-diagonal elements. Note that these local lower and upper connections lead to the sparsity of the Hodge Laplacian.



(a) Noise-free flow $\mathbf{f}_0$ (b) Noisy flow $\mathbf{f}$ (c) $\mathbf{H} = (\mathbf{I} + 0.5\mathbf{L}_1)^{-1}$

(d) $\mathbf{H} = (\mathbf{I} + 0.5\mathbf{L}_{1,\ell})^{-1}$ (e) $\mathbf{H}_1$, $L = 4$ (f) $\mathbf{H}_1^{\mathrm{SV}}$, $L_1, L_2 = 1$

Fig. 2. Gradient flow denoising on a SC (cf. Section V). The estimation error e is the normalized RMSE. (a) Edge flow $\mathbf{f}_0$ induced by a node signal. (b) The noisy observation $\mathbf{f}$. (c) Denoised by the filter in [11]. (d) Denoised by the filter in [17]. (e) Denoised by filter (1) with length $L = 4$. (f) Denoised by the subspace-varying filter (5) with $L_1 = L_2 = 1$.

Simplicial signal shift. Applying the filter (1) to an input signal $\mathbf{f}$, i.e., $\mathbf{H}_1\mathbf{f}$ can be understood in terms of the basic signal shift operation $\mathbf{L}_1\mathbf{f}$. We denote a shift of an edge flow $\mathbf{f}$ as $\mathbf{f}^{(1)} \triangleq \mathbf{L}_1\mathbf{f} = \mathbf{B}_1^\top \mathbf{B}_1 \mathbf{f} + \mathbf{B}_2 \mathbf{B}_2^\top \mathbf{f}$, where we define $\mathbf{f}_{\mathrm{l}}^{(1)} \triangleq \mathbf{B}_1^\top \mathbf{B}_1 \mathbf{f}$ and $\mathbf{f}_{\mathrm{u}}^{(1)} \triangleq \mathbf{B}_2 \mathbf{B}_2^\top \mathbf{f}$. The shifting result on the i-th edge, i.e., $[\mathbf{f}^{(1)}]_i$, can be expressed as

$$\begin{aligned}[\mathbf{f}^{(1)}]_i &= [\mathbf{f}_{\mathrm{l}}^{(1)}]_i + [\mathbf{f}_{\mathrm{u}}^{(1)}]_i \\ &= \sum_{j \in \{\mathcal{N}_{\mathrm{l},i} \cup\, i\}} [\mathbf{L}_{1,\mathrm{l}}]_{ij} [\mathbf{f}]_j + \sum_{j \in \{\mathcal{N}_{\mathrm{u},i} \cup\, i\}} [\mathbf{L}_{1,\mathrm{u}}]_{ij} [\mathbf{f}]_j. \end{aligned} \tag{2}$$

Thus, the shift operation in the edge space is local. For each edge, it collects information from its lower and upper adjacent neighbors and combines this information with its own state. Figs. 1b and 1c show the operation of shifting an all-one flow by $\mathbf{L}_{\mathrm{l},1}$ and $\mathbf{L}_{1,\mathrm{u}}$ on one edge. For a given input $\mathbf{f}$, we can now write the filter output as

$$\mathbf{f}_o = \mathbf{H}_1\mathbf{f} = \sum_{l=0}^{L-1} h_l \mathbf{L}_1^l \mathbf{f} = \sum_{l=0}^{L-1} h_l \mathbf{f}^{(l)}, \tag{3}$$

where $\mathbf{f}^{(l)} \triangleq \mathbf{L}_1^l \mathbf{f} = \mathbf{L}_1 \mathbf{f}^{(l-1)}$ is the l-th shift of the edge flow, which contains information up to its l-th hop lower and upper neighbors. Thus, the filter output can be seen as a weighted linear combination of consecutively shifted edge flows, which is similar to the concepts of graph signal shifting in graph filters [3], [4].

Distributed implementation and complexity. Observe that (3) is characterized by the recursion $\mathbf{f}^{(l)} = \mathbf{L}_1 \mathbf{f}^{(l-1)}$. That is, edges can locally compute $\mathbf{f}^{(l)}$ by exchanging information with their neighbors about the previous shifted signal, $\mathbf{f}^{(l-1)}$. This implies that in (3) we need in total L such shifts by $\mathbf{L}_1$. Moreover, as each shift (2) can be implemented via local operations, which can be implemented by distributed communication between edges, the final output of the simplicial FIR filter can be computed in a distributed way. If we denote the maximal edge degree by D, then the communication cost

Authorized licensed use limited to: TU Delft Library. Downloaded on December 09,2021 at 13:34:32 UTC from IEEE Xplore. Restrictions apply.

to compute (2) is $\mathcal{O}(D)$. Thus, (3) has a total cost of $\mathcal{O}(LN_1D)$. Usually, the edge degree is independent of (and much smaller than) the number of edges, so we would have a complexity of $\mathcal{O}(LN_1)$.

Spectral analysis. By applying an eigen-decomposition on the filter (1), we obtain $\mathbf{H}_1 = \sum_{l=0}^{L-1} h_l \mathbf{U}_1 \mathbf{\Lambda}_1^l \mathbf{U}_1^\top = \mathbf{U}_1 \Big(\sum_{l=0}^{L-1} h_l \mathbf{\Lambda}_1^l \Big) \mathbf{U}_1^\top$, where we define $\tilde{\mathbf{H}}_1 \triangleq \sum_{l=0}^{L-1} h_l \mathbf{\Lambda}_1^l$ as the frequency response of $\mathbf{H}_1$. For simplicity of notation, let us assume hereafter that $\mathbf{\Lambda}_1 = \text{diag}(\lambda_1, \ldots, \lambda_{N_1})$. At frequency λ_i, the filter implements the response $\tilde{H}_1(\lambda_i) = \sum_{l=0}^{L-1} h_l \lambda_i^l$ leading to the spectral input-output relation $\tilde{f}_o(\lambda_i) = \tilde{H}_1(\lambda_i) \tilde{f}(\lambda_i)$. As different types of frequencies are present in simplicial signals, filter (1) needs to control the different signal components according to their corresponding frequencies. In other words, the gradient component needs to be tuned using $\tilde{H}_1(\lambda_i)$ with $\lambda_i \in \mathcal{Q}_G$, the curl component using $\tilde{H}_1(\lambda_i)$ with $\lambda_i \in \mathcal{Q}_C$, and the harmonic component using $\tilde{H}_1(\lambda_i)$ with $\lambda_i \in \mathcal{Q}_H$. By properly designing the frequency response at each frequency, we can thus implement any desired filtering operation.

Filter design. Given a filter specification in the frequency domain $\tilde{H}_1(\lambda_i) = \sum_{l=0}^{L-1} h_l \lambda_i^l = g_i$, for $i = 1, \ldots, N_1$, we can design the filter coefficients by solving the least squares problem

$$\min_{\mathbf{h}} \|\mathbf{\Phi}\mathbf{h} - \mathbf{g}\|^2, \tag{4}$$

where $\mathbf{\Phi} \in \mathbb{R}^{N_1 \times L}$ is a Vandermonde matrix with entries $[\mathbf{\Phi}]_{i,j} = \lambda_i^{j-1}$, and $\mathbf{g} = [g_1, \ldots, g_{N_1}]^\top$. Hence, given a desired frequency response $\mathbf{g}$, we first identify the frequency sets $\mathcal{Q}_G$, $\mathcal{Q}_C$ and $\mathcal{Q}_H$ by computing the eigenvalues of $\mathbf{L}_{1,\text{l}}$ and $\mathbf{L}_{1,\text{u}}$, and then solve problem (4) to obtain $\mathbf{h}$. We can then build a simplicial FIR filter which has (approximately) the desired frequency response and apply the filter directly in the simplicial space as in (3). Thus, to solve the problem in Example 1, we can design a *gradient-preserving* filter by setting the desired frequency response to one at the gradient frequencies and zero at the rest, as shown in Figs. 2e and 2f.

Subspace-varying simplicial filter. As seen in Section III, the gradient space is fully described by $\mathbf{L}_{1,\text{l}}$ and the curl space by $\mathbf{L}_{1,\text{u}}$. However, for the filter (1), the shift operators $\mathbf{L}_{1,\text{l}}$ and $\mathbf{L}_{1,\text{u}}$ are jointly weighted by $\{h_l\}_{l=0}^{L-1}$, and consequently we cannot control the frequency response at the gradient and curl frequencies independently. To enable an independent design of the frequency responses at gradient and curl frequencies we thus assign $\mathbf{L}_{1,\text{l}}$ and $\mathbf{L}_{1,\text{u}}$ a different set of weights, leading to the following *subspace-varying FIR filter*

$$\mathbf{H}_1^{\text{SV}} = h_0 \mathbf{I} + \sum_{l_1=1}^{L_1} \alpha_{l_1} (\mathbf{B}_1^\top \mathbf{B}_1)^{l_1} + \sum_{l_2=1}^{L_2} \beta_{l_2} (\mathbf{B}_2 \mathbf{B}_2^\top)^{l_2}, \tag{5}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{L_1}]^\top$ and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_{L_2}]^\top$ control the gradient and curl frequency responses, and h_0 controls the harmonic frequency response. We use the convention that if $L_1 = 0$ or $L_2 = 0$ the corresponding terms are zero. This can be useful if we only want to tune the gradient or curl component. Our earlier discussion on the shift operator and the complexity also applies to (5).

Using a spectral decomposition of (5), we can see that its frequency response is given by

$$\tilde{H}_1^{\text{SV}}(\lambda_i) = \begin{cases} h_0, \text{ for } \lambda_i \in \mathcal{Q}_H, \\ h_0 + \sum_{l_1=1}^{L_1} \alpha_{l_1} \lambda_i^{l_1}, \text{ for } \lambda_i \in \mathcal{Q}_G, \\ h_0 + \sum_{l_2=1}^{L_2} \beta_{l_2} \lambda_i^{l_2}, \text{ for } \lambda_i \in \mathcal{Q}_C. \end{cases} \tag{6}$$

From these equations, we see that in the subspace-varying filter, the frequency responses at gradient and curl frequencies can indeed be controlled independently via the different weights. As a result, the filter is more flexible than $\mathbf{H}_1$ (and has increased degrees of freedom).

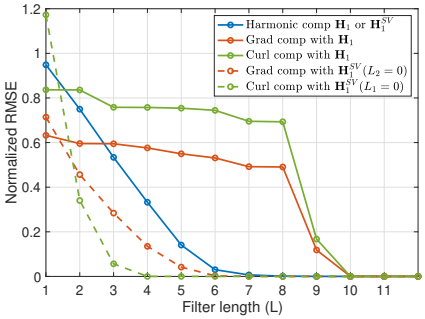


Fig. 3. Sub-component extraction by filters $\mathbf{H}_1$ and $\mathbf{H}_1^{\text{SV}}$. The extraction becomes better as the filter length grows. For the harmonic component, both filters perform the same as expected. But for the gradient and curl component extraction, the filter $\mathbf{H}_1$ cannot obtain an accurate extraction until the filter length is nine. However, $\mathbf{H}_1^{\text{SV}}$ performs much better than $\mathbf{H}_1$, which is consistent to the discussion after (7).

For a specified frequency response $\tilde{\mathbf{H}}_1^{\text{SV}} = \text{diag}(\mathbf{g})$, we can again design the filter coefficients by solving the least squares problem

$$\min_{h_0, \boldsymbol{\alpha}, \boldsymbol{\beta}} \left\| \left[\mathbf{1} \;\middle|\; \begin{array}{cc} \mathbf{0}^\top & \\ \hline \mathbf{\Phi}_G & \mathbf{0}_{N_G \times L_2} \\ \mathbf{0}_{N_C \times L_1} & \mathbf{\Phi}_C \end{array} \right] \begin{bmatrix} h_0 \\ \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} - \mathbf{g} \right\|, \tag{7}$$

where $\mathbf{1}$ ($\mathbf{0}$) is an all-one (all-zero) vector, $\mathbf{0}_{m \times n}$ is an all-zero matrix of size $m \times n$, and $\mathbf{\Phi}_G \in \mathbb{R}^{N_G \times L_1}$ and $\mathbf{\Phi}_C \in \mathbb{R}^{N_C \times L_2}$ are Vandermonde matrices with respective entries $[\mathbf{\Phi}_G]_{i,j} = \lambda_{G,i}^j$ and $[\mathbf{\Phi}_C]_{i,j} = \lambda_{C,i}^j$. An immediate observation is that when L_1 (or L_2) is zero, the number of rows for this system of equations is one plus the number of curl (or gradient) frequencies, which is less than the corresponding number of rows of $\mathbf{\Phi}$ in (4). This makes the filter (5) more suitable when only gradient (or curl) components need to be tuned, as the remaining components can be jointly controlled by the frequency response h_0 at zero as in (6).

V. Numerical Experiments

In this section, we provide experimental results for the simplicial filters (1) and (5) to process the edge flow in a network. To evaluate the performance, we consider the normalized root mean square error (NRMSE), defined as $e = \|\hat{\mathbf{f}} - \mathbf{f}_0\|_2 / \|\mathbf{f}_0\|_2$, where $\hat{\mathbf{f}}$ is the flow estimate obtained by the filters and $\mathbf{f}_0$ is the groundtruth flow.

Sub-component extraction. We first constructed a synthetic SC as in Fig. 1a with seven nodes, ten edges, and three triangles [11]. We then generated an edge flow with a flat spectrum $\mathbf{f} = \mathbf{U}_1 \tilde{\mathbf{f}}$ with $\tilde{\mathbf{f}} = \mathbf{1} \in \mathbb{R}^{10}$. We then designed FIR filters to preserve only the gradient, curl or harmonic signal components respectively, and compared their results to an exact projection of the signals on the corresponding subspace (see [11], [14], [15], [17]) in terms of the error.

Fig. 3 displays the performance of filter (1) (solid line) and the subspace-varying filter (5) (dashed line). As expected, the subspace extraction becomes more accurate as L grows (indeed for a filter of order $L = 10$ the least squares problem can always be solved exactly). Importantly, to extract a particular subspace component, filter (1) performs always worse than the subspace-varying filter (5). The reason is that for filter (5), we can tune the frequency response in the respective subspace independently, and thus require less coefficients to approximate the desired frequency response.

Edge flow denoising. Using the same network again, we generated an edge flow induced by a node signal which contains all ones in the graph frequency domain and corrupted by white noise, i.e., similar

Authorized licensed use limited to: TU Delft Library. Downloaded on December 09,2021 at 13:34:32 UTC from IEEE Xplore. Restrictions apply.

TABLE I
Flow prediction on Sioux Falls network. L_{total} is the total filter length and e_1 and e_2 the prediction errors by $\mathbf{H}_1$ and $\mathbf{H}_1^{\text{SV}}$ on the test data.

L_{total}	e_1	e_2	L_{total}	e_1	e_2
1	0.786 ± 0.003	–	6	0.070 ± 0.010	$\mathbf{0.013 \pm 2 \cdot 10^{-5}}$
2	0.659 ± 0.005	$\mathbf{0.613 \pm 0.007}$	7	0.048 ± 0.014	$\mathbf{0.003 \pm 2 \cdot 10^{-4}}$
3	$\mathbf{0.086 \pm 0.005}$	0.220 ± 0.005	8	0.009 ± 0.010	$\mathbf{5 \cdot 10^{-4} \pm 6 \cdot 10^{-5}}$
4	$\mathbf{0.071 \pm 0.006}$	0.082 ± 0.003	9	0.007 ± 0.012	$\mathbf{8 \cdot 10^{-5} \pm 3 \cdot 10^{-5}}$
5	0.037 ± 0.010	$\mathbf{0.034 \pm 5 \cdot 10^{-4}}$	10	0.002 ± 0.012	$\mathbf{2 \cdot 10^{-5} \pm 10^{-5}}$

to an instance of the problem in Example 1. The noisy flow (Fig. 2b) introduces an error of 0.46 originally. When using the (low-pass) denoising filters discussed in [17] and [11] (Figs. 2c and 2d), the error becomes even larger. This is expected as the assumption of a low-pass signal used in [17], [11] is not fulfilled in our problem setup. Using a proper filter design that preserves the gradient components, filter (1) with $L = 4$ and filter (5) with $L_1 = L_2 = 1$ can, however, achieve errors of 0.39 and 0.23, as shown in Figs. 2e and 2f.

Sioux Falls network. We now consider a real world Sioux Falls transportation network [21]. It contains 24 nodes, 38 edges, and 2 triangles (1-simplices). We assumed an autoregressive (AR) model to generate a set of time-evolving edge flows as

$$\mathbf{f}_{t+1} = (0.5\mathbf{I} + 0.3\mathbf{B}_1^\top \mathbf{B}_1 + \mathbf{B}_2\mathbf{B}_2^\top + 0.5(\mathbf{B}_2\mathbf{B}_2^\top)^2)^{-1}\mathbf{f}_t, \quad (8)$$

where t is the time instance. Assuming the model to be unknown to the analyst, we sought to train filters (1) and (5) in order to learn a data-driven model for flow prediction. Accordingly, we generated a training set containing 10 sample pairs $\{\mathbf{f}_{s,\text{in}}, \mathbf{f}_{s,\text{out}}\}$, where $\mathbf{f}_{s,\text{in}}$ is a random Gaussian flow to be used as input for model (8), and $\mathbf{f}_{s,\text{out}}$ is the output. Note that we can rewrite the filtering equation (3) as

$$[\mathbf{f}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(L-1)}]\mathbf{h} = \mathbf{f}_o, \quad (9)$$

where the system matrix contains the shifted versions of the input. With the training set, we built the system matrices by shifting each input and horizontally concatenated them and the outputs. Then, we used the least squares solution to (9) as the trained filter coefficients $\mathbf{h}$ to build a data-driven filter $\mathbf{H}_1$. The test data, $\mathcal{T} = \{\mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_{99}\}$, is generated by initializing (8) with a random flow $\mathbf{f}_0$. Then, with each observation $\mathbf{f}_t$, $t = 0, 1, \ldots, 99$, we obtained a one-step prediction $\hat{\mathbf{f}}_{t+1} = \mathbf{H}_1\mathbf{f}_t$. Finally, we evaluated the average performance by computing the error between $\mathbf{f}_{t+1}$ and $\hat{\mathbf{f}}_{t+1}$. Similar steps were followed for the subspace-varying filter $\mathbf{H}_1^{\text{SV}}$.

The results are shown in Table. I. We reported the best performance for $\mathbf{H}_1^{\text{SV}}$ for each L_{total}, as different L_1 and L_2 lengths lead to different performance. When L_{total} is larger, both filters fit the model (8) better and the prediction becomes more accurate as well. Moreover, to deal with models like (8) in the simplicial space, the more flexible subspace-varying filter $\mathbf{H}_1^{\text{SV}}$ usually learns the model better.

VI. Conclusion

We proposed two linear FIR filters in the form of (1) and (5) for filtering simplicial signals (and more specifically edge flows). We revisited the simplicial Fourier transform, and defined gradient and curl frequencies for edge flows, which measure the divergence and rotational properties of the flow. Both of our proposed filters are able to tune signals for these two kinds of frequencies and can be distributively implemented, since the simplicial shift operator defined by the Hodge Laplacian can be built from local operations. We showed that the subspace-varying filter is more flexible and is especially beneficial if we desire to tune only gradient or curl frequencies, since the different weights on the lower and upper Laplacians in (5) enable an independent tuning of those components. Finally, experimental results were reported to support our findings. Future work will concern: 1) examining the utility of simplicial filters beyond the edge space; 2) performing filter design for both (1) and (5) with smaller costs; 3) combining simplicial filters into simplicial neural networks as considered, e.g., in [20].

References

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

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

[3] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[4] ——, "Discrete signal processing on graphs: Frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

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

[6] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 128–138, 2020.

[7] W. Huang and A. Ribeiro, "Metrics in the space of high order networks," *IEEE Transactions on Signal Processing*, vol. 64, no. 3, pp. 615–629, 2015.

[8] A. C. Wilkerson, T. J. Moore, A. Swami, and H. Krim, "Simplifying the homology of networks via strong collapses," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 5258–5262.

[9] S. Barbarossa and M. Tsitsvero, "An introduction to hypergraph signal processing," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 6425–6429.

[10] S. Zhang, Z. Ding, and S. Cui, "Introducing hypergraph signal processing: Theoretical foundation and practical applications," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 639–660, 2019.

[11] 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," *arXiv preprint arXiv:2101.05510*, 2021.

[12] M. Coutino, G. V. Karanikolas, G. Leus, and G. B. Giannakis, "Self-driven graph volterra models for higher-order link prediction," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3887–3891.

[13] G. Leus, M. Yang, M. Coutino, and E. Isufi, "Topological volterra filters," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, (accepted) 2021.

[14] S. Barbarossa and S. Sardellitti, "Topological signal processing over simplicial complexes," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2992–3007, 2020.

[15] L.-H. Lim, "Hodge laplacians on graphs," *SIAM Review*, vol. 62, no. 3, pp. 685–715, 2020.

[16] X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye, "Statistical ranking and combinatorial hodge theory," *Mathematical Programming*, vol. 127, no. 1, pp. 203–244, 2011.

[17] M. T. Schaub and S. Segarra, "Flow smoothing and denoising: Graph signal processing in the edge-space," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2018, pp. 735–739.

[18] J. Jia, M. T. Schaub, S. Segarra, and A. R. Benson, "Graph-based semi-supervised & active learning for edge flows," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 761–771.

[19] M. T. Schaub, A. R. Benson, P. Horn, G. Lippner, and A. Jadbabaie, "Random walks on simplicial complexes and the normalized hodge 1-laplacian," *SIAM Review*, vol. 62, no. 2, pp. 353–391, 2020.

[20] T. M. Roddenberry and S. Segarra, "Hodgenet: Graph neural networks for edge data," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2019, pp. 220–224.

[21] L. J. Leblanc, "An algorithm for the discrete network design problem," *Transportation Science*, vol. 9, no. 3, pp. 183–199, 1975.

Authorized licensed use limited to: TU Delft Library. Downloaded on December 09,2021 at 13:34:32 UTC from IEEE Xplore. Restrictions apply.