Forecasting Graph Signals with Recursive MIMO Graph Filters

van der Hoeven, Jelmer; Natali, Alberto; Leus, Geert

DOI
10.23919/EUSIPCO58844.2023.10289997

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 31st European Signal Processing Conference (EUSIPCO)

Forecasting Graph Signals with Recursive MIMO Graph Filters

Jelmer van der Hoeven
Faculty of EEMCS
Delft University of Technology
Delft, The Netherlands
Jelmer.van.der.hoeven@pwc.com

Alberto Natali
Faculty of EEMCS
Delft University of Technology
Delft, The Netherlands
a.natali@tudelft.nl

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

***Abstract*—Forecasting time series on graphs is a fundamental problem in graph signal processing. When each entity of the network carries a vector of values for each time stamp instead of a scalar one, existing approaches resort to the use of product graphs to combine this multidimensional information, at the expense of creating a larger graph. In this paper, we show the limitations of such approaches, and propose extensions to tackle them. Then, we propose a recursive multiple-input multiple-output graph filter which encompasses many already existing models in the literature while being more flexible. Numerical simulations on a real world data set show the effectiveness of the proposed models.**

***Index Terms*—Forecasting, Graph Signal Processing, Product Graph, Multi-dimensional graph signals**

I. Introduction

Forecasting time series collected by entities of a network is a central problem in graph signal processing (GSP) [1], finding applications in sensor [2] and social networks [3], [4], to name a few. Accounting for the network structure in the forecasting model serves as an inductive bias to reduce the number of estimation parameters of the model [5], [6] compared to classical vector autoregressive (VAR) models [7]. However, existing models under a GSP lens consider a scalar value at each node for each time instant [6], [8], [9], which is a limitation in networks where a (feature) vector of measurements is available at each node for each time instant, such as in meteorological sensor networks or 3D point clouds.

This *multidimensional* case has been recently addressed in [10] with the introduction of a so-called feature graph, capturing the possible relationships among the features of a node of the network. There, the authors make use of product graphs [11] to create a new larger graph that combines in a principled way the original and the feature graphs, which is then used to forecast these multidimensional content by taking into account relationships among features of different nodes. However, the knowledge of which product graph to use and how to construct the feature graph has not been properly elaborated.

This work can be considered an extension of [10] in that: *i)* we first delineate drawbacks of the proposed model and introduce minor extensions to tackle them; then, *ii)* we propose a recursive multiple-input multiple-output (MIMO) graph filter which generalizes the already existing filters while being more flexible; finally, *iii)* we put forth a method which learns the weights of the feature graph during the training process. We limit our exposition to linear models, although non-linear autoregressive models are a subject worth investigating. Numerical experiments on real-world data show the effectiveness of the proposed approaches.

II. Background

Consider an N-dimensional time series $\mathbf{x}_t \in^N$, where entry $x_t(i)$ represents the value at time t associated to the entry i. For instance, $x_t(i)$ might represent the amount of water in the ith reservoir of a water network at time t. When the relationships between the different time series of $\mathbf{x}_t$ can be captured by a network, let the graph $\mathcal{G} := (\mathcal{V}, \mathcal{E}, \mathbf{S})$ model such network, where $\mathcal{V} = \{v_1, \dots, v_N\}$ is the set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{S} \in \mathbb{R}^{N \times N}$ is the matrix representation of the graph, which captures its sparsity pattern; i.e., $S_{ij} \neq 0$ if and only if $(v_i, v_j) \in \mathcal{E}$ or $v_i = v_j$. We refer to matrix $\mathbf{S}$ as the graph shift operator (GSO), examples of which include the adjacency matrix and the Laplacian matrix [12], [13]. In this context, $\mathbf{x}_t$ is called a graph signal.

We can instantly process a graph signal $\mathbf{x}_t$ to obtain a new graph signal $\mathbf{y}_t$ by means of the so-called graph convolution [1]:

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

where $\mathbf{H}(\mathbf{S}) := \sum_{k=0}^{K-1} h_k \mathbf{S}^k$ is the graph filter [14] of order $K-1$ with scalar coefficients $h_0, \dots, h_{K-1}$. Because matrix $\mathbf{S}$ is sparse, the computational complexity of the graph filtering operation (1) is $\mathcal{O}(|\mathcal{E}|K)$. To forecast time series residing on the nodes, a graph vector autoregressive (G-VAR) model has been introduced in [6] as the combination of a finite number of graph convolutions; specifically:

$$\mathbf{x}_t = -\sum_{p=1}^{P} \mathbf{H}_p(\mathbf{S}) \mathbf{x}_{t-p} = -\sum_{p=1}^{P} \sum_{k=0}^{K-1} h_{kp} \mathbf{S}^k \mathbf{x}_{t-p}, \tag{2}$$

where $\mathbf{x}_{t-p}$ represents the graph signal at time instant $t-p$ and h_{kp} represents the kth scalar filter coefficient of the p-th graph filter $\mathbf{H}_p(\mathbf{S})$. The computational complexity of (2) is $\mathcal{O}(PK|\mathcal{E}|)$ and the number of parameters of the filter is PK, both independent of the size of the network.

Multidimensional. When each node of the network carries a vector of F values (features) for each time instant t, we refer to such a graph signal as being F-dimensional, and denote it as $\mathbf{x}_t = [\mathbf{x}_t^\top(1), \dots, \mathbf{x}_t^\top(F)]^\top \in \mathbb{R}^{NF}$, where each $\mathbf{x}_t(f) \in \mathbb{R}^N$ is the one-dimensional graph signal associated to feature f. In other words, $\mathbf{x}_t$ is the concatenation of F graph signals, each one representing the graph signal associated to one specific feature.

To efficiently deal with such multi-dimensional graph signals, the work in [10] proposes to model the dependencies among the features with a so-called *feature graph* defined as $\mathcal{G}_\mathcal{F} := (\mathcal{V}_\mathcal{F}, \mathcal{E}_\mathcal{F}, \mathbf{S}_\mathcal{F})$, where $\mathcal{V}_\mathcal{F} = \{f_1, ..., f_F\}$ is the set of F nodes representing the features, $\mathcal{E}_\mathcal{F} \subseteq \mathcal{V}_\mathcal{F} \times \mathcal{V}_\mathcal{F}$ is the set of edges defining how the features are connected, and $\mathbf{S}_\mathcal{F}$ is the associated GSO. To formally capture the dependencies among features of different nodes, $\mathcal{G}$ and $\mathcal{G}_\mathcal{F}$ can be combined with the use of product graphs [15] to create a new graph $\mathcal{G}_\diamond = (\mathcal{V}_\diamond, \mathcal{E}_\diamond, \mathbf{S}_\diamond)$, with node set $\mathcal{V}_\diamond$ of cardinality $|\mathcal{V}_\diamond| = NF$,

edge set $\mathcal{E}_\diamond \subseteq \mathcal{V}_\diamond \times \mathcal{V}_\diamond$ and the $NF \times NF$ graph shift operator $\mathbf{S}_\diamond$. Its sparsity pattern depends by the type of product graph adopted. A novel aspect of the work in [10] comes from the realization that all common types of product graphs, such as the Kronecker and the Cartesian, can be parametrized as:

$$\mathbf{S}_\diamond = \sum_{i=0}^{1} \sum_{j=0}^{1} s_{ij} \left(\mathbf{S}_\mathcal{F}^i \otimes \mathbf{S}^j \right), \tag{3}$$

where $s_{ij} \in \{0, 1\}$, which creates a level of abstraction on the specific product-graph choice for $\mathbf{S}_\diamond$. Given a particular product graph $\mathbf{S}_\diamond$, according to (3), a product graph filter $\mathbf{H}(\cdot)$ can be defined as:

$$\mathbf{H}(\mathbf{S}_\diamond) = \sum_{k=0}^{K-1} h_k \mathbf{S}_\diamond^k. \tag{4}$$

This filter is then used to forecast the F-dimensional graph process $\mathbf{x}_t$ as:

$$\mathbf{x}_t = -\sum_{p=1}^{P} \sum_{k=0}^{K-1} h_{kp} \mathbf{S}_\diamond^k \mathbf{x}_{t-p}, \tag{5}$$

which is termed product graph VAR (PG-VAR) filter, yielding a total of PK number of parameters.

Despite the fact that the PG-VAR takes the information on related features into account, it has some limitations. The first is that it uses a limited amount of parameters independent of the number of features, which might be restrictive. Secondly, even though the edge weights $\mathcal{E}$ and $\mathcal{E}_\mathcal{F}$ may well describe the strength of the relationships in the original graph $\mathcal{G}$ and in the feature graph $\mathcal{G}_\mathcal{F}$, their product-graph combination may not. In order to overcome those limitations, in Section III we propose two new graph-based VAR models with an increased amount of flexibility, and in Section IV we discuss how to optimally learn the feature graph weights during the training of the forecasting process.

III. An Overarching Forecasting Model

In this section, we propose two autoregressive graph-based models, which encompass the already existing ones in the literature, tackle the mentioned limitations and bring increased flexibility. The first is a direct extension of the filters introduced in Section II; the latter is a generalization of the multiple-input multiple-output GF which eliminates the need for knowledge of the feature graph.

Combined. Consider the PG-VAR in (5) and consider the term associated to the index $k = 0$, i.e.,:

$$\mathbf{x}_t = -\sum_{p=1}^{P} h_{0p} \left(\mathbf{I}_F \otimes \mathbf{I}_N \right) \mathbf{x}_{t-p}. \tag{6}$$

The sum in (6) simply represents NF uni-variate auto-regressive filters, where all filters are constrained to have the same set of parameters h_{0p}. This might be a restriction since it weighs similarly all the features in every node.

For this reason, we first propose a minor extension of (5) by adapting its zero-th order coefficients, with a set of F feature-dependent parameters $h_{0p}^{(f)}$. That is:

$$\mathbf{x}_t = -\left(\sum_{p=1}^{P} \left(\operatorname{diag}(\mathbf{h}_{0p}) \otimes \mathbf{I}_N \right) + \sum_{p=1}^{P} \sum_{k=1}^{K-1} h_{kp} \mathbf{S}_\diamond^k \right) \mathbf{x}_{t-p}, \tag{7}$$

where $\mathbf{h}_{0p} = [h_{0p}^{(1)}, \ldots, h_{0p}^{(F)}]^\top \in \mathbb{R}^F$, and $h_{0p}^{(f)}$ represents the zero-th order filter coefficient associated to the fth feature of the pth filter. The GSO $\mathbf{S}_\diamond$ in the second sum represents any type of product graph, without loss of generality. This model can be seen as a combination of a G-VAR model for each feature independently, where the graph filter order is zero, with a PG-VAR model, where the graph filter orders of zero are not included. To even further extend the degrees of freedom of (7), we generalize it through the combination of a G-VAR model, which forecasts each feature independently, with a PG-VAR. We will refer to the combined PG-VAR and G-VAR as the PG-G-VAR model, which is defined as

$$\mathbf{x}_t = -\left(\sum_{p=1}^{P} \sum_{k=0}^{K-1} \left(\operatorname{diag}(\mathbf{h}_{kp}) \otimes \mathbf{S}^k \right) + \sum_{p=1}^{P} \sum_{k=0}^{K-1} h_{kp} \mathbf{S}_\diamond^k \right) \mathbf{x}_{t-p}, \tag{8}$$

where $\mathbf{h}_{kp} = [h_{kp}^{(1)}, \ldots, h_{kp}^{(F)}]^\top \in \mathbb{R}^F$, and $h_{kp}^{(f)}$ is the kth order graph filter coefficient associated to the fth feature of the pth filter. Compared to a G-VAR per feature, the number of parameters is increased only by a small amount to $PK(F+1)$. This increase comes with the flexibility of modeling each feature separately using the G-VAR and also include information of related features using a type of product graph and the importance of each model is weighted by their graph filter coefficients.

MIMO G-VAR. We can further extend the expressiveness of the filter, especially when the feature graph $\mathbf{S}_\mathcal{F}$ is not readily available. To this extent, consider an F-dimensional graph signal $\mathbf{x}_t$, and consider F separate G-VAR filters, all with same orders P and K (without loss of generality), which independently forecast each feature of $\mathbf{x}_t$; i.e.:

$$\mathbf{x}_t = -\sum_{p=1}^{P} \sum_{k=0}^{K-1} \left(\operatorname{diag}(\mathbf{h}_{kp}) \otimes \mathbf{S}^k \right) \mathbf{x}_{t-p}. \tag{9}$$

There are a total of FKP coefficients in this model to estimate. To take the influence that features have on each other into account, the diagonal matrix of filter coefficients can be extended into a full matrix with learnable parameters:

$$\mathbf{x}_t = -\sum_{p=1}^{P} \sum_{k=0}^{K-1} \left(\mathbf{H}_{kp} \otimes \mathbf{S}^k \right) \mathbf{x}_{t-p}, \tag{10}$$

where the matrices $\mathbf{H}_{kp} \in \mathbb{R}^{F \times F}$ contain the filter coefficients associated to each time-lag p and filter order k. To gain more intuition on (10), we can rewrite it as:

$$\mathbf{X}_t = -\sum_{p=1}^{P} \sum_{k=0}^{K-1} \mathbf{S}^k \mathbf{X}_{t-p} \mathbf{H}_{kp}, \tag{11}$$

where $\mathbf{X}_t = [\mathbf{x}_t(1), \ldots, \mathbf{x}_t(F)]$, i.e., it is the matrix which contains in the fth column the one-dimensional graph signal related to feature f at time t. From (11), it is easy to see that the predicted feature values of $\mathbf{x}_t(f)$ are given by a linear combination of F G-VAR models where each model uses a different feature as input. Another interpretation of the operation in (11) is that the left-multiplication of the data matrix $\mathbf{X}_{t-p}$ shifts over the graph $\mathcal{G}$, while the right-multiplication with matrix $\mathbf{H}_{kp}$ shifts (averages) over the features of each individual node. This multidimensional graph filtering operation corresponds to a MIMO graph filter [16], and we refer to this graph-based VAR model as the MIMO G-VAR. This model has a total amount of PKF^2 learnable parameters and a computational complexity of $\mathcal{O}(PKF^2 |\mathcal{E}|)$.

IV. Parameter estimation

In this section we first elaborate on the least-squares (LS) estimation of the graph filter coefficients, then we propose an estimation method that jointly learns the filter coefficients and the feature graph.

A. Multivariate Least Squares Estimator

The first method is the multivariate LS estimator, which is one of the most used methods to estimate VAR coefficients [7]. Notice that all the proposed graph-based VAR models are linear in the filter coefficients; as such, with a proper reshaping of the matrices involved in the filtering operations, they can be estimated in a similar way. For simplicity, we will show the estimation for the G-VAR case. Assume there are $T+P$ multi-dimensional graph signal samples available. Let $\mathbf{h} \in \mathbb{R}^{KP}$ represent the vector that contains the unknown filter coefficients of the G-VAR. The least-squares estimator to find the filter coefficients that best fit the data is given by the argument that minimizes the sum of squared errors. For the G-VAR this is written as,

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h}} \sum_{t=1}^{T} \left\| \mathbf{x}_t + \sum_{p=1}^{P} \mathbf{H}_p(\mathbf{S}) \mathbf{x}_{t-p} \right\|_2^2. \tag{12}$$

where $\mathbf{h} = [h_{01}, \ldots, h_{0P}, \ldots, h_{K1}, \ldots, h_{KP}]^T$. This minimization has a closed-form solution, which for the G-VAR can easily be formulated if the model is rewritten as a matrix-vector product. Using all $T+P$ samples, we define

$$\begin{aligned} \mathbf{X} &= \begin{bmatrix} \mathbf{x}_{P-1} & \cdots & \mathbf{x}_0 \\ \vdots & \ldots & \vdots \\ \mathbf{x}_{T+P-1} & \cdots & \mathbf{x}_T \end{bmatrix} \\ \mathbf{A} &= \begin{bmatrix} \left(\mathbf{I}_T \otimes \mathbf{S}^0\right) \mathbf{X} & \cdots & \left(\mathbf{I}_T \otimes \mathbf{S}^{K-1}\right) \mathbf{X} \end{bmatrix} \\ \mathbf{b} &= [\mathbf{x}_{P-1}^\top, \ldots, \mathbf{x}_{T+P}^\top]^\top \end{aligned} \tag{13}$$

With this notation in place, we can rewrite (12) as:

$$\hat{\mathbf{h}} = \operatorname*{argmin}_{\mathbf{h}} \|\mathbf{b} + \mathbf{A}\mathbf{h}\|_2^2, \tag{14}$$

for which the LS solution is given by:

$$\hat{\mathbf{h}} = -(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = -\mathbf{A}^\dagger\mathbf{b}. \tag{15}$$

B. A Joint estimation approach

When the feature graph is not available from the context of interest, a possible approach is to learn it together with the filter coefficients. To find the feature graph that best suits the product graph-based VAR model, we propose to identify the weights of the GSO by minimizing the forecasting error. As the optimal graph filter coefficients are dependent on the used feature GSO we jointly estimate them, e.g. for the Kronecker PG-VAR this joint problem is defined as:

$$\begin{aligned} \hat{\mathbf{h}}, \hat{\mathbf{S}}_{\mathcal{F}} = \operatorname*{argmin}_{\mathbf{h}, \mathbf{S}_{\mathcal{F}}} & \sum_{t=1}^{T} \left\| \mathbf{x}_t - \sum_{p=1}^{P} \sum_{k=0}^{K-1} h_{kp} (\mathbf{S}_{\mathcal{F}} \otimes \mathbf{S})^k \mathbf{x}_{t-p} \right\|_2^2 \\ \text{s.t.} \quad & \operatorname{supp}(\mathbf{S}_{\mathcal{F}}) \subseteq \operatorname{supp}(\mathbf{S}_{\mathcal{F}}^{(0)}) \end{aligned} \tag{16}$$

where $\mathbf{S}_{\mathcal{F}}^{(0)}$ represents the initial chosen GSO of the feature graph and $\operatorname{supp}(\cdot)$ is the set of non-zero elements of the argument, i.e., we assume its sparsity pattern is available from the application at hand. This is a non-convex problem, due to the polynomials of the feature GSO.

In order to tackle the non-convexity of the problem, we use the method in [17], where $\mathbf{h}$ and $\mathbf{S}_{\mathcal{F}}$ are estimated iteratively using an alternating minimization (AM) approach where the non-convex portion of the problem is solved through the sequential convex programming (SCP) paradigm [18]. The pseudo-code that describes our AM approach is given in Algorithm 1. In this algorithm, we use the estimate of $\mathbf{S}_{\mathcal{F}}$ at the $(n-1)$th iteration to find the vector of parameters $\mathbf{h}$, which is obtained by solving

$$\hat{\mathbf{h}} = \operatorname*{argmin}_{\mathbf{h}} \sum_{t=1}^{T} \left\| \mathbf{x}_t - \sum_{p=1}^{P} \sum_{k=0}^{K-1} h_{kp} (\mathbf{S}_{\mathcal{F}} \otimes \mathbf{S})^k \mathbf{x}_{t-p} \right\|_2^2. \tag{17}$$

As shown above, in Section IV-A, this is a linear least-squares problem, which has a closed-form solution as defined in (15). We then use the estimated graph-based VAR parameters at the n^{th} iteration to estimate $\mathbf{S}_{\mathcal{F}}$. This estimate is found by minimizing the original objective function with respect to the feature GSO:

$$\begin{aligned} \hat{\mathbf{S}}_{\mathcal{F}} = \operatorname*{argmin}_{\mathbf{S}_{\mathcal{F}}} & \sum_{t=1}^{T} \left\| \mathbf{x}_t - \sum_{p=1}^{P} \sum_{k=0}^{K-1} h_{kp} (\mathbf{S}_{\mathcal{F}} \otimes \mathbf{S})^k \mathbf{x}_{t-p} \right\|_2^2 \\ \text{s.t.} \quad & \operatorname{supp}(\mathbf{S}_{\mathcal{F}}) \subseteq \operatorname{supp}(\mathbf{S}_{\mathcal{F}}^{(0)}) \end{aligned} \tag{18}$$

This problem is still non-convex, but it is a lighter problem than (16). A non-convex method can be used to obtain $\hat{\mathbf{S}}_{\mathcal{F}}$, as the gradient is not hard to find, we use the sequential quadratic programming (SQP) method; see [17] for details.

Algorithm 1 Joint GF and feature GSO

Require: $\mathbf{S}_{\mathcal{F}}^{(0)}, \epsilon > 0$
1: $n \leftarrow 1$
2: **while** not converged **do**
3: $\quad \mathbf{h}^{(n)} \leftarrow \arg\min_{\mathbf{h}} f\left(\mathbf{h}, \mathbf{S}_{\mathcal{F}}^{(n-1)}\right)$ ▷ See equation (17)
4: $\quad \mathbf{S}_{\mathcal{F}}^{(n)} \leftarrow \arg\min_{\mathbf{S}_{\mathcal{F}}} f\left(\mathbf{h}^{(n)}, \mathbf{S}_{\mathcal{F}}\right)$ ▷ See equation (18)
5: $\quad$ Check convergence $(\mathbf{h}^{(n)}, \mathbf{S}_{\mathcal{F}}^{(n)}, \epsilon)$
6: $\quad n \leftarrow n+1$
7: **end while**
8: **return** $\mathbf{h}^{(n)}, \mathbf{S}_{\mathcal{F}}^{(n)}$

V. Numerical Results

To evaluate the models we use a sliding window cross-validation setup [19]; specifically, the time series are split along the temporal axis into three parts: an in-sample, an out-of-sample, and a left-out part. The in-sample data, which is further split into a training and validation set following a $70\%/30\%$ scheme [20], is used to find the optimal model hyperparameters through a grid search (such as P and K) and to estimate the filter coefficients; the out-of-sample data serves as a "test" set to measure the prediction accuracy. The left-out part of each iteration consist of the data that is not taken into account. At each iteration the in-sample and out-of-sample parts slide over the data set, creating multiple data sets where the models can be trained and evaluated over. The performance metric considered is the root normalized mean squared error (RNMSE) over the out-of-sample data for all iterations, which is defined as:

$$\text{RNMSE} = \sqrt{\frac{\sum_{t=1}^{\tau} \|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|_2^2}{\sum_{t=1}^{\tau} \|\mathbf{x}_t\|_2^2}}, \tag{19}$$

where τ is the number of signals considered to compute the RNMSE, $\mathbf{x}_t$ is the true graph signal, and $\tilde{\mathbf{x}}_t$ is the predicted signal.

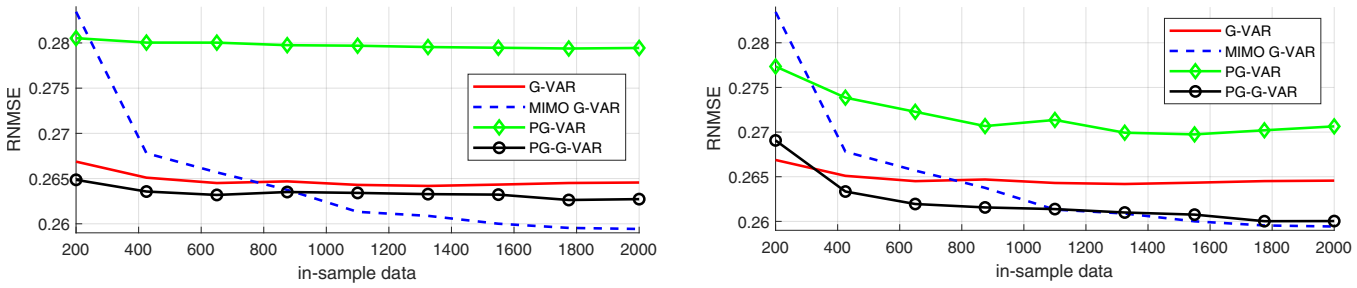

Fig. 1: RNMSE versus the amount of in-sample data for the different graph-based models. Where at (a) the results are shown when the initial feature graph is used and at (b) the joint estimation method is used to find the optimal feature edge weights.

A. Data set

We evaluate the performance of the graph-based forecasting models on the Beijing air-quality data set [21], [22]. There are $F = 10$ features considered, 6 types of air pollutants (PM2.5, PM10, SO2, NO2, CO, O3) and 4 weather-related variables (temperature, pressure, dew point, and wind speed), all recorded by $N = 12$ air-quality monitoring stations, representing our nodes, in the Beijing area. The data considered is from 20 July 2015 at 7:00 to 5 September 2016 at 13:00, which results in a set of $T = 9918$ hourly measurements. The graph regarding the measurement stations is constructed with a 3 nearest neighbors approach, based on the geographical distances and has edge weights defined by a Gaussian kernel weighting function, similar to [6]. The feature graph is constructed by connecting each feature to its two most correlated features. We use the normalized Laplacian as GSO for both the station graph and the feature graph.

B. Results

We evaluate our proposed models , the MIMO G-VAR and the PG-G-VAR, and compare their performance with the PG-VAR and G-VAR. The Cartesian graph product is considered as the product graph type that models the relations between the station graph and the feature graph. The following sets of parameter values are taken into consideration : $P, K \in \{1, \ldots, 5\}$. For all data sets, the range of in-sample data samples considered is from 200 to 2000, and the out-of-sample data consists of 168 hourly measurements, i.e. one week of data. The amount of iterations is 20, and at every iteration, all data is shifted with the number of out-of-sample data points.

First, we evaluate the results with the initial defined feature GSO and do not use the joint estimation method to estimate it. From the results in Fig. 1a, it can be seen that, as we expected due to its limitations, the PG-VAR model has a decreased performance compared to the G-VAR for each feature. The combined PG-G-VAR model increases the accuracy compared to the G-VAR. The MIMO G-VAR needs more training data since it is the most flexible, but eventually has the best performance. Secondly, in Fig. 1b the results are shown for the case we do estimate the weights of the feature GSO jointly with the graph filter coefficients. A significant increase in accuracy can be seen for the PG-VAR and PG-G-VAR models. However, although the PG-VAR shows an improvement it still performs less than the other methods. The joint estimation method results for the PG-G-VAR model show that it outperforms all other methods for smaller amounts of in-sample data. Except for the 200 in-sample data size where it has a similar performance as the G-VAR, and for larger amounts of in-sample data where it has now a similar performance as the MIMO G-VAR model. This illustrates the importance of estimating the edge weights of the feature graph when a product graph is applied. Further, it showcases the advantage offered by using a priori knowledge of the feature graph, which makes it work well with lower amounts of in-sample data.

VI. Conclusion

In this paper, we proposed two new models to forecast multi-dimensional graph signals, the PG-G-VAR and MIMO G-VAR models. Further, we applied a joint estimation approach to estimate the graph filter coefficients together with the weights of the feature graph. The results indicate that our limitation assumptions regarding the PG-VAR are correct, as it shows a reduced prediction performance compared to the G-VAR model per feature. On the other hand, the results show a clear improved prediction accuracy for the models introduced in this paper, especially for more training data.

Further work is needed to enhance the proposed PG-G-VAR model's estimation cost. Especially in the direction of finding optimal weights of the feature graph, as the numerical experiments showed that estimating $\mathbf{S}_{\mathcal{F}}$ leads to increased forecasting performance, but with the cost that a non-convex problem has to be solved. A possible solution for this would be to apply a constrained edge-variant graph filter [23], to the feature graph. In this filter, each element in the GSO is weighted individually by a graph filter coefficient, before it is used in the filtering operation. This results in a linear model with respect to these graph filter coefficients.

REFERENCES

[1] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[2] Ireneusz Jabłoński, "Graph signal processing in applications to sensor networks, smart grids, and smart cities," *IEEE Sensors Journal*, vol. 17, no. 23, pp. 7659–7666, 2017.

[3] Matthew A Russell, *Mining the social web: data mining Facebook, Twitter, LinkedIn, Google+, GitHub, and more*, " O'Reilly Media, Inc.", 2013.

[4] Dmitri Goldenberg, "Social network analysis: From graph theory to applications with python," *arXiv preprint arXiv:2102.10014*, 2021.

[5] Jonathan Mei and José MF Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077–2092, 2016.

[6] Elvin Isufi, Andreas Loukas, Nathanael Perraudin, and Geert Leus, "Forecasting time series with varma recursions on graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 18, pp. 4870–4885, 2019.

[7] Helmut Lütkepohl, *New introduction to multiple time series analysis*, Springer Science & Business Media, 2005.

[8] Daniel Romero, Vassilis N Ioannidis, and Georgios B Giannakis, "Kernel-based reconstruction of space-time functions on dynamic graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 856–869, 2017.

[9] Paolo Di Lorenzo, Paolo Banelli, Elvin Isufi, Sergio Barbarossa, and Geert Leus, "Adaptive graph signal processing: Algorithms and optimal sampling strategies," *IEEE Transactions on Signal Processing*, vol. 66, no. 13, pp. 3584–3598, 2018.

[10] Alberto Natali, Elvin Isufi, and Geert Leus, "Forecasting multi-dimensional processes over graphs," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 5575–5579.

[11] Aliaksei Sandryhaila and Jose MF Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE signal processing magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[12] 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.

[13] 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.

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

[15] Richard H Hammack, Wilfried Imrich, Sandi Klavžar, Wilfried Imrich, and Sandi Klavžar, *Handbook of product graphs*, vol. 2, CRC press Boca Raton, 2011.

[16] Fernando Gama, Antonio G Marques, Alejandro Ribeiro, and Geert Leus, "Mimo graph filters for convolutional neural networks," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2018, pp. 1–5.

[17] Alberto Natali, Mario Coutino, and Geert Leus, "Topology-aware joint graph filter and edge weight identification for network processes," in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2020, pp. 1–6.

[18] John Duchi, Stephen Boyd, and Jacob Mattingley, "Sequential convex programming," *Notes for EE364b, Stanford University*, 2018.

[19] Vitor Cerqueira, Luis Torgo, and Igor Mozetič, "Evaluating time series forecasting models: An empirical study on performance estimation methods," *Machine Learning*, vol. 109, pp. 1997–2028, 2020.

[20] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2, Springer, 2009.

[21] Shuyi Zhang, Bin Guo, Anlan Dong, Jing He, Ziping Xu, and Song Xi Chen, "Cautionary tales on air-quality improvement in beijing," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2205, pp. 20170457, 2017.

[22] Dheeru Dua and Casey Graff, "UCI machine learning repository," 2017.

[23] E. Isufi, *Graph-time signal processing: Filtering and sampling strategies*, Ph.D. thesis, Delft, The Netherlands, 2019.