Online Graph Filtering over Expanding Graphs

Das, Bishwadeep; Isufi, Elvin

DOI
10.1109/TSP.2024.3460194

Publication date
2024

Document Version
Final published version

Published in
IEEE Transactions on Signal Processing

Online Graph Filtering Over Expanding Graphs

Bishwadeep Das, *Student Member, IEEE*, and Elvin Isufi, *Senior Member, IEEE*

***Abstract*—Graph filters are a staple tool for processing signals over graphs in a multitude of downstream tasks. However, they are commonly designed for graphs with a fixed number of nodes, despite real-world networks typically grow over time. This topological evolution is often known up to a stochastic model, thus, making conventional graph filters ill-equipped to withstand such topological changes, their uncertainty, as well as the dynamic nature of the incoming data. To tackle these issues, we propose an online graph filtering framework by relying on online learning principles. We design filters for scenarios where the topology is both known and unknown, including a learner adaptive to such evolution. We conduct a regret analysis to highlight the role played by the different components such as the online algorithm, the filter order, and the growing graph model. Numerical experiments with synthetic and real data corroborate the proposed approach for graph signal inference tasks and show a competitive performance w.r.t. baselines and state-of-the-art alternatives.**

***Index Terms*—Graph filters, graph signal processing, online learning.**

I. Introduction

GRAPH filters are a well-established tool to process network data and have found use in a variety of applications, including node classification [2], [3], signal interpolation [4], and product recommendation [5]. They are a flexible parametric and localized operator that can process signals defined over the nodes through a weighted combination of successive shifts between neighbours [6]. Being the analogue of filters in discrete signal processing, graph filters can be interpreted in the graph frequency domain [7]. Compared to other tools such as graph kernels [8], filters do not need any prior knowledge about the data w.r.t. the topology when used for inference tasks.

Most of the filters in literature are designed out over graphs with a fixed number of nodes [6] despite graphs often growing through the addition of nodes, sometimes sequentially over time [9], [10]. An example is collaborative filtering in recommender systems where new users continuously join an existing network, e.g., a social network recommendation [11] or an abstract user-user collaborative filter network [12]. Such an expanding graph setting poses a three-fold challenge: (i) The data comes in a streaming nature, i.e., we do not have access to all the incoming nodes at once. This requires an on-the-fly filter design as batch-based solutions are no longer an alternative. (ii) The topology may evolve slowly or rapidly; hence, influencing the online filter design. (iii) The data over the incoming nodes is not guaranteed to follow a well-known distribution, thus requiring an adaptation of the filter to the task at hand. Often times, we may not even know how the incoming nodes connect to the existing graph. Typically, this happens in the absence of information for the incoming node, i.e., in pure cold-start recommendation, where we know nothing about user preferences, but we need to recommend items nevertheless [13]. The users may consume items later on, which can be used to infer their attachment but this can take time. Such challenges limit existing graph data processing methods which rely on the knowledge of the topology [14]. Another example where these challenges occur is in epidemic spreading over networks. We want to predict the future number of active cases for a city that is not yet affected but anticipates some cases shortly after. It may be difficult to obtain the underlying connections that influence the disease spread; hence, using statistical models is typically an option [9], [15]. In this scenario, filter design should account for the evolving topological model as well as for the data over it. This is possible by building upon online learning principles where the learning models are updated based on the incoming data stream [16], [17].

Existing works dealing with online learning over expanding graphs can be divided into Attachment, Feature Aware Methods, and Stochastic Methods. *Attachment and feature aware methods* know the connectivity of the incoming nodes and their features. For example, the work in [18] performs online node regression over fixed-size graphs by using their connectivity pattern to generate random kernel features [19]. An extension of this is the work in [20] which considers multi-hop connectivity patterns. Works like [21], [22], and [23] track changing attachment patterns over time but for graphs with a fixed number of node, which can be relevant for a large-scale setting. Another instance of online processing on expanding graphs is the work in [24] which obtains embeddings for signals over expanding graphs. Some works such as [25] classify an incoming node by using its features and the filter trained over the existing graph.

Received 1 February 2024; revised 12 July 2024; accepted 10 September 2024. Date of publication 16 September 2024; date of current version 23 October 2024. This research was supported in part by the TTW-OTP project GraSPA under Project 19497, in part by Dutch Research Council (NWO), and in part by the TU Delft AI Programme. Preliminary results were presented at the 2022 IEEE Asilomar Conference on Signals, Systems and Computations, Pacific Grove, USA [DOI: 10.1109/IEEECONF56349.2022.10052045]. The associate editor coordinating the review of this article and approving it for publication was Dr. Paolo Di Lorenzo. *(Corresponding author: Bishwadeep Das.)*

The authors are with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628XE Delft, The Netherlands (e-mail: b.das,e.isufi-1@tudelft.nl).

Digital Object Identifier 10.1109/TSP.2024.3460194

Then, there are works such as [26] that classify a stream of incoming nodes by using their features to estimate the attachment. The work in [27] classifies incoming nodes using spectral embeddings updated from the known attachment information. The kernel-based methods in this category [18], [20] rely on pre-selecting a suitable kernel that can fit the data which may be challenging to obtain. Additionally, the works in [28], [29], [30], [31] develop distributed solutions to estimate the filter parameters locally at each node. Differently, in this paper, we work with a centralized approach to estimate the filter, as we focus more on the expanding graph scenario. All in all, these methods concern either a graph with a fixed or a streaming number of nodes but with available attachment or feature information that may be unavailable.

Stochastic methods deal with unknown incoming node attachment and use models for it. For example, [32] uses heuristic stochastic attachment model to design graph filters only for one incoming node, while [33] learns an embedding by using a stochastic attachment to influence the propagation. In our earlier works [34], [35], we learn the attachment behaviour for inference with a fixed filter. However, this approach is limited to studying the effect of one node attaching with unknown connectivity and it assumes a pre-trained filter over the existing graph. Differently, here we consider the filter design over a stream of incoming nodes.

We perform online graph filtering over a stream of incoming nodes when the topology is both known and unknown. Our contribution is threefold:

1) We develop an online filter design framework for inference over expanding graphs. This is done by casting the inference problem as a time-varying loss function over the existing topology, data, and the incoming node attachment. Subsequently, we update the filter parameters via online learning principles.
2) We adapt the online filter design problem to two scenarios: (i) the *deterministic* setting where the connectivity of each incoming node is available; (ii) the *stochastic* setting where this connectivity is unavailable. For both settings, we conduct a regret analysis to discuss the influence of the incoming node attachment and the role of the graph filter.
3) We develop an online ensemble and adaptive stochastic update where, in addition to the filter parameters, we also learn the combination parameters of the different stochastic attachment rules. This concerns the stochastic setting where a single attachment model might be insufficient. We also discuss the regret in this setting and analyze how the ensemble affects it.

We corroborate the proposed approach with numerical experiments on synthetic and real data from recommender systems and COVID cases prediction. Results show that the online filters perform better than other alternatives like kernels or pre-trained filters; and, that stochastic online filters can also perform well w.r.t. deterministic approaches.

This paper is structured as follows. Sec. II elaborates on the sequentially expanding graph scenario, along with the basic formulation of online inference with graph filters. Sec. III and IV contain the online learning methods and their respective analysis in the deterministic and in the stochastic setting, respectively. Sec. V contains the numerical results, while Sec. VI concludes the paper. All proofs are collected in the appendix.

II. Problem Formulation

Consider a starting graph $\mathcal{G}_0 = \{\mathcal{V}_0, \mathcal{E}_0\}$ with node set $\mathcal{V}_0 = \{v_{0,1}, \ldots, v_{0,N_0}\}$ of N_0 nodes, edge set $\mathcal{E}_0$, and adjacency matrix $\mathbf{A}_0 \in \mathbb{R}^{N_0 \times N_0}$, which can be symmetric or not, depending on the type of graph (undirected or directed). Let $v_1, \ldots, v_T$ be a set of T sequentially incoming nodes where at time t, node v_t attaches to graph $\mathcal{G}_{t-1}$ forming the graph $\mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t\}$ with $N_t = N_0 + t$ nodes, M_t edges, and adjacency matrix $\mathbf{A}_t \in \mathbb{R}^{N_t \times N_t}$. The connectivity of v_t is represented by the attachment vector $\mathbf{a}_t = [a_1, \ldots, a_{N_{t-1}}]^\top \in \mathbb{R}^{N_{t-1}}$, where a non-zero element implies a directed edge from $v \in \mathcal{V}_{t-1}$ to v_t. This connectivity suits inference tasks at v_t, where the existing nodes influence the incoming ones. This is the case of cold-starters in graph-based collaborative filtering [5], [12]. Here, the nodes represent existing users, the edges capture similarities among them (e.g., Pearson correlation), and a cold starter is a new node that attaches to this user-user graph. The task is to collaboratively infer the preference of the cold-starter from the existing users [36].

Depending on the availability of $\mathbf{a}_t$, we can have a deterministic attachment setting or a stochastic attachment setting. In a deterministic setting, the incoming node attachment vector $\mathbf{a}_t$ is known or it is estimated when v_t appears. This occurs in growing physical networks or in collaborative filtering where side information is used to establish the connectivity [12]. The expanded adjacency matrix $\mathbf{A}_t \in \mathbb{R}^{N_t \times N_t}$ reads as

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1} & \mathbf{0}_{N_{t-1}} \\ \mathbf{a}_t^\top & 0 \end{bmatrix} \tag{1}$$

where $\mathbf{A}_{t-1}$ is the $N_{t-1} \times N_{t-1}$ adjacency matrix and $\mathbf{0}_{N_{t-1}}$ is the all-zero vector of size (N_{t-1}). In a stochastic setting, $\mathbf{a}_t$ is unknown (at least before inference), which is the typical case in cold start collaborative filtering [35]. A new user/item enters the system and we have neither side information nor available ratings to estimate the connectivity. The attachment of v_t is modelled via stochastic models from network science [9]. Node v_t attaches to $v_i \in \mathcal{V}_{t-1}$ with probability $p_{i,t}$ forming an edge with weight $w_{i,t}$. The probability vector $\mathbf{p}_t = [p_{1,t}, \ldots, p_{N_{t-1},t}]^\top \in \mathbb{R}^{N_{t-1}}$ and the weight vector $\mathbf{w}_t = [w_{1,t}, \ldots, w_{N_{t-1},t}]^\top \in \mathbb{R}^{N_{t-1}}$ characterize the attachment and imply that $[\mathbf{a}_t]_i = w_{i,t}$ with probability $p_{i,t}$, and zero otherwise. We consider vector $\mathbf{a}_t$ be composed of independent, weighted Bernoulli random variables with respective mean and covariance matrix

$$\mathbb{E}[\mathbf{a}_t] = \mathbf{p}_t \circ \mathbf{w}_t \; ; \; \mathbf{\Sigma}_t = \operatorname{diag}(\mathbf{w}_t^{\circ 2} \circ \mathbf{p}_t \circ (\mathbf{1} - \mathbf{p}_t)) \tag{2}$$

where $\operatorname{diag}(\mathbf{x})$ is a diagonal matrix with $\mathbf{x}$ comprising the diagonal elements, and $\mathbf{x}^{\circ 2} = \mathbf{x} \circ \mathbf{x}$ is the element-wise product of a vector $\mathbf{x}$ with itself. The new adjacency matrix for a realization $\mathbf{a}_t$ is the same as in (1). The attachment is revealed after the inference task; e.g., after a cold start user has consumed one or more items and we can estimate it.

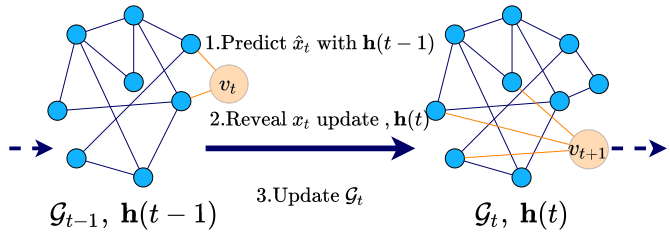

Fig. 1. Online filter learning process at time t through the addition of node v_t with signal x_t and the update of the filter $\mathbf{h}(t)$. (Left) node v_t attaches to the previous graph $\mathcal{G}_{t-1}$ forming $\mathcal{G}_t$; the edges in blue denote the existing edges while those in orange denote the edges formed by the incoming node; (Centre) Signal x_t is predicted, then the true value x_t is revealed, and the filter parameter $\mathbf{h}(t)$ is updated from $\mathbf{h}(t-1)$; (Right) the next node v_{t+1} attaches to $\mathcal{G}_t$.

A. Filtering Over Expanding Graphs

Let $\mathbf{x}_t \in \mathbb{R}^{N_t}$ be the graph signal over graph $\mathcal{G}_t$, which writes in terms of the previous signal $\mathbf{x}_{t-1} \in \mathbb{R}^{N_{t-1}}$ as $\mathbf{x}_t = [\mathbf{x}_{t-1}, x_t]^\top$ with x_t being the signal at the latest incoming node v_t. To infer x_t, we consider the temporary graph signal $\tilde{\mathbf{x}}_t = [\mathbf{x}_{t-1}, 0]^\top$ where the zero at v_t indicates that its value is unknown. To process such signals we use graph convolutional filters, which are linear and flexible tools for processing them [6]. A filter of order K acts on $\tilde{\mathbf{x}}_t$ to generate the output $\tilde{\mathbf{y}}_t$ on graph $\mathcal{G}_t$ as

$$\tilde{\mathbf{y}}_t = \sum_{k=0}^{K} h_k \mathbf{A}_t^k \tilde{\mathbf{x}}_t \tag{3}$$

where h_k is the weight given to the kth shift $\mathbf{A}_t^k \tilde{\mathbf{x}}_t$. Substituting the kth adjacency matrix power

$$\mathbf{A}_t^k = \begin{bmatrix} \mathbf{A}_{t-1}^k & \mathbf{0}_{N_{t-1}} \\ \mathbf{a}_t^\top \mathbf{A}_t^{k-1} & 0 \end{bmatrix} \tag{4}$$

into (3), we write the filter output as

$$\tilde{\mathbf{y}}_t = \begin{bmatrix} \sum_{k=0}^{K} h_k \mathbf{A}_{t-1}^k \mathbf{x}_t \\ \mathbf{a}_t^\top \sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1} \mathbf{x}_t \end{bmatrix} \tag{5}$$

where we grouped w.l.o.g. the output at the incoming node v_t in the last entry. I.e.,

$$[\tilde{\mathbf{y}}_t]_{N_t} := \hat{x}_t = \mathbf{a}_t^\top \sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1} \mathbf{x}_t = \mathbf{a}_t^\top \mathbf{A}_{x,t-1} \mathbf{h}. \tag{6}$$

Here $\mathbf{h} = [h_1 \ldots, h_K]^\top \in \mathbb{R}^K$ collects the filter coefficients and $\mathbf{A}_{x,t-1} = [\mathbf{x}_t, \mathbf{A}_{t-1}\mathbf{x}_t, \ldots, \mathbf{A}_{t-1}^{K-1}\mathbf{x}_t] \in \mathbb{R}^{N_{t-1} \times K}$ contains the higher-order shifts of $\mathbf{x}_t$. The coefficient h_0 does not play a role in the output $\hat{x}_t$, thus the zero in the N_tth position of $\tilde{\mathbf{x}}_t$ does not influence the inference task on the incoming node. In the stochastic setting, the output is random as it depends on the attachment rule. In turn, this needs a statistical approach to characterize both the filter and its output. We shall detail this in Section IV.

Remark 1: The above discussion considers one incoming node at a time which is common in the streaming setting. The analysis can be extended to multiple nodes arriving at a certain time interval. Here, we consider inference tasks where the existing nodes affect the incoming streaming ones. For tasks where the influence is bidirectional, the adjacency matrix in (1) is symmetric and the analysis follows analogously. One way to do this is to build on [32], where we discuss the case for a single incoming node.

B. Online Filter Learning

Our goal is to process signal $\tilde{\mathbf{x}}_t$ to make inference on the incoming nodes v_t by designing the filters in (3). We consider a data-driven setting where we estimate the filter parameters from a training set $\mathcal{T} = \{v_t, x_t, \mathbf{a}_t\}_{t=1:T}$ in which each datum comprises an incoming node v_t, its signal x_t, and the attachment vector $\mathbf{a}_t$. Given set $\mathcal{T}$, we find the filter parameters $\mathbf{h}$ by solving

$$\underset{\mathbf{h} \in \mathbb{R}^K}{\operatorname{argmin}} \sum_{t=1}^{T} f_t(\hat{x}_t, x_t; \mathbf{h}) + r(\mathbf{h}) \tag{7}$$

where $f_t(\hat{x}_t, x_t; \mathbf{h}) := f_t(\mathbf{h}, x_t)$ measures the goodness of fit between the prediction $\hat{x}_t$ and the true signal x_t and $r(\mathbf{h})$ is a regularizer. For example, $f_t(\cdot, \cdot)$ can be the least-squares error for regression problems such as signal denoising or interpolation; or the logistic error for classification problems such as assigning a class label to node v_t. For convex and differentiable $f_t(\cdot, \cdot)$ and $r(\mathbf{h})$, we can find an optimal filter that solves the batch problem over $\mathcal{T}$. However such a solution is not ideal since the incoming nodes v_t are streaming and evaluating a new batch for each v_t is computationally demanding. A batch solution also suffers in non-stationary environments where the test set distribution differs from $\mathcal{T}$. Targeting a non-stationary setting with incoming nodes, we turn to online learning to update the filter parameters on-the-fly [17].

We initialize the filter before the arrival of incoming nodes, $\mathbf{h}(0)$ by training a filter over $\mathcal{G}_0$, using $\mathbf{A}_0$ and $\mathbf{x}_0$. The training follows a regularized least square problem with ℓ_2 norm squared loss on the filter. We call this *Pre-training*. In high-level terms, the online filter update proceeds at time t as follows:

1) The environment reveals the node v_t and its attachment $\mathbf{a}_t$ in the deterministic setting.
2) We use the filter at time t, $\mathbf{h}(t-1)$ to infer the signal value $\hat{x}_t$ at the incoming node using (6).
3) The environment reveals the loss as a function of the filter $\mathbf{h}(t-1)$ and the true signal x_t as
$$l_t(\mathbf{h}, x_t) = f_t(\mathbf{h}, x_t) + r(\mathbf{h}) \tag{8}$$
which is evaluated at $\mathbf{h}(t-1)$.
4) We update the filter parameters $\mathbf{h}(t)$ based on the loss and the current estimate $\mathbf{h}(t-1)$.
5) In the stochastic setting, the true attachment $\mathbf{a}_t$ is revealed.

With this in place, our problem statement reads as follows:

Problem statement: Given the starting graph $\mathcal{G}_0 = \{\mathcal{V}_0, \mathcal{E}_0\}$, adjacency matrix $\mathbf{A}_0$, graph signal $\mathbf{x}_0$, and the training set $\mathcal{T}$, our goal is to predict online a sequence of graph filters $\{\mathbf{h}(t)\}$ w.r.t. loss functions $l_t(\mathbf{h}, x_t)$ to process signals at the incoming nodes for both the deterministic and the stochastic attachments.

Algorithm 1 Deterministic Online Graph Filtering (**D-OGF**)

Input: Graph $\mathcal{G}_0$, $\mathbf{A}_0$, $\mathbf{x}_0$, $\mathcal{T}=\{v_t, x_t, \mathbf{a}_t\}_{t=1:T}$.
Initialize: Pre-train $\mathbf{h}(0)$ over $\mathcal{G}_0$ using $\mathbf{A}_0$, $\mathbf{x}_0$.
for $t=1:T$ **do**
  Obtain v_t and true connection $\mathbf{a}_t$, update $\mathbf{A}_t$
  Predict $\hat{x}_t=\mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h}(t-1)$ (cf. (6))
  Reveal loss $l_t(\mathbf{h}, x_t)$ (cf. (9))
  Update $\mathbf{h}(t)$ using (10)
  Update $\mathbf{x}_t$
end for

III. Deterministic Online Filtering

Targeting regression tasks[1], we can take $f_t(\mathbf{h}, x_t)$ as the squared error and $r(\mathbf{h})$ as the scaled l_2-norm penalty to define the loss

$$l_t(\mathbf{h}, x_t)=\frac{1}{2}(\mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h}-x_t)^2+\mu||\mathbf{h}||_2^2 \tag{9}$$

where $\mu>0$. For the online update, we perform projected online gradient descent [16], which comprises one projected gradient descent step evaluated at $\mathbf{h}(t-1)$ as

$$\mathbf{h}(t)=\prod_{\mathcal{H}}(\mathbf{h}(t-1)-\eta\nabla_h l_t(\mathbf{h}, x_t)|_{\mathbf{h}(t-1)}) \tag{10}$$

with set $\mathcal{H}$ bounding the filter energy $\mathcal{E}(\mathbf{h})=||\mathbf{h}||_2^2$ and $\prod_{\mathcal{H}}(\cdot)$ denotes the projection operator on $\mathcal{H}$. Here, $\eta>0$ is the step size, and the gradient has the expression

$$\nabla_h l_t(\mathbf{h}, x_t)=(\mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h}-x_t)\mathbf{A}_{x,t-1}^\top \mathbf{a}_t+2\mu\mathbf{h}. \tag{11}$$

The gradient depends on $\mathbf{a}_t$ through the term $(\hat{x}_t-x_t)$ $\mathbf{A}_{x,t-1}^\top \mathbf{a}_t$. Operation $\mathbf{A}_{x,t-1}^\top \mathbf{a}_t$ is a weighted combination of only those columns of $\mathbf{A}_{x,t-1}^\top$ where the corresponding entry of $\mathbf{a}_t$ is non-zero. In turn, each column of $\mathbf{A}_{x,t-1}^\top$ contains shifted graph signals at each node, which get scaled by the difference between the predicted and the true signal x_t, ultimately, indicating that a larger residue leads to a larger gradient magnitude. The online learner in (10) updates the filter parameters for every incoming node.

Algorithm 1 summarizes the learning process. The computational complexity of the online update at time t is of order $\mathcal{O}(K(M_t+M_{\max}))$, where $M_{\max}$ is the maximum number of edges formed by v_t across all t. Note that $M_{\max}\ll N_{t-1}$, i.e., the maximum number of edges formed by any incoming node is smaller than the existing number of nodes. Appendix D breaks down this complexity.

Regret analysis. We analyze the deterministic online graph filtering algorithm to understand the effect of the filter updates and how the expanding graph influences it. Specifically, we conduct a regret analysis that quantifies the performance difference between the online updates and the static batch solution where all the incoming node information is available. The normalized regret w.r.t. a fixed filter $\mathbf{h}^\star$ is defined as

$$\frac{1}{T}R_T(\mathbf{h}^\star)=\frac{1}{T}\sum_{t=1}^{T} l_t(\mathbf{h}(t-1), x_t)-l_t(\mathbf{h}^\star, x_t) \tag{12}$$

where $\sum_{t=1}^{T} l_t(\mathbf{h}(t-1), x_t)$ is the cumulative loss incurred by the online algorithm. The regret measures how much better or worse the online algorithm performs over the sequence compared to a fixed learner. An upper bound on the regret indicates the worst-case performance and it is of theoretical interest. If this bound is sub-linear in time, the average regret tends to zero as the sample size grows to infinity, i.e., $\lim_{T\to\infty}\frac{1}{T}R_T(\mathbf{h}^\star)=0$ [38]. This indicates that the algorithm is learning. We assume the following.

Assumption 1: The incoming nodes form a maximum of $M_{max}\ll N_t$ edges for all t.

Assumption 2: The attachment vectors $\mathbf{a}_t$ and the stochastic model-based weight vectors $\mathbf{w}_t$ are upper-bounded by a scalar w_h. I.e., for all t we have

$$[\mathbf{a}_t]_n\le w_h,\ [\mathbf{w}_t]_n\le w_h. \tag{13}$$

Assumption 3: The filter parameters $\mathbf{h}$ are upper-bounded in their energy, i.e., $\mathcal{E}(\mathbf{h})=||\mathbf{h}||_2^2\le H^2$.

Assumption 4: For all attachment vectors $\mathbf{a}_t$, the residue $r_t=\mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h}-x_t$ is upper-bounded. I.e., there exists a finite scalar $R>0$ such that $|r_t|\le R$.

Assumption 1 holds for graphs in the real world. A node makes very few connections compared to the total number of nodes, i.e., the attachment vector $\mathbf{a}_t$ will be sparse with a maximum of M_{max} non-zero entries. Note that our stochastic model does not take this into account. Assumption 2 bounds all edge-weights which is commonly observed. Assumption 3 ensures finite parameters which mean the filter output does not diverge. This can be guaranteed by projection on to $\mathcal{H}$. Assumptions 3 and 4 imply bounded filter outputs. Then, we claim the following.

Proposition 1: Consider a sequence of Lipschitz losses $\{l_t(\mathbf{h}, x_t)\}$ with Lipschitz constants L_d, $[cf.\ (9)]$ and a learning rate $\eta[cf.(10)]$. Let also Assumptions 1–4 hold. The normalized static regret $R_T(\mathbf{h}^\star)$ for the online algorithm generating filters $\{\mathbf{h}(t)\}\in\mathcal{H}$ relative to the optimal filter $\mathbf{h}^\star\in\mathcal{H}$ is upper-bounded as

$$\frac{1}{T}R_T(\mathbf{h}^\star)\le\frac{||\mathbf{h}^\star||_2^2}{2\eta T}+\frac{\eta}{2}L_d^2 \tag{14}$$

with $L_d=RC+2\mu H$ where $||\mathbf{A}_{x,t-1}^\top \mathbf{a}_t||_2\le C$.

Proof: From Lemma 1 in Appendix D, we have that the loss functions are Lipschitz. Then, we are in the setting of [Thm. 2.13, 16] from which the rest of the proof follows. □

There are two main filter-related factors that influence the regret bound in (14): the filter energy H^2 and the residual energy R^2. A smaller H can lead to a lower bound but it can also increase the prediction error by constraining the parameter set too much. Moreover, a higher regularization weight μ also penalizes high filter energies $||\mathbf{h}||_2^2$. So, for the projected online learner with a high regularization weight μ, a high H can help

[1]For classification tasks, we can consider the surrogate of the gradient of the logistic loss which is also convex and differentiable, as seen in [37].

the inference task, even if it increases the regret bound. Second, from Assumption 4, the residue R is likely small when a filter approximates well the signal on the incoming node. This can happen when the signal values on the incoming node and the existing nodes are similar or when the existing topology and signals over it are expressive enough to represent the incoming node values. Examples of the latter are locally smooth graph signals that can be approximated by a low order filter K. For high values of K, all nodes have similar signals, implying that many potential attachment patterns can generate x_t. This would make that the manner of attachment irrelevant.

IV. Stochastic Online Filtering

Often, the true attachment for the incoming nodes is initially unknown and it is only revealed afterwards. This is the case with rating prediction for cold start recommender systems, where users have initially little to no information, and thus, their connections cannot be inferred. However, their connections can be inferred after they have consumed some items. Instead of waiting for feedback, we can use expanding graph models to infer the signal value and subsequently update the filter online. To address this setting, we first propose an online stochastic update for the filters via specific heuristic models. Then, we propose an adaptive stochastic approach that learns also from an ensemble of topological expansion models.

A. Heuristic Stochastic Online Filtering

We model the connectivity of node v_t via random stochastic models. Specifically, we use the existing topology $\mathbf{A}_{t-1}$ to fix the attachment probabilities $\mathbf{p}_t$ and weights $\mathbf{w}_t$ using a heuristic attachment rule. Given $\mathbf{a}_t$ is a random vector, the environment reveals the statistical loss

$$l_t(\mathbf{h}, x_t) = \mathbb{E}[f_t(\mathbf{h}, x_t)] + r(\mathbf{h}) \tag{15}$$

where the expectation concerns the stochastic attachment model. For $f_t(\mathbf{h}, x_t)$ being the squared loss, we have the mean squared error expression

$$\begin{aligned} l_t(\mathbf{h}, x_t) &= \mathbb{E}\left[\frac{1}{2}(\mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)^2\right] + \mu||\mathbf{h}||_2^2 \\ &= \frac{1}{2}((\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)^2 \\ &\quad + \frac{1}{2}(\mathbf{A}_{x,t-1}\mathbf{h})^\top \mathbf{\Sigma}_t \mathbf{A}_{x,t-1}\mathbf{h} + \mu||\mathbf{h}||_2^2 \end{aligned} \tag{16}$$

where $\mathbf{\Sigma}_t$ is the attachment covariance matrix [cf. (2)]. The first term on the r.h.s. of (16), $s_t^2 = \frac{1}{2}((\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)^2$ is the squared bias between the expected model output $(\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1}\mathbf{h}$ and the true signal x_t. The second term $\frac{1}{2}(\mathbf{A}_{x,t-1}\mathbf{h})^\top \mathbf{\Sigma}_t \mathbf{A}_{x,t-1}\mathbf{h}$ is the variance of the predicted output, and the third term penalizes a high l_2-norm of $\mathbf{h}$.[2] The projected online gradient descent update of the filter parameters $\mathbf{h}(t)$ is

$$\mathbf{h}(t) = \prod_{\mathcal{H}}(\mathbf{h}(t-1) - \eta \nabla_h l_t(\mathbf{h}, x_t)|_{\mathbf{h}(t-1)}) \tag{17}$$

[2]We could also consider adding a penalty parameter to the variance contribution if we want to tweak the bias-variance trade-off in the filter update.

Algorithm 2 Stochastic Online Graph Filtering (**S-OGF**)

Input: Graph $\mathcal{G}_0$, $\mathbf{A}_0$, $\mathbf{x}_0$, $\mathcal{T} = \mathcal{T} = \{v_t, x_t, \mathbf{a}_t\}_{t=1:T}$
Initialize: Pre-train $\mathbf{h}^s(0)$ over $\mathcal{G}_0$ using $\mathbf{A}_0$, $\mathbf{x}_0$.
for $t = 1 : T$ **do**
  Obtain v_t and $\mathbf{p}_t$, $\mathbf{w}_t$ following preset heuristics
  Predict $\hat{x}_t = (\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1}\mathbf{h}^s(t-1)$
  Incur loss $l_t^s(\mathbf{h}, x_t)$ [cf. (16)]
  Update $\mathbf{h}^s(t)$ using (17)
  Reveal $\mathbf{a}_t$, update $\mathbf{A}_t$ and $\mathbf{x}_t$
end for

with gradient

$$\begin{aligned} \nabla_h l_t(\mathbf{h}, x_t) = ((\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)\mathbf{A}_{x,t-1}^\top \\ (\mathbf{w}_t \circ \mathbf{p}_t) + \mathbf{A}_{x,t-1}^\top \mathbf{\Sigma}_t \mathbf{A}_{x,t-1}\mathbf{h} + 2\mu\mathbf{h}. \end{aligned} \tag{18}$$

The stochastic loss [cf. (16)] is differentiable and strongly convex in $\mathbf{h}$. Following Assumption 4, the bias s_t, and the gradient (18) are also upper-bounded, making the loss Lipschitz. Algorithm 2 summarizes learning in this setting. The complexity of the online stochastic filter learning at time t is of order $\mathcal{O}(K(M_t + N_t))$. Check Appendix D for further details. Note the dependency on N_t, the size of the graph at time t. Since we do not know the true attachment at the time of making the prediction, the stochastic attachment model assigns probabilities to each node, along with the weights. This leads to the dependance on N_t while making the prediction [cf. (16)]. This does not exist in the deterministic case, as we know $\mathbf{a}_t$.

Regret analysis: To characterize the role of the stochastic topological model on the filter update, we compare the cumulative loss between the online stochastic update and the deterministic batch solution. This allows quantifying the performance gap by not knowing the attachment pattern. The regret reads as

$$\frac{1}{T}R_{s,T}(\mathbf{h}^\star) = \frac{1}{T}\sum_{t=1}^{T} l_t^s(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^\star, x_t) \tag{19}$$

where $l_t^s(\cdot)$ denotes the stochastic loss and $l_t^d(\cdot)$ the deterministic loss. Similarly, $\mathbf{h}^d(t-1)$ and $\mathbf{h}^s(t-1)$ denote the online filter at time $t-1$ in the deterministic and stochastic settings, respectively. We claim the following.

Theorem 1: At time t, let graph $\mathcal{G}_{t-1}$ have N_{t-1} nodes and $\mathbf{h}^s(t-1)$, $\mathbf{h}^d(t-1)$ be the filters learnt online in the stochastic and deterministic scenarios, respectively. Let the nth element of probability vector $\mathbf{p}_t$ be $[\mathbf{p}_t]_n$. Given Assumptions 1–4, the Lipschitz constant L_d, and learning rate η, the normalized static regret for the stochastic setting is upper-bounded as

$$\begin{aligned} \frac{1}{T}R_{s,T}(\mathbf{h}^\star) \leq \frac{1}{T}\Bigg(\sum_{t=1}^{T} w_h^2 Y^2(||\mathbf{p}_t||_2^2 + M_{max}) \\ + 2Rw_h Y\sqrt{||\mathbf{p}_t||_2^2 + M_{max}} + w_h^2 Y^2 \bar{\sigma}_t^2 \\ + L_d||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||\Bigg) \\ + \frac{||\mathbf{h}^\star||_2^2}{2\eta} + \frac{\eta}{2}L_d^2 T \end{aligned} \tag{20}$$

where $\bar{\sigma}_t^2 = \max_{n=1:N_{t-1}} [\mathbf{p}_t]_n (1 - [\mathbf{p}_t]_n)$ and $||\mathbf{A}_{x,t-1}\mathbf{h}||_2 \leq Y$.

Proof: See Appendix A. □

The regret bound in (20) depends on the stochastic expanding model and the incoming data as follows:

- The sum of squared norms of the probability vectors corresponding to the attachment rule, via the terms $\sum_{t=1}^{T} ||\mathbf{p}_t||_2^2$ and $\sum_{t=1}^{T} \sqrt{||\mathbf{p}_t||_2^2 + M_{max}}$. This makes the choice of attachment probability $\mathbf{p}_t$ important as it influences the online learner. For example if $\mathbf{p}_t = \mathbf{1}_{N_{t-1}}$ for all t, the sum $\sum_{t=1}^{T} ||\mathbf{p}_t||_2^2$ is of the order T^2, which means the regret bound diverges. Thus, the attachment rule should be selected such that $\sum_{t=1}^{T} ||\mathbf{p}_t||_2^2$ is of order $\mathcal{O}(T)$ or less. However, not all decaying attachment probabilities will reduce the bound reducing. It is necessary to have an inverse dependence on N_t, as is the case for the uniform distribution. This is for example the case of the uniformly at random attachment as we elaborate in Corollary 1.
- The term $\frac{w_h^2 Y^2}{T} \sum_{t=1}^{T} \bar{\sigma}_t^2$ is the sum of the maximum variance for an attachment rule $\bar{\sigma}_t^2$ over time. The maximum value of $\bar{\sigma}_t^2$ is 0.25, attained for an attachment probability of 0.5. For an attachment rule which has either high or low attachment probabilities per node, $\bar{\sigma}_t^2$ will be low, thus contributing less to the regret bound. This means a lower regret can result from stochastic attachment rules with a smaller uncertainty in attachment over the nodes.
- The average distance between the stochastic and deterministic filters over the sequence $\frac{1}{T} \sum_{t=1}^{T} ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2$. If the filter trained with a stochastic attachment is further away from the filter updated with known attachment, the regret is higher. This can happen when the attachment rule cannot model the incoming node attachment and the filter prediction incurs a higher squared error. However, we can use this term to modify the filter update. One way to do this is to include a correction step to update the online filter after the true connection has been revealed. We will discuss this in Remark 2.
- The term $\frac{||\mathbf{h}^\star||_2^2}{2\eta T} + \frac{\eta}{2} L_d^2$ suggests similar factors which affect the deterministic regret will also affect the stochastic regret [cf. (14)].

We now present how this regret bound reduces for the uniformly at random attachment.

Corollary 1: Consider a uniformly at random attachment with $[\mathbf{p}_t]_n = \frac{1}{N_{t-1}}$. As the sequence length grows to infinity, i.e., $T \to \infty$, the regret upper bound becomes

$$\frac{1}{T} R_{s,T}(\mathbf{h}^\star) \leq w_h^2 M_{max} Y^2 + R w_h Y (M_{max} + 1) + \frac{1}{T} \sum_{t=1}^{T} L_d ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2 + \frac{||\mathbf{h}^\star||_2^2}{2\eta T} + \frac{\eta}{2} L_d^2 \tag{21}$$

Proof: See Appendix B. □

Corollary 1 shows that the regret bound in (20) can be improved upon with the right choice of attachment rules. Even though the attachment rule helps, there is a chance it fails to model the true attachment process, in which case the term $\frac{1}{T} \sum_{t=1}^{T} L_d ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2$ can diverge, ultimately, not making the learner not useful in the steady state.

B. Adaptive Stochastic Online Filtering

Oftentimes, a single attachment rule cannot describe the connectivity of the incoming nodes and an ensemble of stochastic rules is needed. This is seen in the regret bound in Theorem 1, which depends on the distance between the stochastic and deterministic online filters. This poses the additional challenge of how to combine these rules for the growing graph scenario. To tailor the combined rule to the online setting, we consider a linear combination of different attachment models and update the parameters as we do for the filter coefficients. Specifically, consider M attachment rules parameterized by the probability vectors $\{\mathbf{p}_{m,t}\}_{m=1:M}$ and the corresponding weight vectors $\{\mathbf{w}_{m,t}\}_{m=1:M}$. Here, $[\mathbf{p}_{m,t}]_i$ denotes the probability of v_t attaching to $v_i \in \mathcal{V}_{t-1}$ under the mth rule and $[\mathbf{w}_{m,t}]_i$ the corresponding weight. Upon defining the dictionaries $\mathbf{P}_{t-1} = [\mathbf{p}_{1,t}, \ldots, \mathbf{p}_{M,t}] \in \mathbb{R}^{N_{t-1} \times M}$ and $\mathbf{W}_{t-1} = [\mathbf{w}_{1,t}, \ldots, \mathbf{w}_{N,t}] \in \mathbb{R}^{N_{t-1} \times M}$, we combine these models as

$$\bar{\mathbf{p}}_t = \mathbf{P}_{t-1}\mathbf{m} \quad \text{and} \quad \bar{\mathbf{w}}_t = \mathbf{W}_{t-1}\mathbf{n} \tag{22}$$

where the combination parameters $\mathbf{m}$ and $\mathbf{n}$ belong to the probability simplex

$$\mathcal{S}^M = \{\boldsymbol{\alpha} \in \mathbb{R}^M,\ \mathbf{1}_M^\top \boldsymbol{\alpha} = 1,\ \boldsymbol{\alpha} \succeq \mathbf{0}_M\}. \tag{23}$$

with $\mathbf{1}_M$ being the vector of M ones. For an existing node v_i, the ith row of $\mathbf{P}_t$ contains the corresponding rule-based probabilities. Equation (22) ensures that $\bar{\mathbf{p}}_t$ represents a composite probability vector of attachment with $[\bar{\mathbf{p}}_t]_i = \sum_{l=1}^{M} m_l [\mathbf{p}_{l,t}]_i$ representing the probability of v_t attaching to v_i. It also ensures that the weights in $\bar{\mathbf{w}}_t$ lie in $[0, w_h]$. By representing the expanding graph model via the latent vectors $\mathbf{m}$ and $\mathbf{n}$, we can analyze them in lieu of the growing nature of the problem. This eases the setting as both $\bar{\mathbf{p}}_t$ and $\bar{\mathbf{w}}_t$ grow in dimensions with t since learning these values directly becomes challenging.

Online learner: The instantaneous stochastic loss becomes

$$l_t(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t) = \frac{1}{2}((\mathbf{W}_{t-1}\mathbf{n} \circ \mathbf{P}_{t-1}\mathbf{m})^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)^2 + \frac{1}{2}(\mathbf{A}_{x,t-1}\mathbf{h})^\top \bar{\boldsymbol{\Sigma}}_t \mathbf{A}_{x,t-1}\mathbf{h} + \mu ||\mathbf{h}||_2^2 \tag{24}$$

where $\bar{\boldsymbol{\Sigma}}_t = \text{diag}((\mathbf{W}_{t-1}\mathbf{n})^{\circ 2} \circ (\mathbf{P}_{t-1}\mathbf{m}) \circ (\mathbf{1}_{N_{t-1}} - \mathbf{P}_{t-1}\mathbf{m}))$ is the covariance matrix of this adaptive method. We then proceed with an online alternating gradient descent over the filter parameters $\mathbf{h}$, the composite probability parameters $\mathbf{m}$, and the composite weight parameters $\mathbf{n}$ as

$$\mathbf{h}(t) = \prod_{\mathcal{H}} (\mathbf{h}(t-1) - \eta \nabla_h l_t(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t)|_{\mathbf{h}(t-1)}) \tag{25}$$

$$\mathbf{m}(t) = \prod_{\mathcal{S}^M} (\mathbf{m}(t-1) - \eta \nabla_m l_t(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t)|_{\mathbf{m}(t-1)}) \tag{26}$$

$$\mathbf{n}(t) = \prod_{\mathcal{S}^M} (\mathbf{n}(t-1) - \eta \nabla_n l_t(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t)|_{\mathbf{n}(t-1)}) \tag{27}$$

Algorithm 3 Adaptive stochastic online filtering (**Ada-OGF**)

Input: Starting graph $\mathcal{G}_0$, $\mathbf{A}_0$, $\mathbf{x}_0$, $\mathcal{T}$
Initialization: Pre-train $\mathbf{h}^s(0)$, Initialize $\mathbf{m}(0) = \mathbf{1}_M/M$, $\mathbf{n}(0) = \mathbf{1}_M/M$. Compute $\mathbf{P}_0$ and $\mathbf{W}_0$.
for t=1:T **do**
Obtain v_t, $\bar{\mathbf{p}}_t = \mathbf{P}_{t-1}\mathbf{m}(t-1)$, $\bar{\mathbf{w}}_t = \mathbf{W}_{t-1}\mathbf{n}(t-1)$
Prediction:$(\mathbf{W}_{t-1}\mathbf{n}(t-1) \circ \mathbf{P}_{t-1}\mathbf{m}(t-1))^\top \mathbf{A}_{x,t-1}\mathbf{h}^s(t-1)$
Reveal loss $l_t(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t)$
Update $\mathbf{h}(t)$ following (25)
Update $\mathbf{m}(t)$ following (26)
Update $\mathbf{n}(t)$ following (27)
Reveal $\mathbf{a}_t$, update $\mathbf{A}_t$, $\mathbf{x}_t$, $\mathbf{P}_t$, and $\mathbf{W}_t$
end for

where $\Pi_{\mathcal{S}^M}(\cdot)$ is the projection operator onto the probability simplex $\mathcal{S}^M$ and the gradient closed-form expressions are given in Appendix D. After the update, the environment reveals the true attachment $\mathbf{a}_t$ and we update $\mathbf{A}_t$ and $\mathbf{x}_t$. We also update $\mathbf{P}_t$ based on the ensemble of attachment rules applied on the updated topology and the weight dictionary as $\mathbf{W}_t = [\mathbf{W}_{t-1}; \mathbf{e}_t^\top] \in \mathbb{R}^{N_t \times M}$ where $\mathbf{e}_t \in \mathbb{R}^M$ contains independent positive random variables sampled uniformly between zero and the maximum possible edge weight w_h. Algorithm 3 highlights the adaptive stochastic online learning. The computational complexity at time t for Ada-OGF is of order $\mathcal{O}(K(M_0 + N_t) + N_t M)$. See Appendix D for more details.

The loss function in (24) is jointly non-convex in $\mathbf{h}$, $\mathbf{n}$, and $\mathbf{m}$. It is marginally convex in $\mathbf{n}$ and $\mathbf{h}$ but not in $\mathbf{m}$ due to the nature of the covariance matrix. We can run multiple projected descent steps for each of the variables, but proving convergence is non-trivial. However, convergence to a local minimum of $l_t(\cdot)$ may not even be needed as we are in an online non-stationary setting where the arrival of another node leads to a new loss function. Thus, it is reasonable to take one or a few projected steps for each incoming node even without a full convergence guarantee.

Regret analysis: For the regret analysis of the stochastic adaptive online method, we claim the following.

Corollary 2: Given the hypothesis of Theorem 1 and an adaptive stochastic online method over M attachment rules with $\{\mathbf{P}_t\}$, the normalized static regret w.r.t. the deterministic batch learner is upper-bounded as

$$\begin{aligned}\frac{1}{T}R_{s,T}(\mathbf{h}^\star) \leq\ & w_h^2 Y^2 \frac{1}{T}\sum_{t=1}^{T}\left(||\mathbf{P}_{t-1}||_F^2 + M_{max}\right) \\ &+ R w_h Y \frac{1}{T}\sum_{t=1}^{T}||\mathbf{P}_{t-1}||_2^2 + R w_h Y(1 + M_{max}) \\ &+ w_h^2 Y^2 \frac{1}{T}\sum_{t=1}^{T}\bar{P}_t \\ &+ \frac{1}{T}\sum_{t=1}^{T} L_d ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2 \\ &+ \frac{||\mathbf{h}^\star||_2^2}{2\eta T} + \frac{\eta}{2}L_d^2 \end{aligned} \tag{28}$$

Algorithm 4 Prediction Correction Online Graph Filtering (**PC-OGF**)

Input: Graph $\mathcal{G}_0$, $\mathbf{A}_0$, $\mathbf{x}_0$, $\mathcal{T} = \mathcal{T} = \{v_t, x_t, \mathbf{a}_t\}_{t=1:T}$
Initialize: Pre-train $\mathbf{h}^s(0)$ over $\mathcal{G}_0$ using $\mathbf{A}_0$, $\mathbf{x}_0$.
for $t = 1 : T$ **do**
Obtain v_t and $\mathbf{p}_t$, $\mathbf{w}_t$ following preset heuristics
Predict $\hat{x}_t = (\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1}\mathbf{h}^s(t-1)$
Incur loss $l_t^s(\mathbf{h}, x_t)$ [cf. (16)]
Update $\mathbf{h}^s(t)$ using (17)
Reveal $\mathbf{a}_t$, update $\mathbf{A}_t$ and $\mathbf{x}_t$
Update $\mathbf{h}^s(t)$ using (10)
end for

where $\bar{P}_t = \max_{n=1:N_{t-1}} ||[\mathbf{P}_{t-1}]_{n,:}||_2$ and M_{max} is the maximum number of edges formed by each incoming node.

Proof: See Appendix C. □

Compared to the single heuristic attachment model, the regret in (28) depends on the sum of l_2 norm squared of all the M attachment rules. It also depends on $\bar{P}_t$, which is the maximum norm of the vector of probabilities for all rules for each node.

The bound in (28) holds when selecting one attachment rule at each time, i.e., $||\mathbf{m}(t)|| = 1$ for all t. However, a smaller norm of $\mathbf{m}(t)$, corresponding to considering all rules leads to a lower regret bound, potentially improving the performance. Moreover, we expect the term concerning the distance between the stochastic and deterministic filters to reduce due to the adaptive updates, thus, reducing the bound. We shall empirically corroborate this in Section V.

Remark 2: Prediction Correction Online Graph Filtering (**PC-OGF**): In the stochastic algorithms the bounds (20) and (28) show that the regret is influenced by the difference between deterministic filters (that know the attachment) and the stochastic filters (that do not know the attachment) via the term $||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2$. One way to reduce the regret is to leverage the attachments after they are revealed and correct the learned stochastic filter coefficients via a deterministic update. This corresponds to using the prediction correction framework [39]. The prediction step corresponds to performing the filter update based on the predicted output in the absence of connectivity information. The correction step performs an additional update on the prediction step by updating the filter for a loss function with the known attachment. The prediction and correction steps corresponds to one step of S-OGF and D-OGF, respectively. Algorithm 4 highlights this approach. The computational complexity of this at time t is of order $\mathcal{O}(K(M_t + N_t + M_{\max}))$, as it comprises one step of **S-OGF** followed by one of **D-OGF**.

V. Numerical Experiments

We corroborate the proposed methods for regression tasks on both synthetic and real data-sets. We consider the following baselines and state-of-the-art alternatives.

1) **D-OGF** [Alg. 1]: This is the proposed online method for deterministic attachment. We search the filter order $K \in \{1, 3, 5, 7, 9\}$ and the learning rate η and the regularization parameter μ from $[10^{-6}, 1]$.

2) **S-OGF** [Alg. 2]: This is the proposed online method using one stochastic attachment rule. We consider a uniformly at random attachment rule for $\mathbf{p}_t$. For $\mathbf{w}_t$, we use the same weight for each possible edge, which is the median of the edge weights in $\mathcal{G}_{t-1}$. We obtain the regularization parameter μ and step-size η via grid-search over $[10^{-5}, 10^{-1}]$.
3) **Ada-OGF**: [Alg. 3]. This is the proposed adaptive stochastic online method. We take $M = 5$ with attachment rules based on the following node centrality metrics: i) Degree centrality; ii) Betweenness centrality [40]; iii) Eigenvector centrality [41]; iv) Pagerank; v) Uniform.
4) **PC-OGF** [Remark 2]: This is the two-step update method. For the prediction step, we perform S-OGF with uniformly-at-random $\mathbf{p}_t$ and $\mathbf{w}_t$ as considered for S-OGF above. For the correction step, we perform one step of D-OGF. Both steps share the same learning rate $\eta \in [10^{-5}, 10^{-1}]$ and $\mu \in [10^{-5}, 10^{-1}]$.
5) **Batch**: This is the filter designed by taking into account the whole node sequence, i.e.,

$$\mathbf{h}^\star = \underset{\mathbf{h}\in\mathbb{R}^{K+1}}{\operatorname{argmin}} \sum_{t=1}^{T} (\mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)^2 + \mu ||\mathbf{h}||_2^2 \quad (29)$$

which has a closed-form least-squares expression for $\mu > 0 \in [10^{-3}, 10]$.
6) **Pre-trained**: This is a fixed filter trained over the existing graph $\mathcal{G}_0$ and used for the expanding graphs. We train the filter over 80% of the data over $\mathcal{G}_0$. The regularization parameter is chosen over $[10^{-3}, 10]$.
7) **OKL** *Online Multi-Hop Kernel Learning* [18]: We consider a Gaussian kernel with variance $\sigma^2 \in \{0.1, 1, 10\}$. The number of trainable parameters is the same as that of the filters for a fair comparison.
8) **OMHKL** *Online Multi-Hop Kernel Learning* [20]: This method considers multi-hop attachment patterns which are then fed into the random feature framework. We take the multi-hop length as the filter order. We consider one kernel for each hop with the same variance selected from $\sigma^2 \in \{0.1, 1, 10\}$. We did not optimize over the combining coefficients for each multi-hop output. This is to keep the comparisons fair, as OMHKL has more parameters. Instead, we take the mean output, while updating the regression parameter for each multi-hop.

The hyper-parameters are chosen via a validation set. For each parameter, we perform a grid search over a specific range for each data-set, as indicated above for each approach. We use the same filter order as determined for **D-OGF** for the other online filters. We use the same filter order as determined for **D-OGF** for the other online filters. For all data sets, we divide the sequence of incoming nodes into a training and a test sequence. The first 80 percent of the incoming node sequence are taken as the training nodes. The remaining 20 percent are the test nodes. The nodes in the training sequence are used to tune the hyper-parameters, while the test set is used to evaluate the online method for the selected hyper-parameters.

A. Experimental Setup

We consider a synthetic setup based on a random expanding graph model; and two real data setups based on recommender systems and COVID case predictions.

Synthetic: We start with a graph $\mathcal{G}_0$ of $N_0 = 100$ nodes and an edge formation probability of 0.2. The edge weights of $\mathbf{A}_0$ are sampled at random from the uniform distribution between zero and one. Each incoming node v_t forms five uniformly at random edges with the existing graph $\mathcal{G}_{t-1}$. Each newly-formed edge weight is the median of the edge weights in $\mathcal{G}_0$. The existing graph signal $\mathbf{x}_0$ is band-limited w.r.t. the graph Laplacian, making it low-pass over $\mathcal{G}_0$ with a bandwidth of three [14]. We generate the true signal x_t at the incoming v_t in three ways to have three different types of data that fit the different methods.

1) *Filter*: The true signal x_t is generated using a pre-trained filter of order five on $\mathcal{G}_0$. This setting is the closest to the proposed approach and is meant as a sanity check. It also helps us to investigate the differences between the deterministic and the stochastic attachments.
2) *WMean*: x_t is the weighted mean of the signals at the nodes v_t attaches to. This is a neutral setting for all methods.
3) *Kernel*: x_t is obtained from a Gaussian kernel following [18]. This prioritises kernel-based solutions and it is considered here as a controlled setting to compare our method in a non-prioritized setup.

We average the performance of all methods over 10 initial graphs $\mathcal{G}_0$ and each having $T = 1000$ incoming nodes with 800 incoming nodes for training and 200 for testing.

Cold-start recommendation: We consider the Movielens100K data-set that comprises $100,000$ ratings provided by 943 users over 1152 items [42]. We build a 31 nearest neighbour starting graph of 500 random users and consider the remaining 443 users as pure cold starters for the incoming sequence. We use the cosine similarity of the rating vectors to build the adjacency matrix of this graph. We use 50 percent of the ratings of each new user v_t to build $\mathbf{a}_t$. We evaluated all methods over 10 realizations of this setup, where, in each realization, we shuffle the order of incoming users. All methods perform online learning over 16875 and 6155 ratings in the training and test sets, respectively.

COVID case prediction: Here, we predict the number of COVID-19 infection cases for an uninfected city in an existing network of currently infected cities. We consider the data from [43] that has daily case totals for 269 cities and focus on a subset of 302 days of this data-set as in [44]. We randomly select 50 cities and build a five nearest neighbour-directed graph $\mathcal{G}_0$. The edge weight between cities v_i and v_j is $A_{ij} = \exp(-\frac{||\mathbf{t}_i - \mathbf{t}_j||^2}{2\sigma^2})$, where $\mathbf{t}_i$ and $\mathbf{t}_j$ are the vector of COVID cases from day one to 250 for cities v_i and v_j, respectively. We also use this interval to calculate the attachment vector $\mathbf{a}_t$ for any incoming city node. We evaluate the performance on each of the days 255, 260, 265, 270, 275, and 280 and predict the COVID case strength for each

TABLE I
AVERAGE NRMSE AND STANDARD DEVIATION OF ALL APPROACHES FOR ALL DATA-SETS

	Synthetic Data						Real Data			
Method	Filter		WMean		Kernel		Movielens100K		COVID	
	NRMSE	Sdev	NRMSE	Sdev	NRMSE	Sdev	NRMSE	Sdev	NRMSE	Sdev
D-OGF (ours)	0.02	0.003	0.02	0.005	0.25	0.04	0.26	0.01	0.21	0.02
S-OGF (ours)	0.18	0.02	0.26	0.06	0.28	0.07	0.28	0.007	0.31	0.02
Ada-OGF (ours)	0.18	0.02	0.25	0.04	0.28	0.05	0.28	0.007	0.26	0.007
PC-OGF (ours)	0.18	0.02	0.22	0.02	0.23	0.04	0.27	0.01	0.26	0.003
Batch	0.04	0.007	0.09	0.04	1.3	0.29	6.7	0.1	0.17	0.03
Pre-trained	0.08	0.03	0.09	0.03	0.53	0.28	0.84	0.02	2.5	0.9
OKL	0.17	0.01	0.23	0.02	0.25	0.04	0.27	0.01	0.25	0.02
OMHKL	0.17	0.01	0.32	0.1	0.34	0.09	0.27	0.01	0.25	0.02

node city in the sequence. For each day, we carried out twenty realizations where we shuffle at random the order in which the cities are added to the starting graph.

We measure the performance via the root normalized mean square error NRMSE

$$\text{NRMSE} = \frac{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{x}_t - x_t)^2}}{\max\limits_t\,(x_t) - \min\limits_t\,(x_t)}. \tag{30}$$

where $\hat{x}_t$ and x_t are the predicted and true signal at v_t, respectively. This measure gives a more realistic view of the performance since the incoming data does not follow a specific distribution and it is susceptible to outliers [45]. Additionally, we measure the normalized static regret (NReg) [cf. (12)] for the online methods w.r.t. the Batch solution.

B. Performance Comparison

Table I comprises the NRMSEs and the standard deviations for all methods. We observe the following:

Deterministic approaches: D-OGF outperforms OKL and OMHKL across all the data-sets. The difference is more pronounced for the data generated using the *Filter* and the *WMean* method, as they are suited for filters, whereas for the *Kernel* data, the difference is smaller. For the Movielens data-set, the difference is also small. We suspect this is because we train one filter across many graph signals (each graph signal corresponds to a different item) over the same user graph, whereas the kernel method ignores the graph signals. It is possible to improve the prediction accuracy by considering item-specific graphs as showcased in [12], [35]. Next, we observe that D-OGF performs better than pre-trained throughout the experiments. This is because the online filters adapt to the incoming data stream, while the pre-trained does not. The only case we can expect a similar performance is where the incoming data is similar to the data over the existing graph.

Concerning the batch solution, we find that the deterministic online learner outperforms Batch in all data-sets apart from the COVID data-set. This shows the limitations of batch-based solutions, i.e., an over-dependence on the observed training data, and also an inability to adapt to the sequence. For *Filter* and *WMean* data, the training and test set distributions are similar, so the difference between D-OGF and Batch can be attributed to the adaptive nature of D-OGF. In the other data-sets, the change in distribution is detrimental for the batch learner, particularly in the Movielens data.

Stochastic approaches: The S-OGF and Ada-OGF approaches have a similar performance for *Filter*, *Kernel* and Movielens data, with Ada-OGF performing better for *WMean* and Covid data. This makes sense for the synthetic data as the constructed graphs expand following a uniformly at random attachment rule, the same rule used for **S-OGF**. The standard deviation is on the lower side for Ada-OGF. Since the existing signal $\mathbf{x}_0$ is band-limited, the signal values obtained via a filtering/mean operation with a uniformly at random attachment will also be similar. However, Ada-OGF performs better for the COVID data. This is because the incoming data in the COVID data-set is quite different from the synthetic data. It does not have properties like smoothness and thus a uniformly at random attachment cannot help in predicting the number of cases. In such a setting, a more adaptive approach will help. For Movielens data, there is no difference between the two methods, possibly due to the high number of ratings.

Deterministic vs stochastic: The deterministic methods outperform the stochastic counterparts as expected. The gap is closer for the *Kernel* and Movielens data. For Movielens this can be attributed to the subsequent filter updates that are done for different graph signals over a fixed graph. This can cause high prediction errors. The same holds also for the kernel method, as it is based on the same graph. Since the filter takes the signal into account, it might be affected more.

In the Movielens, *Kernel*, and COVID data, the pre-trained filter does not update itself and is thus at a disadvantage, compared to the online methods. For COVID data, the signal over the incoming node, i.e., the number of cases can be quite different from the signals over which the pre-trained filter is learnt, accounting for a higher error. Among the proposed online methods, the stochastic online methods perform poorly w.r.t pre-trained for *Filter* and *WMean* data. This is expected as the data distribution of the incoming data is similar to that over $\mathcal{G}_0$ for these scenarios.

The PC-OGF method performs better than the stochastic methods for all data, showing the added value of correcting for the true attachment. It even outperforms D-OGF for *Kernel* data.

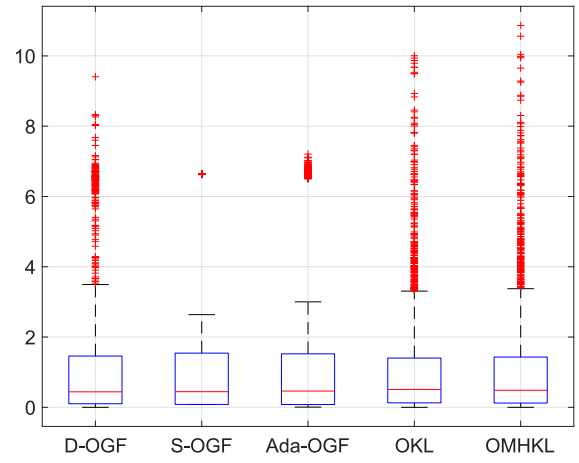

Fig. 2. Box plot of the squared errors for each method in the Movielens data-set.

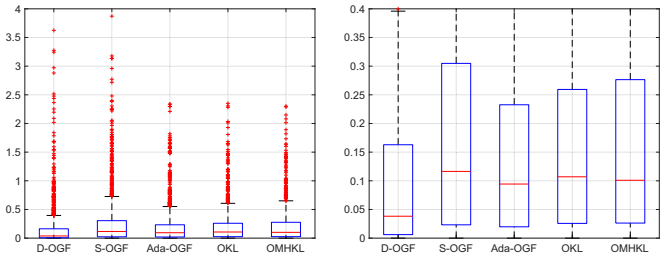

Fig. 3. (Left) Box plot of the squared errors across all data points over the six days. (Right) Box plot on the right zoomed in to highlight differences between all methods.

C. Analysis of Online Methods

We now investigate more in detail the online methods.

Outliers: In Figs. 2 and 3 we show the violin plots of the squared errors over the test set for the Movielens and COVID data. The deterministic methods suffer more from higher outlier errors. This could be attributed to errors in estimating the attachment vectors. For Movielens data, we calculate this similarity over a subset of the items and for certain splits of the items this may lead to estimation errors for the similarity and thus also for the attachment vector. One reason why the stochastic methods are not prone to outliers could be the term in the loss functions [cf. (16), (24)] that penalizes the prediction variance, ultimately, acting as a robust regularizer. Notably, for the Movielens data the errors in the stochastic online learners are fixed at certain levels. This is because the data-set has only five fixed values as ratings and because both S-OGF and Ada-OGF predict fixed values [cf. first term in (16)]. For the Covid data, we calculated the number of outliers in the squared error. The outlier counts are **D-OGF** $= 144$, **S-OGF** $= 123$, **Ada-OGF** $= 119$, **OKL** $= 94$, **OMHKL** $= 98$. This could also be due to the way the starting graph and the links of the incoming nodes are constructed. The figure on the right zooms in on the plot in the range between zero and 0.4. The D-OGF has lower NRMSE, implying the presence of many samples with low squared error. The patterns for the other methods are similar.

Filter order, $\mathbf{p}$, $\mathbf{w}$: Next, we investigate the role of the filter order as well as the impact of training both the attachment probabilities $\mathbf{p}_t$ and weights $\mathbf{w}_t$. Thus, we also want to compare with an alternative adaptive approach where we update only $\mathbf{p}$ while keeping $\mathbf{w}$ fixed to the true edge weights. We call this **Ada2-OGF**. We generate *Filter* data, *WMean* data, and *Kernel* data with a variance of 10. The filter orders evaluated are $K \in \{1, 3, 5, 7, 9\}$. Fig. 4 shows the variation of RNMSE of the filter approaches with filter order K. We see that **Ada2-OGF** performs worse than **Ada-OGF** apart from the *Filter* data. This suggests that updating both $\mathbf{p}$ and $\mathbf{w}$ is beneficial than just updating $\mathbf{p}$. For the filter data, we see that all the three stochastic approaches perform the same. This is because we same stochastic rule, i.e., uniformly at random attachment for data generation. **S-OGF** uses the same, while **Ada-OGF** learns it.

Learning rate: Fig. 6 shows the normalized cumulative regret at each time of **D-OGF** w.r.t. the batch learner for different values of the learning rate η for each synthetic-dataset. Increasing η leads to a lower regret, but after one point, the regret increases. For the *Kernel* data, for example, we see that the regret increases sharply between learning rate $\eta = 3$ and $\eta = 5$. This shows that η indeed influences the online learner and its optimal value is in principle neither too high or too low. A higher value than the optimal misleads the online learner by focusing too much on the current sample. This can lead to high prediction errors for some samples, as seen in the spikes in the plots. A lower value learns about the incoming data-stream at a slower rate.

Regret: Fig. 5 plots the normalized cumulative regret at each time for **S-OGF** and **Ada-OGF** w.r.t. the batch solution for the *Filter* (left), *WMean* (center) and *Kernel* (right) data, respectively. In all three cases, the average cumulative regret converges, implying that the cumulative error or the gap with the batch solution does not diverge. This shows that the stochastic learners, despite not having access to the connectivity at the time of making a prediction, can learn from more incoming nodes. Second, **Ada-OGF** showcases a lower regret than **S-OGF**, showing that it can learn faster from the incoming nodes by trying to predict the attachment behaviour. This is in agreement with the regret bounds in Theorem 1, Corollaries 1 and 2.

Finally, we investigate the normalized regret over the whole sequence for the online methods in Table II. Since we evaluate this over the training set, we have positive values, which implies the batch solution has a lower cumulative error. However, having a positive regret during training can also lead to a lower NRMSE than the batch solution over the test set, as is the case for **D-OGF** [cf. Table I]. This is because the batch filter is fixed and cannot perform as well as in the training set if the distribution of data in the test set is different. The lower regret for **D-OGF** compared to the stochastic approaches stems from the fact that the connectivity is known and because the Batch solution also has a similar loss function. The normalized regret for **PC-OGF** is lesser than that of the stochastic approaches, showing that incorporating the attachment can counter the effect of the gap between the stochastic and deterministic filter.

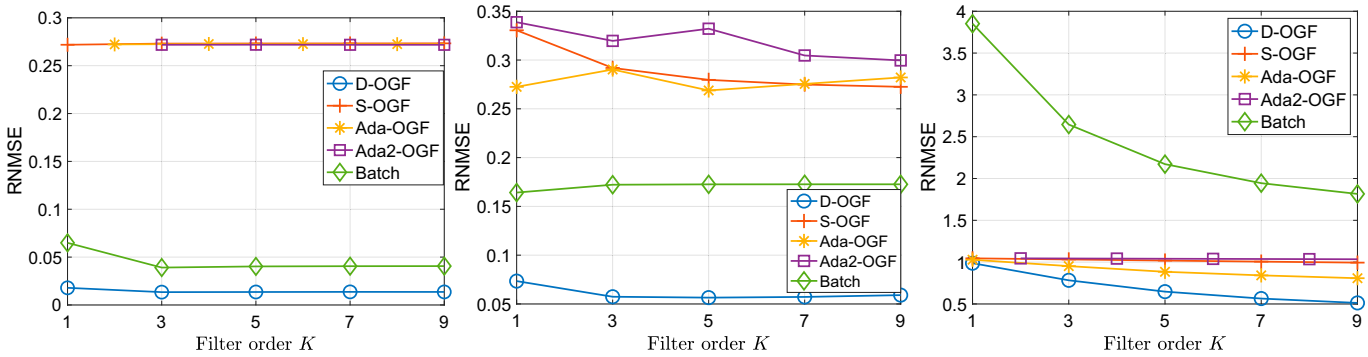

Fig. 4. RNMSE for different values of filter order K for (left) *Filter*, (centre) *WMean*, and (right) *Kernel* data, respectively.

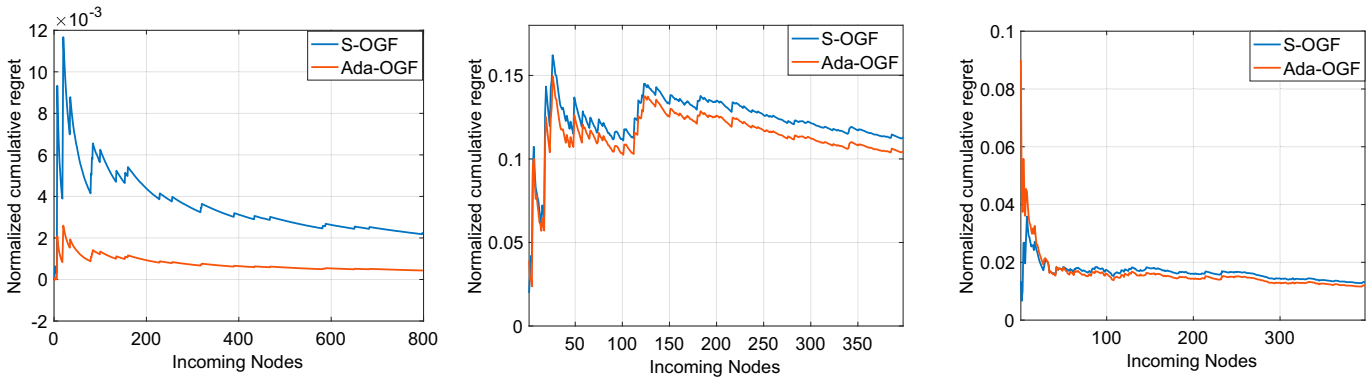

Fig. 5. Evolution of the normalized cumulative regret for S-OGF and Ada-OGF for the synthetic (left) *Filter*, (center) *WMean* and (right) *Kernel* data for $T = 800$, T and 400 incoming nodes, respectively.

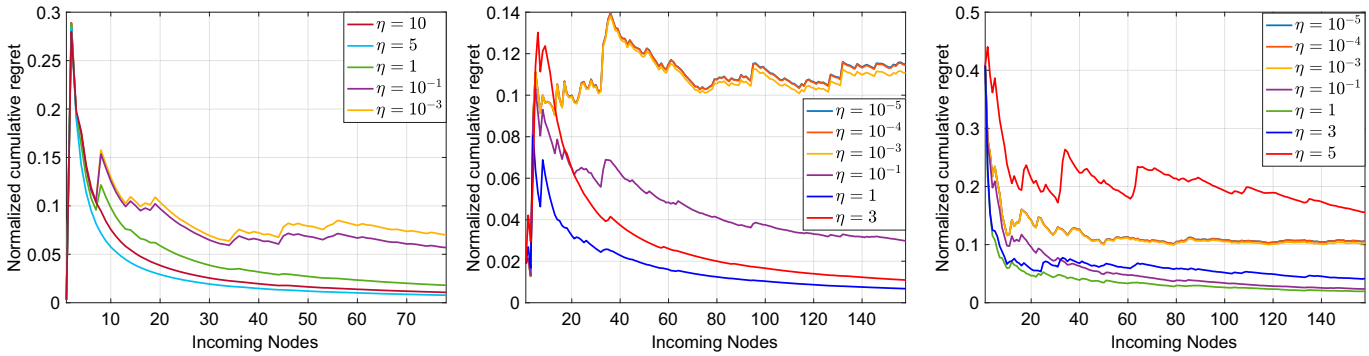

Fig. 6. Normalized cumulative regret evolution for different values of learning rate η for (left) *Filter*, (centre) *WMean*, and (right) *Kernel* data, respectively. The reference average error for the batch solution over the training set are 3×10^{-5}, 6.8×10^{-4}, and 1.1×10^{-2}, respectively.

VI. Conclusion

We proposed online filtering over graphs that grow sequentially over time. We adapted the formulation to the deterministic scenario where the connection of the incoming nodes is known and to a stochastic scenario where this connection is known up to a random model. We performed a simple projected online gradient descent for the online filter update and provided performance bounds in terms of the static regret. In the stochastic setting, the regret is a function of the rule-specific probabilities along with their variance. Numerical results for inference tasks over synthetic and real data show that graph filters trained online learning perform collectively better than kernel methods which do not utilize the data, pre-trained filters, and even a batch filter.

For future work, we will consider the scenario where the signal also varies over the existing graph, i.e., it has a spatio-temporal nature. It is also possible to consider the scenario of joint topology and filter learning over the expanding graphs, where we estimate the true attachment of the incoming node instead of a stochastic model along with the filter used for

TABLE II
NORMALIZED REGRET FOR THE ONLINE METHODS FOR SYNTHETIC DATA

Method	Filter	WMean	Kernel
D-OGF	1.6×10^{-4}	0.03	0.01
S-OGF	2.2×10^{-3}	0.84	0.08
Ada-OGF	4.1×10^{-3}	0.82	0.11
PC-OGF	1.9×10^{-4}	0.27	0.02

making the inference. Finally, to account for the robustness of the online methods, one can also perform a weighted update, where the loss at a particular time is a weighted sum of the previous samples. The complexity of the stochastic approaches grow with the size of the graph. To tackle this, distributed filter updates can be a viable approach.

APPENDIX A
PROOF OF THEOREM 1

The regret relative to the optimal filter $\mathbf{h}^\star$ is

$$R_{s,T}(\mathbf{h}^\star) = \sum_{t=1}^{T} l_t^s(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^\star, x_t). \tag{31}$$

By adding and subtracting the terms $\sum_{t=1}^{T} l_t^d(\mathbf{h}^s(t-1), x_t)$ and $\sum_{t=1}^{T} l_t^d(\mathbf{h}^d(t-1), x_t)$ we obtain

$$\begin{aligned} R_{s,T}(\mathbf{h}^\star) = &\sum_{t=1}^{T} l_t^s(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^s(t-1), x_t) \\ &+ \sum_{t=1}^{T} l_t^d(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^d(t-1), x_t) \\ &+ \sum_{t=1}^{T} l_t(\mathbf{h}^d(t-1), x_t) - l_t^d(\mathbf{h}^\star, x_t). \end{aligned} \tag{32}$$

where $l_t^d(\mathbf{h}^s(t-1), x_t)$ is the deterministic loss at time t evaluated with the filter updated in the stochastic scenario. The regret in (32) comprises three sums over the T-length sequence, each of which contributes to the overall regret.

The first term in (32), $\sum_{t=1}^{T} l_t^s(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^s(t-1), x_t)$ measures the difference in the stochastic and the deterministic loss for the filter updated online in the stochastic setting. We substitute

$$l_t^s(\mathbf{h}^s(t-1), x_t) = ((\mathbf{w}_t \circ \mathbf{p}_t)^\top \bar{\mathbf{y}}_t - x_t)^2 + \bar{\mathbf{y}}_t^\top \boldsymbol{\Sigma}_t \bar{\mathbf{y}}_t \tag{33}$$

$$+ \mu ||\mathbf{h}^s(t-1)||_2^2 \tag{34}$$

where $\bar{\mathbf{y}}_t = \mathbf{A}_{x,t-1}\mathbf{h}^s(t-1)$ and

$$l_t^d(\mathbf{h}^s(t-1), x_t) = (\mathbf{a}_t^\top \bar{\mathbf{y}}_t - x_t)^2 + \mu ||\mathbf{h}^s(t-1)||_2^2 \tag{35}$$

to get the difference at time t

$$\begin{aligned} &l_t^s(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^s(t-1), x_t) \\ &\quad = ((\mathbf{w}_t \circ \mathbf{p}_t)^\top \bar{\mathbf{y}}_t - x_t)^2 - (\mathbf{a}_t^\top \bar{\mathbf{y}}_t - x_t)^2 + \bar{\mathbf{y}}_t^\top \boldsymbol{\Sigma}_t \bar{\mathbf{y}}_t. \end{aligned} \tag{36}$$

After some simplification, we get

$$\begin{aligned} &l_t^s(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^s(t-1), x_t) \\ &\quad = ((\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t)^\top \bar{\mathbf{y}}_t)^2 \\ &\qquad + 2(\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t)^\top \bar{\mathbf{y}}_t (\mathbf{a}_t^\top \bar{\mathbf{y}}_t - x_t) + \bar{\mathbf{y}}_t^\top \boldsymbol{\Sigma}_t \bar{\mathbf{y}}_t. \end{aligned} \tag{37}$$

The r.h.s. of equation (37) has three terms. For the first term we have

$$\begin{aligned} ((\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t)^\top \bar{\mathbf{y}}_t)^2 &\le ||\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t||_2^2 ||\bar{\mathbf{y}}_t||_2^2 \\ &\le w_h^2 (||\mathbf{p}_t||_2^2 + M_{max}) Y^2 \end{aligned} \tag{38}$$

where the first inequality follows from the Cauchy-Schwartz inequality and the second inequality from Lemmas 2 and Lemma 3 in Appendix D.

For the second term we have

$$\begin{aligned} &2(\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t)^\top \bar{\mathbf{y}}_t (\mathbf{a}_t^\top \bar{\mathbf{y}}_t - x_t) \\ &\quad \le 2||\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t||_2 ||\bar{\mathbf{y}}_t||_2 ||\mathbf{a}_t^\top \bar{\mathbf{y}}_t - x_t||_2 \\ &\quad \le 2R w_h Y \sqrt{||\mathbf{p}_t||_2^2 + M_{max}} \end{aligned} \tag{39}$$

where the first inequality follows from the Cauchy-Schwartz inequality and the second inequality from Assumption 4, Lemmas 3 and 2. For the third term, we have

$$\begin{aligned} \bar{\mathbf{y}}_t^\top \boldsymbol{\Sigma}_t \bar{\mathbf{y}}_t &= \sum_{n=1}^{N_{t-1}} [\bar{\mathbf{y}}_t]_n^2 [\mathbf{w}_t]_n^2 [\mathbf{p}_t]_n (1 - [\mathbf{p}_t]_n) \\ &\le w_h^2 \bar{\sigma}_t^2 Y^2 \end{aligned} \tag{40}$$

where the inequality follows the definition of $\bar{\sigma}_t^2$ and Lemma 3. Adding (38)-(40) we can upper-bound (37) as

$$\begin{aligned} &\sum_{t=1}^{T} l_t^s(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^s(t-1), x_t) \\ &\quad \le w_h^2 Y^2 (||\mathbf{p}_t||_2^2 + M_{max}) + 2R w_h Y \sqrt{||\mathbf{p}_t||_2^2 + M_{max}} \\ &\quad + w_h^2 \bar{\sigma}_t^2 Y^2. \end{aligned} \tag{41}$$

The second term in (32), $\sum_{t=1}^{T} l_t^d(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^d(t-1), x_t)$ measures the sum of the differences in the deterministic loss between the deterministic and stochastic online filter. Since $l_t^d(\cdot,\cdot)$ is Lipschitz with constant L_d from Lemma 1, we can write

$$\begin{aligned} &|l_t^d(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^d(t-1), x_t)| \\ &\quad \le L_d ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2 \end{aligned} \tag{42}$$

which implies $l_t(\mathbf{h}^s(t-1), x_t) - l_t(\mathbf{h}^d(t-1), x_t) \le L_d ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2$. Summing over t, we have

$$\begin{aligned} &\sum_{t=1}^{T} l_t^d(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^d(t-1), x_t) \\ &\quad \le L_d \sum_{t=1}^{T} ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2. \end{aligned} \tag{43}$$

The third term in (32), $\sum_{t=1}^{T} l_t^d(\mathbf{h}^d(t-1), x_t) - l_t^d(\mathbf{h}^\star, x_t)$ corresponds to the static regret in the deterministic case and has the upper bound

$$\sum_{t=1}^{T} \left(l_t(\mathbf{h}^d(t-1), x_t) - l_t^d(\mathbf{h}^\star, x_t)\right) \leq \frac{||\mathbf{h}^\star||_2^2}{2\eta} + \frac{\eta}{2} L_d^2 T \quad (44)$$

as shown in Proposition 1.

By summing equations (41), (43), and (44), we obtain

$$R_{s,T}(\mathbf{h}^\star) \leq \sum_{t=1}^{T} w_h^2 Y^2 (||\mathbf{p}_t||_2^2 + M_{max}) + 2Rw_h Y \sqrt{||\mathbf{p}_t||_2^2 + M_{max}} + w_h^2 \bar{\sigma}_t^2 Y^2 + L_d ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)|| \Big) + \frac{||\mathbf{h}^\star||_2^2}{2\eta} + \frac{\eta}{2} L_d^2 T \quad (45)$$

Finally, dividing both sides by T and using Lemma 3, we complete the proof. □

Appendix B
Proof of Corollary 1

We substitute $\mathbf{p}_t = \frac{1}{N_{t-1}} \mathbf{1}_{N_{t-1}}$, in each term of the stochastic regret bound. For the first term we have

$$\sum_{t=1}^{T} w_h^2 Y^2 (||\mathbf{p}_t||_2^2 + M_{max}) \leq w_h^2 Y^2 \sum_{t=1}^{T} \frac{1}{N_{t-1}} + w_h^2 M_{max} Y^2 T \quad (46)$$

Substituting $N_t = N_0 + t - 1$, we have

$$\sum_{t=1}^{T} w_h^2 Y^2 (||\mathbf{p}_t||_2^2 + M_{max}) \quad (47)$$

$$\leq w_h^2 Y \sum_{t=1}^{T} \frac{1}{N_0 + t - 1} + w_h^2 M_{max} Y^2 T \quad (48)$$

Next, we bound the summation $\sum_{t=1}^{T} \frac{1}{N_0+t-1}$ as[3]

$$\sum_{t=1}^{T} \frac{1}{N_0 + t - 1} \leq \int_{t=1}^{T} \frac{1}{t + N_0 - 1} dt \quad (49)$$

which gives us

$$\sum_{t=1}^{T} \frac{1}{N_0 + t - 1} \leq \log(T + N_0 - 1) - \log(N_0) \quad (50)$$

Substituting (50) in (47), we have

$$\sum_{t=1}^{T} w_h^2 Y^2 (||\mathbf{p}_t||_2^2 + M_{max}) \quad (51)$$

$$\leq w_h^2 Y^2 (\log(T + N_0 - 1) - \log(N_0)) + w_h^2 M_{max} Y^2 T \quad (52)$$

[3]For a function $f(t) > 0$, we have $\sum_{t=1}^{T} f(t)dt \leq \int_{t=1}^{T} f(t)dt$.

On dividing by T and taking its limit to infinite, we have

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} w_h^2 Y^2 (||\mathbf{p}_t||_2^2 + M_{max}) \leq w_h^2 M_{max} Y^2 \quad (53)$$

where the first term vanishes as T grows faster than $\log(T)$.

For the second term we have

$$\sum_{t=1}^{T} 2Rw_h Y \sqrt{||\mathbf{p}_t||_2^2 + M_{max}} \quad (54)$$

$$\leq 2Rw_h Y \sum_{t=1}^{T} \frac{1}{2} (||\mathbf{p}_t||_2^2 + M_{max} + 1) \quad (55)$$

$$\leq Rw_h Y \left(\sum_{t=1}^{T} ||\mathbf{p}_t||_2^2 + T(M_{max} + 1) \right) \quad (56)$$

where we use the fact that the geometric mean is lesser than or equal to the arithmetic mean. Utilizing the fact that $\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} ||\mathbf{p}_t||_2^2$ is equal to zero (as shown above in (53)), we have

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} 2Rw_h Y \sqrt{||\mathbf{p}_t||_2^2 + M_{max}} \leq Rw_h Y (M_{max} + 1) \quad (57)$$

For the third term $w_h^2 Y^2 \bar{\sigma}_t^2$, we have

$$\sum_{t=1}^{T} w_h^2 \bar{\sigma}_t^2 ||\bar{\mathbf{y}}_t||_2^2 \leq w_h^2 Y \sum_{t=1}^{T} \frac{1}{N_{t-1}} \left(1 - \frac{1}{N_{t-1}}\right) \leq w_h^2 Y \sum_{t=1}^{T} \frac{1}{N_{t-1}} \quad (58)$$

which holds for the uniformly at random attachment rule. Given $\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N_{t-1}} = 0$, the third term vanishes in the limit. Adding (53) and (57), along with the other terms from the stochastic regret bound, we have the required bound for Corollary 1. □

Appendix C
Proof of Corollary 2

To prove this corollary, we start with the result of Proposition 1. For the first term, we have $||\mathbf{p}_t||_2^2 = ||\mathbf{P}_{t-1}\mathbf{m}(t-1)||_2^2$, which is upper bounded as $||\mathbf{m}(t-1)||_2^2 ||\mathbf{P}_{t-1}||_F^2$. The maximum value of $||\mathbf{m}(t-1)||_2^2$ is one for $\mathbf{m}(t-1) \in \mathcal{S}_M$.

For the second term, we use the Arithmetic Mean-Geometric Mean inequality as in Corollary 1 and use again $||\mathbf{p}_t||_2^2 \leq ||\mathbf{P}_{t-1}||_F^2$.

Similarly, for the third term we have

$$\begin{aligned} \bar{\sigma}_t^2 &= \max_{n=1:N_{t-1}} [\mathbf{p}_t]_n (1 - [\mathbf{p}_t]_n) \\ &= \max_{n=1:N_{t-1}} [\mathbf{P}_{t-1}\mathbf{m}(t-1)]_n (1 - [\mathbf{P}_{t-1}\mathbf{m}(t-1)]_n) \\ &\leq \max_{n=1:N_{t-1}} [\mathbf{P}_{t-1}\mathbf{m}(t-1)]_n \leq \bar{P}_t ||\mathbf{m}(t-1)||_2 \leq P_t \end{aligned} \quad (59)$$

Substituting $\bar{\sigma}_t^2$ in the regret for the stochastic setting, we complete the proof. □

APPENDIX D
RELEVANT DERIVATIONS

Lemma 1: Under Assumption 1 and 3, the loss function $l_t(\mathbf{h}, x_t)$ is Lipschitz in $\mathbf{h}$. That is, the l_2 norm of the gradient of the loss at time t is upper-bounded as

$$||\nabla_h l_t(\mathbf{h}, x_t)||_2 \leq L_d \tag{60}$$

where $L_d = (RC + 2\mu H)$

Proof: We apply the Cauchy-Schawrtz inequality on the r.h.s. of (11) and get

$$\begin{aligned}||\nabla_h l_t(\mathbf{h}, x_t)||_2 &\leq |\mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t||\mathbf{A}_{x,t-1}^\top \mathbf{a}_t||_2 + 2\mu||\mathbf{h}||_2 \\ &\leq RC + 2\mu H. \end{aligned} \tag{61}$$

where we use Assumption 1. □

Lemma 2: At time t, given the probability of attachment $\mathbf{p}_t$, the weight $\mathbf{w}_t$, and the attachment $\mathbf{a}_t$. Let N_{t-1} be the number of existing nodes. We have

$$||\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t||_2^2 \leq w_h^2(||\mathbf{p}_t||_2^2 + M_{max}). \tag{62}$$

Proof: We can write the squared norm as

$$||\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t||_2^2 = \sum_{n=1}^{N_{t-1}} [\mathbf{w}_t]_n^2[\mathbf{p}_t]_n^2 - 2[\mathbf{w}_t]_n[\mathbf{p}_t]_n[\mathbf{a}_t]_n + [\mathbf{a}_t]_n^2 \tag{63}$$

The second term in this summation is always negative, so we can write

$$||\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t||_2^2 \leq \sum_{n=1}^{N_{t-1}} [\mathbf{w}_t]_n^2[\mathbf{p}_t]_n^2 + [\mathbf{a}_t]_n^2 \tag{64}$$

Note that $\sum_{n=1}^{N_{t-1}} [\mathbf{a}_t]_n^2 \leq M_{max} w_h^2$ using Assumptions 1 and 2; thus, we have

$$||\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t||_2^2 \leq w_h^2||\mathbf{p}_t||_2^2 + M_{max} w_h^2 \tag{65}$$

□

Lemma 3: The term $||\mathbf{A}_{x,t-1}\mathbf{h}||_2$ is bounded in its ℓ_2 norm for all t, i.e., $||\mathbf{A}_{x,t-1}\mathbf{h}||_2 \leq Y$

Proof: From the expression of $\tilde{\mathbf{y}}_t$ in (5), we have

$$||\tilde{\mathbf{y}}_t||_2 \leq ||\sum_{k=0}^{K} h_k \mathbf{A}_{t-1}^k \mathbf{x}_t||_2 + ||\mathbf{a}_t^\top \sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1}\mathbf{x}_t||_2 \tag{66}$$

$$\leq ||\sum_{k=0}^{K} h_k \mathbf{A}_{t-1}^k \mathbf{x}_t||_2 + ||\mathbf{a}_t||_2||\sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1}\mathbf{x}_t||_2 \tag{67}$$

The first term on the R.H.S. is bounded for a bounded $\mathbf{h}$ and $\mathbf{A}_{t-1}$. So is the second term, following Assumptions 1 and 2. Thus both the output $\tilde{\mathbf{y}}_t$ and $\sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1}\mathbf{x}_t$ are bounded. We denote the bound for $||\sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1}\mathbf{x}_t||_2$ as Y. □

Gradients: Here we provide the expressions for the gradients w.r.t $\mathbf{h}$, $\mathbf{m}$, and $\mathbf{n}$ for the adaptive stochastic online learner.

$$\begin{aligned}\nabla_h l_t^s(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t) = &((\mathbf{W}_{t-1}\mathbf{n} \circ \mathbf{P}_{t-1}\mathbf{m})^\top \\ &\mathbf{A}_{x,t-1}\mathbf{h} - x_t)\mathbf{A}_{x,t-1}^\top(\mathbf{w}_t \circ \mathbf{p}_t) \\ &+ \mathbf{A}_{x,t-1}^\top \bar{\mathbf{\Sigma}}_t \mathbf{A}_{x,t-1}\mathbf{h} + 2\mu\mathbf{h}. \end{aligned} \tag{68}$$

The gradient w.r.t. $\mathbf{m}$ is

$$\begin{aligned}&\nabla_m l_t(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t) \\ &= \mathbf{P}_{t-1}^\top(\mathbf{W}_{t-1}\mathbf{n} \circ \mathbf{A}_{x,t-1}\mathbf{h})(\mathbf{W}_{t-1}\mathbf{n} \circ \mathbf{P}_{t-1}\mathbf{m})^\top \\ &\quad \times \mathbf{A}_{x,t-1}\mathbf{h} - x_t) + \mathbf{P}_{t-1}^\top((\mathbf{A}_{x,t-1}\mathbf{h})^{\circ 2} \circ (\mathbf{W}_{t-1}\mathbf{n})^{\circ 2}) \\ &\quad - 2\mathbf{P}_{t-1}^\top(\mathbf{P}_{t-1}\mathbf{m} \circ (\mathbf{A}_{x,t-1}\mathbf{h})^{\circ 2} \circ (\mathbf{W}_{t-1}\mathbf{n})^{\circ 2}). \end{aligned} \tag{69}$$

Finally, the gradient w.r.t $\mathbf{n}$ is

$$\begin{aligned}&\nabla_n l_t(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t) \\ &= \mathbf{W}_t^\top(\mathbf{P}_{t-1}\mathbf{m} \circ \mathbf{A}_{x,t-1}\mathbf{h})((\mathbf{W}_{t-1}\mathbf{n} \circ \mathbf{P}_{t-1}\mathbf{m})^\top \\ &\quad \mathbf{A}_{x,t-1}\mathbf{h} - x_t) \end{aligned} \tag{70}$$

□

Computational complexity: Here we provide the computational complexity of the online learners. All methods rely on computing $\mathbf{A}_{x,t-1} = [\tilde{\mathbf{x}}_t, \mathbf{A}_{t-1}\tilde{\mathbf{x}}_t, \ldots, \mathbf{A}_{t-1}^{K-1}\tilde{\mathbf{x}}_t]$ at time t. We construct $\mathbf{A}_{x,t-1}$ with a complexity of order $\mathcal{O}(M_{t-1}K)$ formed by shifting $\tilde{\mathbf{x}}_t$ $K-1$ times over $\mathcal{G}_{t-1}$. At time $t-1$, we need not calculate $\mathbf{A}_{x,t} = [\tilde{\mathbf{x}}_{t+1}, \mathbf{A}_t\tilde{\mathbf{x}}_{t+1}, \ldots, \mathbf{A}_t^{K-1}\tilde{\mathbf{x}}_{t+1}]$ from scratch. From the structure of $\mathbf{A}_t$ [cf. (1)], we only need to calculate the diffusions over the incoming edges $K-1$ times, which amounts to an additional complexity of $\mathcal{O}(M_{\max}K)$. The computational complexity of each online method is detailed next.

D-OGF: The complexity of update (10) is governed by the gradient, which depends on the output at time $\hat{x}_t = \mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h}$. It has a complexity of $\mathcal{O}(M_{\max}K + M_{t-1}K)$, where the complexity for $\mathbf{A}_{x,t-1}\mathbf{h}$ is $M_{t-1}K$ and that for the diffusion over the newly formed edges is $\mathcal{O}(M_{\max}K)$.

S-OGF: The only difference between S-OGF and D-OGF is that we use the expected attachment vector $(\mathbf{w}_t \circ \mathbf{p}_t)$ to calculate the output, which means the complexity incurred depends on N_{t-1} for all t.

Ada-OGF: The Ada-OGF, being a stochastic approach has already a complexity of $\mathcal{O}(K(M_0 + N_t))$. However, it has an extra complexity of $\mathcal{O}(N_t(M))$ due to both the $\mathbf{m}$ and $\mathbf{n}$ update.

REFERENCES

[1] B. Das and E. Isufi, "Online filtering over expanding graphs," in *Proc. IEEE Asilomar Conf. Signals, Syst. Comput.,* Pacific Grove, USA, 2022.

[2] X. Dong, D. Thanou, L. Toni, M. Bronstein, and P. Frossard, "Graph signal processing for machine learning: A review and new perspectives," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 117–127, Nov. 2020.

[3] D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis, "Adaptive diffusions for scalable learning over graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 5, pp. 1307–1321, Mar. 2019.

[4] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[5] E. Isufi, M. Pocchiari, and A. Hanjalic, "Accuracy-diversity trade-off in recommender systems via graph convolutions," *Inf. Process. & Manage.*, vol. 58, no. 2, 2021, Art. no. 102459.

[6] E. Isufi, F. Gama, D. I. Shuman, and S. Segarra, "Graph filters for signal processing and machine learning on graphs," *IEEE Trans. Signal Process.*, early access, Jan. 10, 2024, doi: 10.1109/TSP.2024.3349788.

[7] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, Jun. 2014.

[8] D. Romero, M. Ma, and G. B. Giannakis, "Kernel-based reconstruction of graph signals," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 764–778, Feb. 2017.

[9] A. L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, Oct. 1999, AAAS.
[10] A.-L. Barabási et al., *Network Science*. Cambridge, U.K.: Cambridge Univ. Press, 2016.
[11] Z. Wang, Y. Tan, and M. Zhang, "Graph-based recommendation on social networks," in *Proc. 12th Int. Asia-Pacific Web Conf.,* Piscataway, NJ, USA: IEEE Press, 2010, pp. 116–122.
[12] W. Huang, A. G. Marques, and A. R. Ribeiro, "Rating prediction via graph signal processing," *IEEE Trans. Signal Process.*, vol. 66, no. 19, pp. 5066–5081, Oct. 2018.
[13] N. Silva, D. Carvalho, A. C. Pereira, F. Mourão, and L. Rocha, "The pure cold-start problem: A deep study about how to conquer first-time users in recommendations domains," *Inf. Syst.*, vol. 80, pp. 1–12, 2019.
[14] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, appliications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.
[15] P. Erdos, "On the evolution of random graphs," *Bull. Inst. Int. Statist.*, vol. 38, pp. 343–347, 1961.
[16] F. Orabona, "A modern introduction to online learning," 2019, *arXiv:1912.13213*.
[17] E. Hazan et al., "Introduction to online convex optimization," *Found. Trends® Optim.*, vol. 2, nos. 3-4, pp. 157–325, 2016.
[18] Y. Shen, G. Leus, and G. B. Giannakis, "Online graph-adaptive learning with scalability and privacy," *IEEE Trans. Signal Process.*, vol. 67, no. 9, pp. 2471–2483, May 2019.
[19] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 20, 2007.
[20] Z. Zong and Y. Shen, "Online multi-hop information based kernel learning over graphs," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP),* Piscataway, NJ, USA: IEEE Press, 2021, pp. 2980–2984.
[21] R. Money, J. Krishnan, and B. Beferull-Lozano, "Online non-linear topology identification from graph-connected time series," in *Proc. IEEE Data Sci. Learn. Workshop (DSLW),* Piscataway, NJ, USA: IEEE Press, 2021, pp. 1–6.
[22] R. Money, J. Krishnan, and B. Beferull-Lozano, "Sparse online learning with kernels using random features for estimating nonlinear dynamic graphs," *IEEE Trans. Signal Process.*, vol. 71, pp. 2027–2042, 2023.
[23] R. Shafipour and G. Mateos, "Online topology inference from streaming stationary graph signals with partial connectivity information," *Algorithms*, vol. 13, no. 9, 2020, Art. no. 228.
[24] A. Venkitaraman, S. Chatterjee, and B. Wahlberg, "Recursive Prediction of graph signals with incoming nodes," in *Proc. Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 5565–5569.
[25] S. Chen, F. Cerda, P. Rizzo, J. Bielak, J. H. Garrett, and J. Kovačević, "Semi-supervised multiresolution classification using adaptive graph filtering with application to indirect bridge structural health monitoring," *IEEE Trans. Signal Process.*, vol. 62, no. 11, pp. 2879–2893, Jun. 2014.
[26] F. Dornaika, R. Dahbi, A. Bosaghzadeh, and Y. Ruichek, "Efficient dynamic graph construction for inductive semi-supervised learning," *Neural Netw.*, vol. 94, pp. 192–203, 2017.
[27] L. Jian, J. Li, and H. Liu, "Toward online node classification on streaming networks," *Data Mining Knowl. Discovery*, vol. 32, no. 1, pp. 231–257, 2018.
[28] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "A graph diffusion lms strategy for adaptive graph signal processing," in *Proc. 51st Asilomar Conf. Signals, Syst., Comput.,* Piscataway, NJ, USA: IEEE Press, 2017, pp. 1973–1976.
[29] F. Hua, R. Nassif, C. Richard, H. Wang, and A. H. Sayed, "Online distributed learning over graphs with multitask graph-filter models," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 6, pp. 63–77, 2020.
[30] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "Distributed diffusion adaptation over graph signals," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP),* Piscataway, NJ, USA: IEEE Press, 2018, pp. 4129–4133.
[31] V. R. Elias, V. C. Gogineni, W. A. Martins, and S. Werner, "Adaptive graph filters in reproducing kernel hilbert spaces: Design and performance analysis," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 7, pp. 62–74, 2020.
[32] B. Das and E. Isufi, "Graph filtering over expanding graphs," in *Proc. IEEE Data Sci. Learn. Workshop Process. (DSLW)*, May 2022, pp. 1–8.
[33] X. Liu, P. C. Hsieh, N. Duffield, R. Chen, M. Xie, and X. Wen, "Streaming network embedding through local actions," 2018, *arXiv:1811.05932*.
[34] B. Das and E. Isufi, "Learning expanding graphs for signal interpolation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP),* Piscataway, NJ, USA: IEEE Press, 2022, pp. 5917–5921.
[35] B. Das, A. Hanjalic, and E. Isufi, "Task-aware connectivity learning for incoming nodes over growing graphs," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 8, pp. 894–906, 2022.
[36] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf.*, 2002, pp. 253–260.
[37] S. Vlaski, S. Kar, A. H. Sayed, and J. M. Moura, "Networked signal and information processing: Learning by multiagent systems," *IEEE Signal Process. Mag.*, vol. 40, no. 5, pp. 92–105, Jul. 2023.
[38] S. Shalev-Shwartz et al., "Online learning and online convex optimization," *Found. Trends® Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2012.
[39] A. Simonetto and E. Dall'Anese, "Prediction-correction algorithms for time-varying constrained optimization," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5481–5494, Oct. 2017.
[40] M. E. Newman, "A measure of betweenness centrality based on random walks," *Social Netw.*, vol. 27, no. 1, pp. 39–54, 2005.
[41] B. Ruhnau, "Eigenvector-centrality—A node-centrality?" *Social Netw.*, vol. 22, no. 4, pp. 357–365, 2000.
[42] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, 2015.
[43] E. Dong, H. Du, and L. Gardner, "An interactive web-based dashboard to track COVID-19 in real time," *Lancet Infect. Dis.*, vol. 20, no. 5, pp. 533–534, 2020.
[44] J. H. Giraldo, A. Mahmood, B. Garcia-Garcia, D. Thanou, and T. Bouwmans, "Reconstruction of time-varying graph signals via sobolev smoothness," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 8, pp. 201–214, 2022.
[45] M. V. Shcherbakov et al., "A survey of forecast error measures," *World Appl. Sci. J.*, vol. 24, no. 24, pp. 171–176, 2013.