TU Delft

Delft University of Technology

Topological Volterra Filters

Leus, Geert; Yang, Maosheng; Coutino, Mario; Isufi, Elvin

DOI
10.1109/ICASSP39728.2021.9414275

Publication date
2021

Document Version
Final published version

Published in
ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

Citation (APA)
Leus, G., Yang, M., Coutino, M., & Isufi, E. (2021). Topological Volterra Filters. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5385-5389). Article 9414275 IEEE. https://doi.org/10.1109/ICASSP39728.2021.9414275

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.

ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9414275

TOPOLOGICAL VOLTERRA FILTERS

Geert Leus, Maosheng Yang, Mario Coutino, and Elvin Isufi

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, Delft, The Netherlands

ABSTRACT

To deal with high-dimensional data, graph filters have shown their power in both graph signal processing and data science. However, graph filters process signals exploiting only pairwise interactions between the nodes, and they are not able to exploit more complicated topological structures. Graph Volterra models, on the other hand, are also able to exploit relations between triplets, quadruplets and so on. However, they have only been exploited for topology identification and are only based on one-hop relations. In this paper, we first review graph filters and graph Volterra models and then merge the two concepts resulting in so-called topological Volterra filters (TVFs). TVFs process signals over multiple hops of higher-level topological structures. First-level TVFs are basically similar to traditional graph filters, yet higher-level TVFs provide a more general processing framework. We apply TVFs to inverse filtering and recommender systems.

***Index Terms*—** Graph Volterra model, Graph filters, Higher-level interactions, Graph signal processing

1. INTRODUCTION

Filters are central in signal processing and machine learning to manipulate signals and extract patterns. When these signals reside in irregular and complex spaces, such as networks, e.g., social, biological, and financial networks, conventional filtering techniques cannot be applied anymore [1]. In these instances, filtering tools capable of leveraging the signal-topology coupling are necessary. Such tools have recently been developed in the field of graph signal processing (GSP) and they are known under the term *graph filters* [1, 2].

Graph filters process the signal by manipulating its variability w.r.t. pairwise relationships encoded in the graph edges. This has been widely studied from both the graph vertex and graph spectral domain in recent years [2, 3, 4]. There are different ways of implementing a graph filter as an interaction between the graph nodes, with the most popular ones including the finite impulse response (FIR) graph filters [3, 4], the autoregressive moving average (ARMA) graph filters [5], as well as their node-varying [6] and edge-varying [7, 8] counterparts. All these implementations are based on nodes aggregating local information from their neighbors, potentially improving their expressive power by *differently* weighting the aggregation per node or the information coming from different neighbors. Nonlinear graph filters have also been proposed, such as median graph filters [9] or graph filters living in a reproducing kernel Hilbert space [10].

M. Yang is supported by the AIDU program at the Delft University of Technology in the Aidrolab call. Mario Coutino is partially supported by CONACYT. E-mails: {g.j.t.leus; m.yang-2; m.a.coutinominguez; e.isufi-1}@tudelft.nl

The common aspect of current graph filter implementations is that they process the signal by leveraging only pairwise relationships between the nodes. However, pairwise representations are often insufficient to capture the irregular structure that is often present in data. Therefore, in [11, 12, 13, 14, 15, 16], signals defined over a topological space, i.e., a set of points along with a set of neighborhood relations, are analyzed. For instance, in co-authorship networks, many papers are written by more than two authors and hence just considering pairs of authors does not provide enough information [17]; or in social networks, pairwise friend relationships are not able to identify the group relationship structure [18]. In these situations, higher-level topological relations, such as proximity between nodes and groups of nodes (tuples), are needed to extract relevant information from the signal-topology coupling. From this perspective, conventional FIR graph filters can be seen as topological filters of level one; i.e., working only with pairwise topological information.

Altogether, the above works build up towards the extension of graph filters to topological spaces. To successfully process these signals there is the need to extract two types of information from the data: i) the incidence matrices at the different topological levels; and ii) the topological signals on each level, both of which are open research directions. In many applications, the topological information is directly visible in the data, e.g., in co-authorship or social networks, the groups that interact and how strong they interact can be determined. Alternatively, it can be estimated from topological signals that might be available at different levels, by exploiting the signal-topology coupling. This is similar to the works that build a graph (i.e., pairwise relationships) from signals that are available at the node level [19, 20]. An example of such a higher-level topology inference approach relies on the so-called graph Volterra model [21, 22]. This approach is also based on node-level signals and attempts to explain the signal value on one node by a superposition of linear, quadratic, cubic, etc. combinations of the signal values on the other nodes. The trained Volterra kernels then reveal the topological information at the different levels. Approaches that also exploit higher-level topological signals remain limited since such information is often not available.

The kernels of a graph Volterra model describe how a specific node is connected to individual nodes, pairs of nodes, triplets of nodes, etc. Viewing these relations as one-hop connections, we aim to answer *how to extend such one-hop connections to a processing platform that can filter a node-level signal over multiple hops of a topological structure*. The answer will be provided by the so-called topological Volterra filter (TVF), which extends traditional graph filters i) by processing node-level signals not only over multiple hops of pairwise connections, but also over multiple hops of higher-level interactions, and ii) by introducing nonlinear combinations of the node-level signal values to define higher-level signal values. We show the potential of TVFs in solving a simple inverse problem on a graph and for rating prediction in recommender systems.

Authorized licensed use limited to: TU Delft Library. Downloaded on August 24,2021 at 12:39:04 UTC from IEEE Xplore. Restrictions apply.

2. PRELIMINARIES

2.1. Graph Signal Processing

GSP relies on the assumption that data lives in an irregular domain described by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, \ldots, N\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are the vertex (node) and edge set, respectively. We can then associate to each node $i \in \mathcal{V}$ the datum x_i and collect them all in the vector $\boldsymbol{x} \in \mathbb{R}^N$, which is called a *graph signal*. The graph wherein the data is defined is typically represented by the so-called graph shift operator (GSO) [3], which is an $N \times N$ matrix $\boldsymbol{S}$ that captures the graph structure; that is, the entries $[\boldsymbol{S}]_{i,j}$ for $i \neq j$ are nonzero only if node j is connected to node i by an edge. Typical choices for the GSO are the (weighted) adjacency matrix $\boldsymbol{W}$ and the graph Laplacian $\boldsymbol{L}$ as they provide a natural way to extend the notion of *frequencies* to graphs [3, 4].

Similar to traditional signal processing, the workhorse of GSP is *graph filtering* [2, 3, 4]. The most common graph filters are linear operators that process the graph signal. They can be represented by means of a polynomial of the GSO of order K, i.e.,

$$\boldsymbol{y} = \sum_{k=0}^{K} h_k \boldsymbol{S}^k \boldsymbol{x}, \tag{1}$$

with scalar coefficients $h_0, \ldots, h_K$. Graph filters have found success in several applications such as anomaly detection in sensor networks, data classification [4], and high-dimensional data denoising [2], to name a few. However, structures of the form (1) only exploit the pairwise relationships between the nodes encoded by the edges of the graph, but not any higher-level connections such as triplets or quadruples of nodes that interact.

2.2. Graph Volterra Models

A model that addresses higher-level topological interactions is the *graph Volterra model*, which was introduced in [21, 22]. Every value of the graph signal (also called a *node-level signal* in this context) for a particular node is then expressed as a function of the signal values on all the nodes that are involved in higher-level interactions with that node. More specifically, similar to a Volterra series, the signal value of node i is expressed as a linear combination of terms at different topological levels. At level p, if node i is involved in a $(p+1)$-tuple together with nodes $(j_1, j_2, \ldots, j_p)$, one of the terms will be the product of the signal values at these nodes, i.e., $x_{j_1} x_{j_2} \ldots x_{j_p}$, thereby implicitly introducing some higher-level topological signal. Considering then a scalar offset $s_{i,0} \in \mathbb{R}$ relative to node i and a vector $\boldsymbol{s}_{i,p} \in \mathbb{R}^{N^p \times 1}$ describing the influence of all p-tuples of nodes on node i, for $p = 1, 2, \ldots, P$, we can express the signal value x_i of node i as

$$x_i = s_{i,0} + \sum_{p=1}^{P} \boldsymbol{s}_{i,p}^{\top} \boldsymbol{x}^{\otimes p}, \tag{2}$$

where we define the Kronecker product powers $\boldsymbol{x}^{\otimes(p)} = \boldsymbol{x}^{\otimes(p-1)} \otimes \boldsymbol{x}$ with $\boldsymbol{x}^{\otimes 1} = \boldsymbol{x}$ and $\otimes$ the Kronecker product. We call (2) a Pth-level graph Volterra model as it accounts for the influence of at most P-tuples. By stacking the x_is of all nodes, we obtain

$$\boldsymbol{x} = \boldsymbol{s}_0 + \sum_{p=1}^{P} \boldsymbol{S}_p \boldsymbol{x}^{\otimes p}, \tag{3}$$

where $[\boldsymbol{s}_0]_i = s_{i,0}$ and $[\boldsymbol{S}_p]_{i,:} = \boldsymbol{s}_{i,p}^{\top}$. Here $\boldsymbol{s}_0$ is the global offset signal and $\boldsymbol{S}_p$ describes how every p-tuple of nodes influences every individual node because together they form a $(p+1)$-tuple. As such, $\boldsymbol{S}_1 \in \mathbb{R}^{N \times N}$ connects nodes to nodes and thus could play the role of the traditional GSO. We label this here as a *first-level GSO*, since similarly $\boldsymbol{S}_p \in \mathbb{R}^{N \times N^p}$ connects p-tuples with nodes and can be viewed as a pth-level GSO. An example illustrating the structure of $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ is shown in Figures 1 and 2, respectively. Following the hypergraph literature, we can basically use $\boldsymbol{S}_p$ to encode hyperedges of size $p+1$. Also note that not all p-tuples of an active $(p+1)$-tuple should be active as in simplicial complexes. Such topological constraints can be included, but are not required here.

Graph Volterra models are self-driven, i.e., in contrast to (1) both the input and output are based on the signal $\boldsymbol{x}$, and have been successfully used to identify higher-level interactions from data [21, 22]. However, the identified connections between p-tuples and nodes only describe one-hop connections.

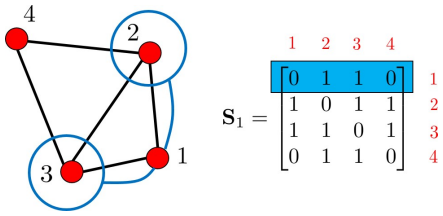

Fig. 1. Illustration of first-level GSO $\boldsymbol{S}_1$ for a set of 5 active pairwise interactions (in black), i.e., $(1,2), (1,3), (2,3), (2,4)$, and $(3,4)$. Thus, node 1 is connected to the nodes 2 and 3 (in blue).

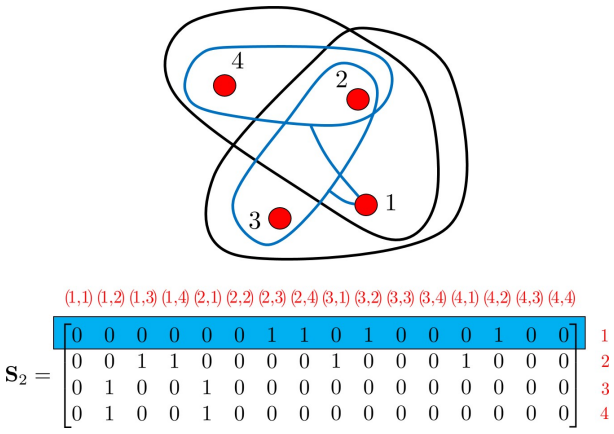

Fig. 2. Illustration of second-level GSO $\boldsymbol{S}_2$ for a set of 2 active triplets (in black), i.e., $(1,2,3)$ and $(1,2,4)$. Thus, node 1 is connected to the pairs $(2,3), (3,2), (2,4)$, and $(4,2)$ (in blue).

3. TOPOLOGICAL VOLTERRA FILTERS

In this section, we merge graph filters and graph Volterra models, leading to a processing framework for *node-level* signals that we refer to as a *topological Volterra filter* (TVF). On one hand, a TVF will extend graph filters from a multi-hop processing tool that exploits only pairwise connections to one that also considers higher-level interactions. On the other hand, a TVF will extend graph Volterra models from a higher-level topology identification tool inferring only one-hop connections to a higher-level processing tool exploiting also multi-hop connections. We will present the TVF in different steps, by looking at every term of the Volterra model separately.

Authorized licensed use limited to: TU Delft Library. Downloaded on August 24,2021 at 12:39:04 UTC from IEEE Xplore. Restrictions apply.

3.1. First-Level Topological Volterra Filter

Let us rewrite the first-level term of the graph Volterra model as an input-output model:

$$\boldsymbol{y} = \boldsymbol{S}_1\boldsymbol{x}, \tag{4}$$

where the first-level GSO $\boldsymbol{S}_1 \in \mathbb{R}^{N\times N}$ describes the pairwise relationships between all nodes. This model can be viewed as a one-hop diffusion exploiting the node-node interactions defined by $\boldsymbol{S}_1$. Extending this to multiple hops (higher-order filter) we can basically use the fact that $\boldsymbol{S}_1^k$ describes the node-node interactions through k hops in $\boldsymbol{S}_1$. As a result, we can extend (4) to a higher-order filter as

$$\boldsymbol{y} = \sum_{k=0}^{K} h_k \boldsymbol{S}_1^k \boldsymbol{x}, \tag{5}$$

which is labeled as a *first-level TVF* and which compared to (1) boils down to a traditional graph filter in $\boldsymbol{S}_1$.

3.2. Second-Level Topological Volterra Filter

Let us likewise rewrite the second-level term of the graph Volterra model as an input-output model:

$$\boldsymbol{y} = \boldsymbol{S}_2\boldsymbol{x}^{\otimes 2}, \tag{6}$$

where the second-level GSO $\boldsymbol{S}_2 \in \mathbb{R}^{N\times N^2}$ describes the relationships of all node pairs with all individual nodes. This model can be viewed as a one-hop diffusion exploiting the pair-node interactions defined by $\boldsymbol{S}_2$. Extending this to multiple hops (higher-order filter) requires a description of all pair-node interactions over multiple hops given the one-hop interactions defined by $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$.

To realize that, observe that if $\boldsymbol{S}_1$ describes the one-hop node-node interactions in $\boldsymbol{S}_1$, then $\boldsymbol{S}_1 \otimes \boldsymbol{S}_1$ describes the one-hop pair-pair interactions in $\boldsymbol{S}_1$. Similarly, if $\boldsymbol{S}_1^l$ describes the node-node interactions through l hops in $\boldsymbol{S}_1$, then $\boldsymbol{S}_1^{l_1} \otimes \boldsymbol{S}_1^{l_2}$ describes the pair-pair interactions through l_1 hops in $\boldsymbol{S}_1$ for one node and l_2 hops in $\boldsymbol{S}_1$ for the other node. Considering the above, multi-hop pair-node interactions given $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ can be represented by $\boldsymbol{S}_1^k\boldsymbol{S}_2(\boldsymbol{S}_1^{l_1} \otimes \boldsymbol{S}_1^{l_2})$ as illustrated in Figure 3. This operation concatenates three types of interactions: i) the pair-pair interactions through l_1 hops in $\boldsymbol{S}_1$ for one node and l_2 hops in $\boldsymbol{S}_1$ for the other node, ii) the pair-node interactions defined by $\boldsymbol{S}_2$, and finally iii) the node-node interactions through k hops in again $\boldsymbol{S}_1$. This allows us to extend (6) to a higher-order filter as

$$\boldsymbol{y} = \sum_{k=0}^{K} \sum_{l_1,l_2=0}^{L_1,L_2} h_{k,l_1,l_2} \boldsymbol{S}_1^k \boldsymbol{S}_2(\boldsymbol{S}_1^{l_1} \otimes \boldsymbol{S}_1^{l_2})\boldsymbol{x}^{\otimes 2} \tag{7}$$

which combined with (5) is labeled as a *second-level TVF*.

3.3. Higher-Level Graph Filters

Applying the same reasoning for higher levels, we basically need a description of all (P-tuple)-node interactions over multiple hops given the one-hop interactions defined by $\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_P$. Similar to the second-level TVF, such a description is provided by $\boldsymbol{S}_1^k\boldsymbol{S}_P(\boldsymbol{S}_1^{l_1} \otimes \cdots \otimes \boldsymbol{S}_1^{l_P})$ which is similar to Figure 3 yet with the pair-node interaction replaced by a (P-tuple)-node interaction. This would lead to a filter of the form

$$\boldsymbol{y} = \sum_{k=0}^{K} \sum_{l_1,\ldots,l_P=0}^{L_1,\ldots,L_P} h_{k,l_1,\ldots,l_P} \boldsymbol{S}_1^k \boldsymbol{S}_P(\boldsymbol{S}_1^{l_1} \otimes \cdots \otimes \boldsymbol{S}_1^{l_P})\boldsymbol{x}^{\otimes P}, \tag{8}$$

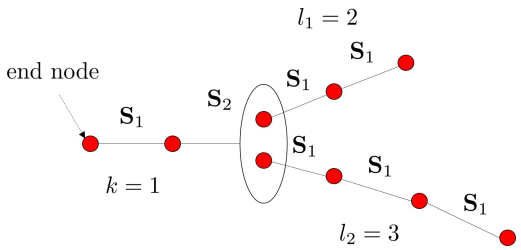

Fig. 3. Illustration of the general multi-hop pair-node interaction given the first- and second-level GSOs $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$. This interaction can be described by $\boldsymbol{S}_1^k\boldsymbol{S}_2(\boldsymbol{S}_1^{l_1} \otimes \boldsymbol{S}_1^{l_2})$.

which combined with the related lower-level terms is labeled as a *Pth-level TVF*.

However, in contrast to the second-level TVF, there are more paths to connect a P-tuple to a single node over multiple hops when $P > 2$. For instance, triplet-node interactions over multiple hops can be described by $\boldsymbol{S}_1^k\boldsymbol{S}_3(\boldsymbol{S}_1^{l_1} \otimes \boldsymbol{S}_1^{l_2} \otimes \boldsymbol{S}_1^{l_3})$ but alternative paths are obtained by $\boldsymbol{S}_1^k\boldsymbol{S}_2(\boldsymbol{S}_1^{m_1}\otimes\boldsymbol{S}_1^{m_2})(\boldsymbol{I}\otimes\boldsymbol{S}_2)(\boldsymbol{S}_1^{l_1}\otimes\boldsymbol{S}_1^{l_2}\otimes\boldsymbol{S}_1^{l_3})$. This term actually describes a concatenation of five types of interactions: i) the triplet-triplet interactions through l_1 hops in $\boldsymbol{S}_1$ for the first node, l_2 hops in $\boldsymbol{S}_1$ for the second node, and l_3 hops in $\boldsymbol{S}_1$ for the third node, ii) the pair-node interactions defined by $\boldsymbol{S}_2$ for one pair of nodes not processing the other node, iii) the pair-pair interactions through m_1 hops in $\boldsymbol{S}_1$ for one node and m_2 hops in $\boldsymbol{S}_1$ for the other node, iv) the pair-node interactions defined by $\boldsymbol{S}_2$ and finally v) the node-node interactions through k hops in $\boldsymbol{S}_1$. Higher-level TVFs can be defined by taking also such interactions into account. However, since this becomes rather complicated, we will restrict ourselves to higher-level TVFs of the form (8). Note hereby that the filtering operation in (8) [and thus also the one in (7)] can be rewritten using $(\boldsymbol{S}_1^{l_1} \otimes \cdots \otimes \boldsymbol{S}_1^{l_P})\boldsymbol{x}^{\otimes P} = \boldsymbol{S}_1^{l_1}\boldsymbol{x} \otimes \cdots \otimes \boldsymbol{S}_1^{l_P}\boldsymbol{x}$.

4. NUMERICAL EXPERIMENTS

We corroborate the performance of a TVF for solving an inverse problem on a synthetic graph and for rating prediction with the MovieLens100K dataset [23].

4.1. Synthetic Data

We consider a 2nd-level TVF [cf. (5) and (7)] for graph inverse filtering and compare it with a 1st-level TVF [cf. (5)], e.g., a graph filter. We generate a $50-$node sensor graph using the GSPBOX [24] whose adjacency matrix is used also as 1st-level GSO $\boldsymbol{S}_1$. To build the 2nd-level GSO $\boldsymbol{S}_2$, we set the entry connecting node i with the pair (j,k) to $[\boldsymbol{S}_2]_{i,(j,k)} = 1$ iff $[\boldsymbol{S}_1]_{i,j} = 1, [\boldsymbol{S}_1]_{i,k} = 1$ and $[\boldsymbol{S}_1]_{j,k} = 1$, i.e., the triplet (i,j,k) is active. We generated 100 graph signals as training data based on a model $\boldsymbol{y} = f(\boldsymbol{x})$ and 100 test signals to compute the RMSE of estimating the signal $\boldsymbol{x}$ by inverse filtering $\boldsymbol{y}$. Since TVFs are linear in the filter coefficients, we used (regularized) least-squares to estimate the latter [25]. For the 1st-level TVF we consider filter orders $K = \{0,\ldots,16\}$ while for the 2nd-level TVF we only report results for $K = \{0,1,2,3\}, L_1, L_2 = \{0,1,2\}$ due to space constraints. Note that for the 2nd-level TVF we use the same K in both (5) and (7).

1st-level forward model. We first consider a 1st-level forward model:

$$\boldsymbol{y} = 0.2\boldsymbol{x} + 0.3\boldsymbol{S}_1\boldsymbol{x} + 0.01\boldsymbol{n}, \tag{9}$$

Authorized licensed use limited to: TU Delft Library. Downloaded on August 24,2021 at 12:39:04 UTC from IEEE Xplore. Restrictions apply.

Table 1. RMSE of 2nd-level TVF in inverse filtering of model (9), where row and column indices of each subtable are L_1 and L_2.

$K=0, L_1, L_2=\{0,1,2\}$			$K=1, L_1, L_2=\{0,1,2\}$		
0.58	0.58	0.57	0.43	0.43	0.43
0.58	0.57	0.57	0.43	0.43	0.43
0.58	0.57	0.56	0.43	0.43	0.43
$K=2, L_1, L_2=\{0,1,2\}$			$K=3, L_1, L_2=\{0,1,2\}$		
0.43	0.43	0.42	0.43	0.43	0.43
0.43	0.42	0.42	0.43	0.42	0.42
0.43	0.42	0.42	0.43	0.42	**0.42**

Table 2. RMSE of 2nd-level TVF in inverse filtering of model (10), where row and column indices of each subtable are L_1 and L_2.

$K=0, L_1, L_2=\{0,1,2\}$			$K=1, L_1, L_2=\{0,1,2\}$		
1.06	1.04	1.01	0.96	0.91	0.88
1.06	1.03	0.98	0.97	0.87	0.83
1.06	1.03	0.97	0.96	0.87	0.81
$K=2, L_1, L_2=\{0,1,2\}$			$K=3, L_1, L_2=\{0,1,2\}$		
0.82	0.79	0.76	0.80	0.76	0.74
0.82	0.76	0.73	0.80	0.74	0.70
0.82	0.76	0.72	0.80	0.74	**0.68**

where $\boldsymbol{x} = \mathbf{1}$, i.e., the all-one vector, and $\boldsymbol{n}$ follows a Gaussian distribution. For the 1st-level TVF, as the number of filter taps increases, the RMSE reduces, but it saturates at 0.75 (not shown). However, from Table 1, we observe that with a 2nd-level TVF, we can break this limit and obtain a smaller RMSE of 0.42 (44% improvement). It shows that a 2nd-level TVF can do much better than a 1st-level TVF, i.e., a conventional graph filter, in dealing with FIR modeled graph signals.

2nd-level forward model. We then generate the data following a 2n-level forward model:

$$\boldsymbol{y} = 0.05\boldsymbol{n} + 0.2\boldsymbol{x} + 0.3\boldsymbol{S}_1\boldsymbol{x} + 0.5\boldsymbol{S}_1^2\boldsymbol{x} + \boldsymbol{S}_1\boldsymbol{S}_2(0.2(\boldsymbol{x}\otimes\boldsymbol{x}) + 0.3(\boldsymbol{S}_1\boldsymbol{x}\otimes\boldsymbol{x}) + 0.3(\boldsymbol{S}_1\boldsymbol{x}\otimes\boldsymbol{S}_1\boldsymbol{x}) + 0.2(\boldsymbol{S}_1^2\boldsymbol{x}\otimes\boldsymbol{x})), \quad (10)$$

where $\boldsymbol{x}$ and $\boldsymbol{n}$ are defined as before. The RMSE of the 1st-level TVF fluctuates around 2.65 when K increases, and it reaches a lower bound of RMSE = 2.62 (not shown). If we contrast the latter with a 2nd-level TVF, we can see from Table 2 that the RMSE can be lowered by 74.1% via exploiting the relationships between node pairs and individual nodes.

2nd-level AR forward model. Finally, we generate the data with an AR-type 2nd-level forward model:

$$\boldsymbol{y} = \boldsymbol{x} + 0.3\boldsymbol{S}_1\boldsymbol{S}_2(\boldsymbol{y}\otimes\boldsymbol{y}) + 0.6\boldsymbol{S}_2(\boldsymbol{y}\otimes\boldsymbol{S}_1\boldsymbol{y}). \quad (11)$$

where we set $\boldsymbol{y} = \mathbf{1}$ and obtain $\boldsymbol{x}$ from (11). The least-squares solution for the 2nd-level TVF coefficients will be $h_{0,0,0} = 1, h_{1,0,0} = -0.3, h_{0,0,1} = -0.6$ and the signal $\boldsymbol{x}$ can be perfectly recovered. For the 1st-level TVF this is never the case and the achieved minimum error is RMSE= 1.64. From Table 3, we observe that as long as $K \geq 1, L_1 \geq 0, L_2 \geq 1$, we can achieve a perfect inverse filtering with a 2nd-level TVF.

4.2. Application: Recommender Systems

We also applied the 2nd-level TVF for rating prediction in recommender systems using the MovieLens100K dataset. We compare

Table 3. RMSE of 2nd-level TVF in inverse filtering of model (11), where row and column indices of each subtable are L_1 and L_2.

$K=0, L_1, L_2=\{0,1,2\}$			$K=1, L_1, L_2=\{0,1,2\}$		
1.16	0.40	0.38	0.37	0	0
1.16	0.39	0.36	0.37	0	0
1.16	0.39	0.36	0.37	0	0
$K=2, L_1, L_2=\{0,1,2\}$			$K=3, L_1, L_2=\{0,1,2\}$		
0.16	0	0	0.15	0	0
0.16	0	0	0.15	0	0
0.16	0	0	0.15	0	0

Table 4. RMSE of 2nd-level TVF in rating prediction, where K_1 is the number of hops in the 1st-level TVF, K_2 in the 2nd-level TVF.

K_1	2nd-level TVF with $K_2=2, (L_1, L_2)$					1st-level TVF
	$(0,0)$	$(1,0)$	$(2,0)$	$(2,1)$	$(3,1)$	
1	0.769	0.764	0.762	0.760	0.760	0.801
2	0.752	0.746	0.746	0.743	0.743	0.796
3	0.752	0.747	0.746	0.744	0.743	0.796
4	0.752	0.747	0.746	0.744	0.743	0.796
5	0.752	0.747	0.746	0.743	0.743	0.796
6	0.753	0.746	0.746	0.743	0.743	0.796

the 2nd-level TVF with the item-based multi-hop graph collaborative filtering (1st-level TVF) in [26, Eq. 20]. We considered the first 4646 ratings from 150 users and 200 movies with 4546 ratings for training and 100 for testing. Matrix $\boldsymbol{S}_1$ is an item similarity graph constructed from the training set following [26, Eq. 6]. We built the 2nd-level GSO as in the synthetic experiments. If triplet (i,j,k) is active, we set $[\boldsymbol{S}_2]_{i,(j,k)} = 0.5([\boldsymbol{S}_1]_{(i,j)} + [\boldsymbol{S}_1]_{(i,k)})$. Ratings are predicted by a 1st- and 2nd-level TVF collaborative filter (replacing the FIR filter in [26, Eq. 20] by a 2nd-level TVF) for the results reported in Table 4.

We observe that for both filters, as the number of filter taps increases, the rating prediction becomes better (since we round the values to 3 digits after the decimal, it is not visible after $K_1 > 2$). But after a certain number of hops, the 1st-level TVF, i.e., the method in [26], is not able to further improve the prediction performance. However, a 2nd-level TVF can reduce the RMSE beyond this limit, even when K_2, L_1, L_2 are small (all under 3). This is because extra information between one movie and a pair of movies is explored by a 2nd-level TVF in the prediction.

5. CONCLUSIONS

In this paper, we have proposed topological Volterra filters (TVFs). TVFs generalize traditional graph filters to capture also the influence of higher-level interactions and they generalize graph Volterra models to capture also multi-hop interactions. TVFs are built following the principles of their temporal counterparts by accounting for higher-level topological connections, and consequently, they present themselves as a nonlinear model to process and represent graph signals. An interesting feature of TVFs is that they are linear in the filter coefficients, thus allowing for a simple least-squares design. We have showcased the performance of TVFs for solving inverse problems on graphs and for rating prediction in recommender systems. Future research will be principally based on providing a spectral analysis for TVFs and corroborating their benefits for more applications.

Authorized licensed use limited to: TU Delft Library. Downloaded on August 24,2021 at 12:39:04 UTC from IEEE Xplore. Restrictions apply.

6. REFERENCES

[1] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[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 Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[3] Aliaksei Sandryhaila and José MF Moura, "Discrete signal processing on graphs," *IEEE transactions on signal processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

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

[5] Elvin Isufi, Andreas Loukas, Andrea Simonetto, and Geert Leus, "Autoregressive moving average graph filtering," *IEEE Transactions on Signal Processing*, vol. 65, no. 2, pp. 274–288, 2016.

[6] Santiago Segarra, Antonio G Marques, and Alejandro Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 4117–4131, 2017.

[7] Mario Coutino, Elvin Isufi, and Geert Leus, "Distributed edge-variant graph filters," in *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2017, pp. 1–5.

[8] Mario Coutino, Elvin Isufi, and Geert Leus, "Advances in distributed graph filtering," *IEEE Transactions on Signal Processing*, vol. 67, no. 9, pp. 2320–2333, 2019.

[9] S. Segarra, A. G. Marques, G. R. Arce, and A. Ribeiro, "Design of weighted median graph filters," in *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2017.

[10] V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner, "Adaptive graph filters in reproducing kernel hilbert spaces: Design and performance analysis," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 7, pp. 62–74, 2021.

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

[12] Michael Robinson, *Topological signal processing*, Springer, 2014.

[13] Michael T Schaub, Austin R Benson, Paul Horn, Gabor Lippner, and Ali Jadbabaie, "Random walks on simplicial complexes and the normalized hodge 1-laplacian," *SIAM Review*, vol. 62, no. 2, pp. 353–391, 2020.

[14] Michael T Schaub and Santiago 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.

[15] Marian Boguna, Ivan Bonamassa, Manlio De Domenico, Shlomo Havlin, Dmitri Krioukov, and M. Angeles Serrano, "Network geometry," 2020.

[16] 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, 2020.

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

[18] Adam C Wilkerson, Terrence J Moore, Ananthram Swami, and Hamid 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.

[19] Gonzalo Mateos, Santiago Segarra, Antonio G Marques, and Alejandro Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

[20] Lishan Qiao, Limei Zhang, Songcan Chen, and Dinggang Shen, "Data-driven graph construction and graph learning: A review," *Neurocomputing*, vol. 312, pp. 336–351, 2018.

[21] 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)*, 2020, pp. 3887–3891.

[22] Qiuling Yang, Mario Coutino, Gang Wang, Georgios B Giannakis, and Geert Leus, "Learning connectivity and higher-order interactions in radial distribution grids," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 5555–5559.

[23] F Maxwell Harper and Joseph A Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[24] Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst, and David K Hammond, "Gspbox: A toolbox for signal processing on graphs," *arXiv preprint arXiv:1408.5781*, 2014.

[25] Arthur E Hoerl and Robert W Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[26] Weiyu Huang, Antonio G Marques, and Alejandro R Ribeiro, "Rating prediction via graph signal processing," *IEEE Transactions on Signal Processing*, vol. 66, no. 19, pp. 5066–5081, 2018.

Authorized licensed use limited to: TU Delft Library. Downloaded on August 24,2021 at 12:39:04 UTC from IEEE Xplore. Restrictions apply.