Forecasting Models for Graph Processes

A Study on the Multi-Dimensional Case

FORECASTING MODELS FOR GRAPH PROCESSES

A STUDY ON THE MULTI-DIMENSIONAL CASE

Thesis

to obtain the degree of Master of Science in Electrical Engineering,
at the Delft University of Technology,
to be publicly defended on Monday February 21, 2022.

by

Jelmer VAN DER HOEVEN

Thesis Committee:

Prof. dr. ir. G.J.T. Leus,	TU Delft, Chair
Dr. E. Isufi	TU Delft
Msc. A. Natali,	TU Delft, Daily Supervisor

Keywords: Forecasting, graph signal processing, multi-dimensional signal processing, product graphs

ABSTRACT

In the current Big Data era, large amounts of data are collected from complex systems, such as sensor networks and social networks. The emerging field of graph signal processing (GSP) leverages a network structure (graph) to process signals on an irregular domain. This thesis studies the forecasting of multi-dimensional graph processes, i.e., where each entity in the network carries a multivariate time series. Recent research has proposed to use product graphs to model the dependencies between different variables in multi-dimensional graph processes and employ them in graph-based vector autoregressive models to predict future values. A problem with these product graph-based models is that they can be too restrictive. In this work, it is proposed to combine product graph-based models with multiple one-dimensional models to implement more estimation flexibility. To further increase the degrees of freedom, the use of multiple-input-multiple-output graph filters is also proposed. The proposed models are implemented and tested on synthetic and real-world data sets, which shows an improved forecasting performance compared to state-of-the-art alternatives.

ACKNOWLEDGEMENTS

At this moment, I am a just a few steps away from obtaining my Master of Science degree. Hereby I would like to thank everyone who helped and supported me over the past year to successfully complete this thesis project.

First, I would like to express my gratitude to both my supervisors, with their advice and feedback they made it a great learning experience. I would like to thank my daily-supervisor, Alberto Natali, for his guidance and all the fruitful discussions. Also, with the necessary coffee breaks, he made working at the department more fun. I would like to thank Professor Geert Leus for his supervision. Whenever I was stuck or wanted discuss some new ideas, his door was always open. I am also like to thank and aw knowledge Elvin Isufi, for taking the time to being part of my thesis defense committee.

Secondly, I want to thank Tobias Bonsen and Paul Treanor, for reading my thesis and and proving me with feedback. Further, I want to thank my boyfriend, Dion Treanor, to help me keep focused when I had trouble concentrating and made sure I took time for the necessary relaxation. Lastly, I would like to thank my parents for believing in me and their support. Whenever I needed, they were there to help.

Jelmer van der Hoeven
Delft, February 2022

CONTENTS

ABBREVIATIONS

(I)GFT (Inverse) Graph Fourrier Transform.

Dof Degrees of Freedom.

FIR Finite Impulse Response.

GPG Generalized Product Graph.

GSO Graph Shift Operator.

GSP Graph Signal Processing.

LS Least-Squares.

LSI Linear Shift Invariant.

MIMO Multiple-Input-Multiple-Output.

PG Product Graph.

PPG Parametric product graph.

rAEPR root Average Excess Prediction Risk.

VAR Vector AutoregRessive.

YW Yule-Walker.

NOMENCLATURE

Graph theory

$\mathbf{L}$	Graph Laplacain matrix
$\mathbf{W}$	Weighted adjacency matrix
$\mathbf{D}$	Degree matrix
$G = (\mathcal{V}, \mathcal{E})$	Graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$
$\mathcal{N}_i$	Set of neighbors of node i
$\diamond$	General graph product

Linear Algebra

$\mathbf{A}^T$	Transpose of $\mathbf{A}$
$\mathbf{A}^{-1}$	Inverse of $\mathbf{A}$
$\mathbf{A}^\dagger$	Pseudo-inverse of $\mathbf{A}$
$\otimes$	Kronecker matrix product
$\odot$	Element-wise matrix product
vec(A)	Vectorization of $\mathbf{A}$

Mathematical objects

a	Scalar
$\mathbf{a}$	Vector
$\mathbf{A}$	Matrix
a_i	The ith element of $\mathbf{a}$
$\mathbf{A}_{ij}$	The (i, j)th element of $\mathbf{A}$
$\mathbf{I}_N$	$N \times N$ identity matrix

Other symbols

$\mathbb{E}(\cdot)$	Expectation operation
$\mathcal{O}$	Big O notation

σ	point-wise nonlinearity

Set theory

$\mathbb{R}$	Set of real numbers
$\in$	Belong to

1

INTRODUCTION

The forecasting of processes that evolve over time is a crucial topic in signal processing [1]. Especially in the current era of big data, much complex data is available from which useful information can be extracted to predict future trends. This data can be derived from complex systems, such as sensor networks [2, 3], social networks [4, 5], and protein-protein interaction networks [6, 7]. The underlying network structure can be used to efficiently process large volumes of data, since neighboring elements in the network are likely to influence each other.

For example, opinion data collected from a social network, which stores the connections between people, contains influences from people closely related to each other [8]. Similarly, in an air-quality monitoring network, measurements from a particular station are related to its previous measurements and also to data from neighboring stations. Information about time and the network, which has an irregular structure, should be accounted for in a model to enhance forecasting. Classical signal processing does not offer the tools to deal with this type of irregularly structured data. Therefore, recent research has focused on extending methods for regularly structured data (e.g., time signals and images) to irregular network structured data [9, 10].

A possibility to model such irregular structures is offered by graphs, which are mathematical objects composed of nodes and edges, where nodes represent objects and edges capture the relationship between objects [11]. In the example of the air-quality monitoring network, measurement stations can be defined as nodes, and edges define the spatial relationship between them. An illustrative example of a graph that captures the spatial relationships in an air-monitoring network is depicted in Figure 1.1. The field of graph signal processing (GSP) has emerged to extend classical signal processing techniques, applied to signals defined on a regular domain, to signals defined on irregular domains [12, 13]. Thus far, GSP methods have been developed to solve signal processing problems, such as filtering, denoising, tracking, and forecasting [14, 15, 16].

Most graph signal forecasting models consider a one-dimensional signal at each node [17, 18, 19]. There are also scenarios in which a multi-dimensional signal evolves on each node of a network. For example, when a sensor in a network performs different

1

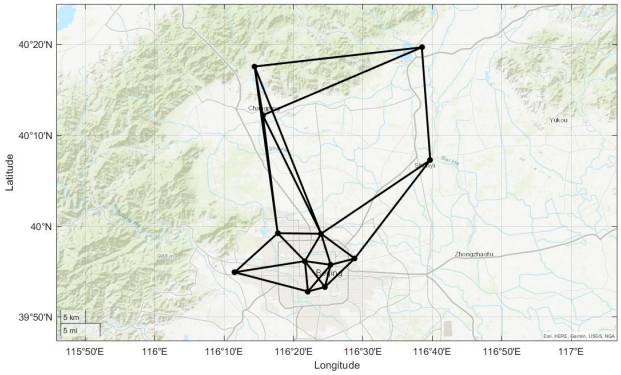


Figure 1.1: Locations of the 12 air-quality monitoring stations in the region of Beijing . The graph is built using a 3-nearest neighbor approach.

kinds of measurements, each node carries a vector of data (features) at every instant. There might be some kind of relationship between the features themselves, which may also be captured by a graph. This graph can add prior information about the domain where the data resides to a model, which may help in low-amount of data situations. In order to model the relationship between features and nodes, an extended graph can be defined that captures the information from both graphs. A suitable option to represent this extended graph is through a product graph [20]. In recent work, product graphs-based models have been applied to predict multi-dimensional processes on graphs [21]. However, more experiments to determine the efficacy of these models to enhance the forecasting of multi-dimensional graph processes are required.

In this thesis, we address the problem of forecasting multi-dimensional graph processes. In order to improve the forecasting accuracy, we propose models that have more freedom to incorporate the relationship between features. Extensive experiments were performed on our proposed models as well as state-of-the-art models to verify their efficacy. The scope of this research is limited to linear models, yet they can be extended to nonlinear approaches.

RESEARCH STATEMENT

This thesis focuses on answering the following research questions:

RQ1: *"How to incorporate the relationship between features of multi-dimensional graph signals into a model to enhance their prediction?"*

RQ2: *"How to learn the parameters that capture the influence between features in multi-dimensional graph signal models?"*

RQ3: *"How well do the proposed architectures perform the task of forecasting multi-dimensional graph processes?"*

To answer the posed research questions, a literature study is first performed on the theory of GSP, and the state-of-the-art linear multi-dimensional graph-based forecasting models. Using the knowledge obtained from the literature, new models to forecast multi-dimensional graph processes are proposed. These models are extensions of state-of-the-art models, where more flexibility is added to enhance the forecasting of multi-dimensional graph processes. To validate the performance, the models are implemented and tested. First, tests are done on synthetic data generated according to different product graph-based models. Finally, the proposed and state-of-the-art models are validated on two real-world datasets.

THESIS OUTLINE

The thesis structure is as follows: Chapter 2 introduces the theory of graph signal processing and discusses the state-of-the-art models to forecast multi-dimensional graph signals. In Chapter 3 the proposed multi-dimensional graph models are presented. Chapter 4 evaluates the performance of our proposed models on synthetic data, while in Chapter 5 we evaluate the proposed models on real-world data. The conclusions and future research directions are discussed in Chapter 6.

2

BACKGROUND

This chapter presents the necessary background on graph signal processing to conduct the research provided in this thesis and addresses relevant work. Section 2.1 introduces the notion of graphs and graph signals. Further, it discusses the concepts of graph filters. In Section 2.2, we present multi-dimensional graph signals and extend the concept of graph filters to the multi-dimensional domain. Finally, we discuss models that model time-varying graph signals in Section 2.3.

2.1. GRAPH SIGNAL PROCESSING

2.1.1. GRAPHS

A graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is defined by a set of N nodes $\mathcal{V} = \{v_1, v_2, ..., v_N\}$, and by a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ which defines how the nodes are connected [11]. An edge between node $u \in \mathcal{V}$ and node $v \in \mathcal{V}$ is denoted as $(u, v) \in \mathcal{E}$. A graph is called directed if the edges have a direction. This means, that if node i is connected to node j, it is not implied that there is an edge that connects node j to node i. For an undirected graph, this implication does hold. A graph can also be represented in an algebraic manner using the adjacency matrix. The adjacency matrix $\mathbf{A}$ is a binary $N \times N$ matrix such that each element $\mathbf{A}_{ij}$ contains a one if there is an edge between node v_1 and v_j, otherwise $\mathbf{A}_{ij} = 0$. In the case of an undirected graph, the adjacency matrix is symmetric. The matrix $\mathbf{W}$ denotes the weighted adjacency matrix, containing the weights of the edges. The next definitions are defined for undirected graphs. The degree of node i is the sum of edge weights connected to node i,

$$d_i = \sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij}, \tag{2.1}$$

where $\mathcal{N}_i = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$ is the set of nodes that define the neighborhood of node i. The degree matrix $\mathbf{D}$ is a diagonal matrix where the i^{th} diagonal element is given by d_i,

$$\mathbf{D}_{ii} = d_i. \tag{2.2}$$

Another matrix representation of a graph is the graph Laplacian, which is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{2.3}$$

The normalized graph Laplacian is defined as

$$\mathbf{L}_{norm} = \mathbf{D}^{-\frac{1}{2}} L \mathbf{D}^{-\frac{1}{2}}, \tag{2.4}$$

and has the property that its eigenvalues satisfy $0 = \lambda_1 \leq \ldots \leq \lambda_N \leq 2$ [22].

2.1.2. SIGNALS ON GRAPHS

The values on top of nodes can be represented by graph signals. A graph signal $\mathbf{x} = [x(1), ..., x(N)]^T \in \mathbb{R}^N$ is defined as the vector containing N data elements, where each data element $x(i)$ corresponds to the value residing on node $v_i \in \mathcal{V}$. For example, a set of measurements in a sensor network can be represented by a graph signal, this is illustrated by Figure 2.1. Each node in the network is a sensor and is connected by edges to its neighbors. The measurement of each sensor is stored in the graph signal $\mathbf{x}$.
A time series can also be represented as a graph signal. The graph that models the underlying structure of a finite periodic times series is represented by a directed cyclic graph [23, 10] , which is shown in Figure 2.2. Each node represents a time instant, and the directed links capture the causality of the time series. The periodic extension of the signal is captured by the edge between node v_N and v_1. In the next sections, the directed cyclic graph will be used to illustrate that graph signal processing can be seen as a generalized form of classical discrete signal processing.

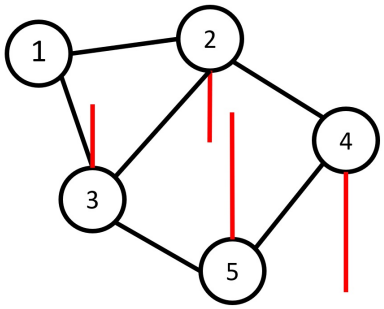


Figure 2.1: A undirected graph of five nodes, with graph signal $\mathbf{x} = [0, -1, 1, -2, 2]$. The red bars represent the signal value

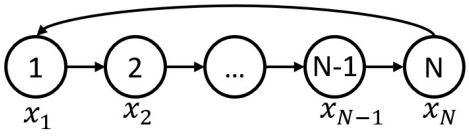


Figure 2.2: Directed cyclic graph

2.1.3. GRAPH SHIFT OPERATOR

The graph shift operator (GSO) is one of the fundamental building blocks in GSP, and defines the one-step information flow over a graph and is noted by the matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ [12, 24]. The shifted version of a graph signal $\mathbf{x}$, is a new graph signal obtained by the matrix-vector multiplication $\mathbf{Sx}$. The n-shifted graph signal, denoted as $\mathbf{x}^{(n)}$, is given by

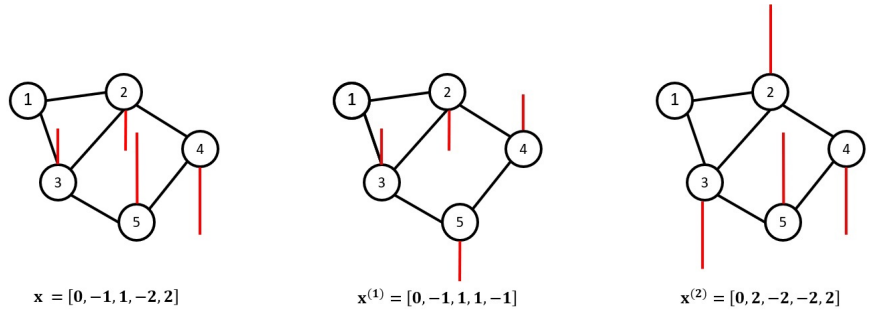
$$\mathbf{x}^{(n)} = \mathbf{S}\mathbf{x}^{(n-1)} = \mathbf{S}^n\mathbf{x}. \tag{2.5}$$

A shift in the classical signal processing sense is similar to that of a graph shift operation performed on a directed cyclic graph. If the adjacency matrix is used as GSO, i.e. $\mathbf{S} = \mathbf{A}$, the value of a node is passed to its one-hop neighbor. The one-shifted graph signal of the graph depicted in Figure 2.2 is

$$\begin{bmatrix} x_N \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}.$$

In the cyclic graph example, the energy of the signal remains constant for every shift, but this is often not the case for more complex graphs, as seen in Figure 2.3. Using the adjacency matrix as the choice for $\mathbf{S}$, the shifted node values are a summation of all its neighboring node values. When the graph Laplacian is used as choice for $\mathbf{S}$, the one-shifted value of node i is given by $x_i^{(1)} = \sum_{j \in \mathcal{N}_i} W_{ij}(x_i - x_j)$, where $\mathcal{N}_i$ defines the set of

nodes that are connected to node i. Both the adjacency matrix and the graph Laplacian are common choices for the GSO, as well as variations of them [25]. The adjacency matrix can be used with undirected and directed graphs, while the graph Laplacian is mainly used for undirected graphs, as $\mathbf{L}$ is symmetric and positive semidefinite [10].



(a) Original graph signal (b) one-shifted graph signal (c) two-shifted graph signal

Figure 2.3: Illustration of the graph shift operation, where the adjacency matrix is used as GSO. i.e. $\mathbf{S} = \mathbf{A}$.

2.1.4. GRAPH FOURIER TRANSFORM

The graph Fourier transform (GFT) uses the eigendecomposition of the GSO. If the the Laplacian is used as GSO, i.e. $\mathbf{S} = \mathbf{L}$, it can be decomposed as $\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^H$, where $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ ...\ \mathbf{u}_N]$ is the orthonormal matrix of eigenvectors. $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues, ordered from $0 = \lambda_1 \leq \lambda_2 \ldots \leq \lambda_N := \lambda_{\max}$. The graph Laplacian has only real eigenvalues as it is a real and symmetric matrix. The GFT of a graph signal $\mathbf{x}$ is defined as

$$\hat{\mathbf{x}} = \mathbf{U}^H\mathbf{x} \tag{2.6}$$

With the unitary property that $\mathbf{U}^H\mathbf{U} = \mathbf{I}$, the inverse graph Fourier transform (IGFT) is then

$$\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}. \tag{2.7}$$

The eigenvalues and eigenvectors of the GSO provide a notion of frequency [13]. Eigenvectors associated with small eigenvalues vary in a smooth way over the graph in comparison with higher eigenvalues. When having a directed graph, the adjacency matrix can be used as GSO, which can be decomposed using the Jordan decomposition $\mathbf{S} = \mathbf{U}\mathbf{J}\mathbf{U}^H$ [25].

2.1.5. GRAPH SIGNAL FILTERING: SPECTRAL-DOMAIN

In GSP filters can be used in a similar way as they are used in traditional signal processing. With filtering in the frequency domain, certain frequencies of a signal are amplified or suppressed. Using the GFT to represent a graph signal in the frequency domain, the described filtering operation can be applied [13]. The graph filter $h(\lambda_k)$ is a function that scales the k^{th} frequency component of $\hat{\mathbf{x}}$ to its desired output. This can be denoted by the point-wise multiplication of the frequency components with the graph

filter, $\hat{y}(k) = h(\lambda_k)\hat{x}(k)$. The output vector $\hat{\mathbf{y}}$ of filtered graph frequencies is given by,

$$\hat{\mathbf{y}} = h(\mathbf{\Lambda})\hat{\mathbf{x}}, \tag{2.8}$$

where $h(\mathbf{\Lambda})$ is the diagonal matrix that contains the the frequency response vector $\mathbf{h} = [h(\lambda_1) \cdots h(\lambda_N)]$. The filtered graph signal can now be obtained by converting $\hat{\mathbf{y}}$ back to the vertex-domain using the IGFT. The three steps that describe the filtering of a graph signal in the spectral domain, computing the GFT of the input graph signal (2.6), scaling $\hat{\mathbf{x}}$ (2.8) and applying the IGFT (2.7), can be written in one equation as

$$\mathbf{y} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^H\mathbf{x} = \mathbf{H}\mathbf{x}. \tag{2.9}$$

Computing the eigenvectors of the GSO and applying the (I)GFT are computationally expensive operations, especially for large graphs. Because of this filtering in the vertex-domain may be preferable [10, 26]. This can be done by constructing the graph filter $\mathbf{H}$, so the output of the filter is given matrix-vector multiplication, with a computational cost of $O(N^2)$.

2.1.6. GRAPH SIGNAL FILTERING: VERTEX DOMAIN

A standard choice of a graph filter in the vertex domain is the finite impulse response (FIR) filter, which is a linear, shift-invariant (LSI) graph filter [12, 26]. All LSI graph filters of order K are given by a linear combination of filter taps and powers of the GSO and are denoted by

$$\mathbf{H}(\mathbf{S}) = \sum_{k=0}^{K-1} h_k \mathbf{S}^k. \tag{2.10}$$

The FIR graph filter operation performed on a graph signal $\mathbf{x}$ is given by

$$\mathbf{y} = h_0\mathbf{x} + h_1\mathbf{S}\mathbf{x} + \ldots + h_{K-1}\mathbf{S}^{K-1}\mathbf{x} = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x}. \tag{2.11}$$

For $k = 0$ the GSO is the identity matrix, i.e. $\mathbf{S} = \mathbf{I}_N$. Since matrix $\mathbf{S}$ is sparse, the computational complexity of the graph filtering operation is $\mathcal{O}(|\mathcal{E}|K)$. This is cheaper than the spectral filter in equation (2.9), as $|\mathcal{E}|$ is of comparable size as N for sparse graphs, and the filter order K is often small [27].

2.2. MULTI-DIMENSIONAL GRAPH SIGNAL PROCESSING

2.2.1. MULTI-DIMENSIONAL GRAPHS SIGNALS

In the case of multi-dimensional graph processes, each node in the network holds a feature vector containing multiple types of values. The graph signal that captures all the data from the multiple features is called a multi-dimensional graph signal. An example of a multi-dimensional graph signal is given in Figure 2.4. In this example, the graph resembles for instance a sensor network, wherein the measured features could be temperature, humidity, and air pressure. An F-dimensional graph signal can be represented by a matrix

$$\mathbf{X} = \begin{bmatrix} | & & | \\ \mathbf{x}^1 & \cdots & \mathbf{x}^F \\ | & & | \end{bmatrix} \in \mathbb{R}^{N \times F}, \tag{2.12}$$

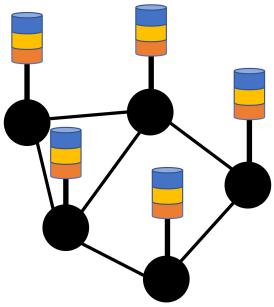


Figure 2.4: A three-dimensional graph signal, where each color represents a feature.

or by a vector, where all the individual graph signals of each feature are stacked together

$$\mathbf{x} = \text{vec}(\mathbf{X}) = \begin{bmatrix} \mathbf{x}^1 \\ \vdots \\ \mathbf{x}^F \end{bmatrix} \in \mathbb{R}^{NF}. \tag{2.13}$$

One can view the multi-dimensional graph signal as F different graph signals, where the entries in the feature vector are somehow related to each other. By processing the multi-dimensional graph signal, instead of each graph signal separately, those relationships can be captured. The discussed notations of multi-dimensional graph signals will be used interchangeably in the rest of the thesis and will be made clear by the context.

2.2.2. MIMO GRAPH FILTER

The MIMO graph filter uses information from multiple features as input to construct multiple output signals. This type of graph filter is mainly used in graph convolutional neural networks, to connect the features between different layers [28]. Without loss of generality, we will consider that the amount of output and input signals are the same.

The output signal of the MIMO graph filter is defined by the summation of F input graph signals (one for each feature), which are processed by F different graph filters. The summation to compute the ith output signal is given by

$$\mathbf{y}^i = \sum_{k=0}^{K-1} \left(h_{ki1}\mathbf{S}^k\right)\mathbf{x}^1 + \ldots + \sum_{k=0}^{K-1} \left(h_{kiF}\mathbf{S}^k\right)\mathbf{x}^F = \sum_{j=1}^{F}\sum_{k=0}^{K-1} \left(h_{kij}\mathbf{S}^k\right)\mathbf{x}^j \tag{2.14}$$

where h_{kij} is the kth filter tap corresponding to the graph filter processing the input graph signal $\mathbf{x}^j$ and is used to obtain the output signal $\mathbf{y}^i$. The influence each feature has on the output is captured by the filter coefficients. Using the matrix form of the F-dimensional graph signal as defined in (2.12), this equation can also be written as

$$\mathbf{y}^i = \sum_{k=0}^{K-1} \mathbf{S}^k\mathbf{X}\mathbf{h}_k, \tag{2.15}$$

where $\mathbf{h}_k = \left[h_{ki1}, \ldots, h_{kiF}\right]^T$. This equation shows in a more intuitive way how each output signal is a linear combination of the input signals from the F available features. To

write the computation of all the the F output signals, the previous equation can be expanded into

$$\mathbf{Y} = \sum_{k=0}^{K-1} \mathbf{S}^k \mathbf{X} \mathbf{H}_k, \tag{2.16}$$

where $\mathbf{Y} \in \mathbb{R}^{N \times F}$ contains the F-dimensional output graph signal and $\mathbf{H}_k \in \mathbb{R}^{F \times F}$ contains the filter coefficients corresponding to kth filter tap of the F^2 employed graph filters. By using the Kronecker matrix product, denoted by $\otimes$, the MIMO graph filter can also be written in another manner as

$$\mathbf{y} = \sum_{k=0}^{K-1} \left(\mathbf{H}_k \otimes \mathbf{S}^k \right) \mathbf{x}, \tag{2.17}$$

where $\mathbf{y}$ and $\mathbf{x}$ are respectively the F-dimensional output and input graph signals, as defined in (2.13). There are a total F^2K number of trainable parameters, and the computational complexity of performing the MIMO graph filter operation is $\mathcal{O}(|\mathcal{E}|F^2K)$. The MIMO graph filter has a lot of trainable parameters, which can be helpful in learning the relationship between the features; when the relationship between features is known, this information could be incorporated into a model acting as an inductive bias to reduce the estimation complexity.

2.2.3. PRODUCT GRAPHS

We assume there exists a relationship between features that can be captured by a graph, referred to as the feature graph. Using this graph structure could be beneficial to model a multi-dimensional network process in an efficient way. Consider a feature graph $\mathcal{G}_\mathcal{F} = (\mathcal{V}_\mathcal{F}, \mathcal{E}_\mathcal{F}, \mathbf{S}_\mathcal{F})$ with a vertex set $\mathcal{V}_\mathcal{F} = \{v_{f1}, \ldots, v_{fF}\}$ of F nodes, where each node represents a feature. $\mathcal{E}_\mathcal{F}$ is the set of edges containing the information about how the features are connected. The feature GSO is given by the matrix $\mathbf{S}_\mathcal{F} \in \mathbb{R}^{F \times F}$. To capture the intra-connections between the feature graph of a node in a network to another node in the network, we can define a graph that combines both $\mathcal{G}$ and $\mathcal{G}_\mathcal{F}$ into a new graph [21]. This can be done by taking the product graph of the two graphs [20]. The most common product graphs are the Cartesian, Kronecker, and Strong graph products [29, 30]. The product graph between $\mathcal{G}$ and $\mathcal{G}_\mathcal{F}$ is mathematically written as

$$\mathcal{G}_\diamond = \mathcal{G} \diamond \mathcal{G}_\mathcal{F}, \tag{2.18}$$

where the symbol $\diamond$ represents the type of product graph used to generate the new graph $\mathcal{G}_\diamond = (\mathcal{V}_\diamond, \mathcal{E}_\diamond, \mathbf{S}_\diamond)$. Each type of product graph has the same vertex set defined by $\mathcal{V}_\diamond = \mathcal{V} \times \mathcal{V}_\mathcal{F}$, with NF nodes. A node in $\mathcal{G}_\diamond$ is noted as v_{ij} and represents the node pair of nodes v_i in $\mathcal{G}$ and $v_{f,j}$ in $\mathcal{G}_\mathcal{F}$. Next, the different types of product graphs will be discussed in more detail.

KRONECKER PRODUCT

The Kronecker product graph is denoted by $\mathcal{G}_\otimes = \{\mathcal{V}_\otimes, \mathcal{E}_\otimes, \mathbf{S}_\otimes\}$, and the GSO is defined as,

$$\mathbf{S}_\otimes = \mathbf{S}_\mathcal{F} \otimes \mathbf{S}. \tag{2.19}$$

There is an edge between nodes v_{ij} and v_{kl} of $\mathcal{G}_\otimes$, if there is an edge between nodes v_i and v_k of $\mathcal{G}$ and between nodes v_{fj} and v_{fl} of $\mathcal{G}_\mathcal{F}$. The number of edges in $\mathcal{G}_\otimes$ is

given by $|\mathscr{E}_{\otimes}| = |\mathscr{E}||\mathscr{E}_{\mathscr{F}}|$. An example of the Kronecker product graph is given in Figure 2.5. Here, we see that in the Kronecker product graph does not retain the original edge structure of both graphs.

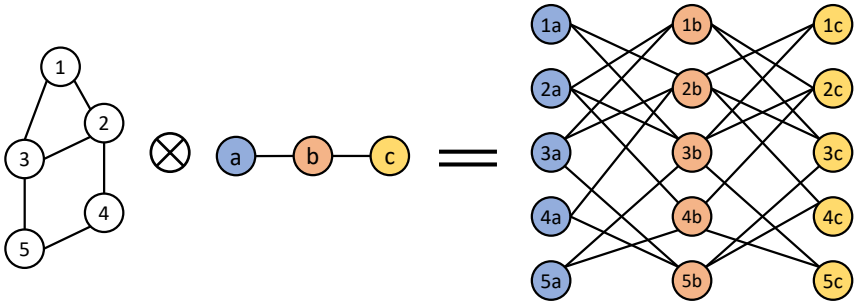


Figure 2.5: Kronecker product of two graphs

CARTESIAN PRODUCT

The Cartesian product graph is denoted as $\mathscr{G}_{\times} = \{\mathscr{V}_{\times}, \mathscr{E}_{\times}, \mathbf{S}_{\times}\}$, with the GSO defined as,

$$\mathbf{S}_{\times} = \mathbf{S}_{\mathscr{F}} \otimes \mathbf{I}_N + \mathbf{I}_F \otimes \mathbf{S}_N, \tag{2.20}$$

where $\mathbf{I}_N(\mathbf{I}_F)$ is the $N \times N(F \times F)$ identity matrix. In the following two cases there is an edge between nodes v_{ij} and v_{kl} of $\mathscr{G}_{\times}$: if $i = k$ and there is an edge between nodes $v_{f,j}$ and $v_{f,l}$ of $\mathscr{G}_{\mathscr{F}}$, or if $j = l$ and there is an edge between nodes v_i and v_k of $\mathscr{G}$. The number of edges in $\mathscr{G}_{\times}$ is given by $|\mathscr{E}_{\times}| = F|\mathscr{E}| + N|\mathscr{E}_{\mathscr{F}}|$. An example of the Cartesian product graph is given in Figure 2.6. Here we see that, unlike with the Kronecker product graph, the original structure of both graphs is retained in the Cartesian product graph.

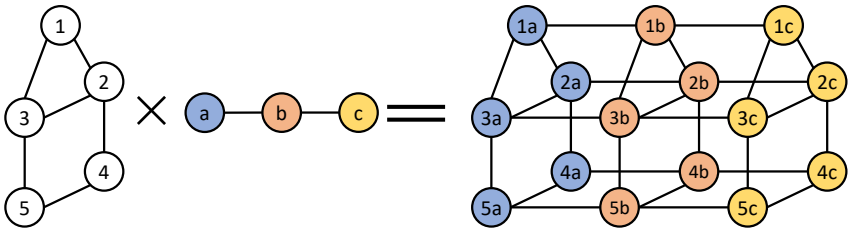


Figure 2.6: Cartesian product of two graphs

STRONG PRODUCT

The strong product graph is a combination of both the Cartesian and Kronecker product graph. The Strong graph product is denoted as $\mathscr{G}_{\boxtimes} = \{\mathscr{V}_{\boxtimes}, \mathscr{E}_{\boxtimes}, \mathbf{S}_{\boxtimes}\}$, with the GSO defined as

$$\mathbf{S}_{\boxtimes} = \mathbf{S}_{\otimes} + \mathbf{S}_{\times} = \mathbf{S}_{\mathscr{F}} \otimes \mathbf{S} + \mathbf{S}_{\mathscr{F}} \otimes \mathbf{I}_N + \mathbf{I}_F \otimes \mathbf{S}. \tag{2.21}$$

The number of edges in the strong product graph is the sum of the number of edges in the Cartesian and Kronecker product graph, $\left|\mathscr{E}_{\boxtimes}\right| = |\mathscr{E}||\mathscr{E}_F| + F|\mathscr{E}| + N|\mathscr{E}_F|$.

PARAMETRIC PRODUCT GRAPH

All of the above-mentioned product graphs can be described in a parametric product graph [21]. The GSO of the parametric product graph is noted by $\mathbf{S}_\diamond$, which is defined as

$$\mathbf{S}_\diamond = \sum_{i=0}^{1}\sum_{j=0}^{1} s_{ij}\left(\mathbf{S}_{\mathscr{F}}^i \otimes \mathbf{S}^j\right), \tag{2.22}$$

with $s_{ij} \in \{0,1\}$. By fitting s_{ij}, the optimal structure of the $\mathbf{S}_\diamond$ can be learned [21]. When $\mathbf{S}_\diamond$ is written out as,

$$\mathbf{S}_\diamond = s_{00}\left(\mathbf{I}_F \otimes \mathbf{I}_N\right) + s_{01}\left(\mathbf{I}_F \otimes \mathbf{S}\right) + s_{10}\left(\mathbf{S}_{\mathscr{F}} \otimes \mathbf{I}_N\right) + s_{11}\left(\mathbf{S}_{\mathscr{F}} \otimes \mathbf{S}\right), \tag{2.23}$$

it can easily been seen how the parametric product graph can represent all product graphs. When $s_{11} = 1$ and all other s_{ij} are zero the Kronecker graph product is represented. For $s_{01} = 1$ and $s_{10} = 1$ and all other s_{ij} are zero, the Cartesian graph product is represented. If all s_{ij} are equal to one, then the strong product graph is given with additional self loop on all nodes. The number of edges is in that case $|\mathscr{E}_\diamond| = N|\mathscr{E}_{\mathscr{F}}| + F|\mathscr{E}| + |\mathscr{E}||\mathscr{E}_{\mathscr{F}}| + NF$.

2.2.4. PRODUCT GRAPH FILTER

To represent a multi-dimensional graph signal we can extend the original graph by a graph that captures both the relationship between features on a node and between nodes. A suitable option to define such an extended graph is a product graph, which we discussed in the previous section. We can then apply the notion of graph filtering, as discussed Section 2.1.6, to this extended graph. Instead of the shift operator $\mathbf{S}$, defined by the graph $\mathscr{G}$, we use the GSO obtained by the product graph of graph $\mathscr{G}$ and the feature graph $\mathscr{G}_{\mathscr{F}}$. Dependent on the type of product graph used, we can apply $\mathbf{S}_\times$, $\mathbf{S}_\otimes$, $\mathbf{S}_\boxtimes$ or, $\mathbf{S}_\diamond$ in (2.10). A product graph filter using the parametric product graph is then written as

$$\mathbf{H}(\mathbf{S}_\diamond) = \sum_{k=0}^{K} h_k \mathbf{S}_\diamond^k = \sum_{k=0}^{K} h_k \left(\sum_{i,j=0}^{1} s_{ij}\left(\mathbf{S}_{\mathscr{F}}^i \otimes \mathbf{S}^j\right)\right)^k, \tag{2.24}$$

with h_k and s_{ij} are the graph filter coefficients and the parametric product graph coefficients, respectively. The output $\mathbf{y}$ defined by the filtering operation of the multi-dimensional graph signal $\mathbf{x} \in \mathbb{R}^{FN\times 1}$ with graph filter $\mathbf{H}(\mathbf{S}_\diamond)$, is given by

$$\mathbf{y} = \mathbf{H}(\mathbf{S}_\diamond)\,\mathbf{x}. \tag{2.25}$$

The product graph filter operation doesn't perform shifts over the graph $\mathscr{G}$ but instead performs shifts over the graph $\mathscr{G}_\diamond$. By applying the product graph filter to process multi-dimensional graph signals, the output signal corresponding to a feature takes now also into account values from neighboring features. This can be illustrated by Figure 2.7, where the neighborhood of a node in a product graph is shown.

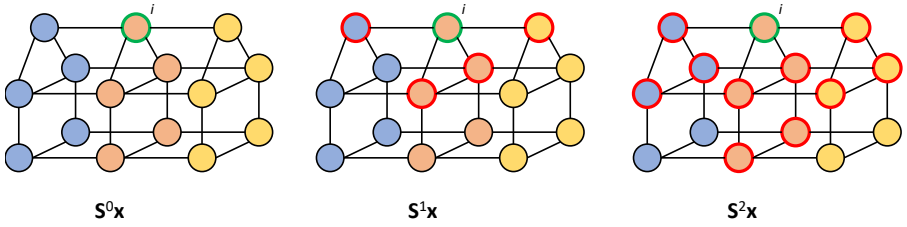


Figure 2.7: Illustration of the neighborhood of node i, with the green border, on the Cartesian product graph defined in Figure 2.6. The different colors of the nodes represent a different feature. The nodes, with the red circle border, are in the k-hop neighborhood of node i.

GENERALIZED PRODUCT GRAPH FILTER

The parametric product graph filter (2.24) can be extended to a more generalized version

$$\mathbf{H}(\mathbf{S}_\diamond) = \sum_{k=0}^{K} \sum_{l=0}^{L} h_{kl} \left(\mathbf{S}_{\mathscr{F}}^l \otimes \mathbf{S}^k \right), \tag{2.26}$$

where the generalized graph filter coefficients are defined by h_{kl} [21]. This filter embodies the product graph coefficients s_{ij} into the filter coefficients. We recall that the k^{th} power of $\mathbf{S}$ carries information of the k-hop neighborhood in $\mathscr{G}$, and the l^{th} power of $\mathbf{S}_{\mathscr{F}}$ carries information about the l-hop neighborhood in $\mathscr{G}_{\mathscr{F}}$. This means that if $k = 4$ and $l = 2$, and we consider feature f corresponding to node i, all feature values in the 2-hop neighborhood of f that belong to the nodes in the 4-hop neighborhood of node i are considered.

2.3. FORECASTING MODELS

2.3.1. VAR

A common model used to forecast multivariate time series is the vector autoregressive (VAR) model [31]. This model describes the evolution of a set of N variables over time. The variables are stored in a vector $\mathbf{x}_t \in \mathbb{R}^N$. The linear relationship between the multiple variables over time can be modeled by a matrix of trainable parameters, denoted by $A \in \mathbb{R}^{N \times N}$. The VAR model is described as follows

$$\mathbf{x}_t = -\sum_{p=1}^{P} \mathbf{A}_p \mathbf{x}_{t-p} + \varepsilon_t. \tag{2.27}$$

where P are the number of previous time steps taken into account, and the vector $\varepsilon_t \in \mathbb{R}^N$ is the error term. This model has in total PN^2 parameters to estimate. As the number of parameters scales quadratically for the number of variables, it can become difficult to estimate the parameters for large N.

2.3.2. GRAPH-VAR

When the relations between the variables in $\mathbf{A}_p$ can be described as connections of a network, GSP can be used to reduce the size of the number of parameters in the VAR

model. Let those relations be described by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{S}\}$. As most graphs are sparse, using the graph information we can already reduce the number of parameters to $P|\mathcal{E}|$. As proposed in [19] graph filters can be used to process the time-varying graph signal, instead of the matrix $\mathbf{A}_p$. Using the definition in (2.11) we can rewrite (2.27) to define the Graph-VAR (G-VAR) model.

$$\mathbf{x}_t = -\sum_{p=1}^{P} \mathbf{H}_p(\mathbf{S})\mathbf{x}_{t-p} = -\sum_{p=1}^{P}\sum_{k=0}^{K} h_{kp}\mathbf{S}^k\mathbf{x}_{t-p}. \tag{2.28}$$

The computation of $\mathbf{Sx}$ has complexity of $\mathcal{O}(|\mathcal{E}|)$. The power function $\mathbf{S}^k$ defines how often a shift is applied, where the $k-th$ power results in a computational complexity of $\mathcal{O}(K|\mathcal{E}|)$. The G-VAR model (2.28) has significantly fewer parameters to estimate than the normal VAR model, namely $P(K+1)$ filter coefficients. This results in a number of parameters that are independent of the number of nodes in the network.

2.3.3. PRODUCT GRAPH-BASED VAR

To model an F-dimensional graph process, one could process each graph process separately. However, if there is any type of relationship between those processes, this does not take them into account. Let us consider that there are relations between the processes and that those relationships can be captured by a graph. As discussed in Section 2.2, product graphs can be used to model multi-dimensional graph signals. To incorporate the relationship between features and nodes into a predictive model Natali et al. in [21] propose to employ product graph filters instead of a graph filter in the G-VAR model. This leads to the following models, where a distinction will be made between the used type of product graph. As we consider multi-dimensional graph processes, $\mathbf{x}_t \in \mathbb{R}^{NF\times 1}$ represents the multi-dimensional graph signal corresponding to time instant t.

PRODUCT G-VAR

The product graph-VAR (PG-VAR) model uses the product graph shift operator $\mathbf{S}_\diamond$, which is defined by the used type of product graph, i.e., the Kronecker, Cartesian or Strong product graph. This model can be written very similar to (2.28), as the main difference is the used GSO. The PG-VAR model is defined as,

$$\mathbf{x}_t = -\sum_{p=1}^{P} \mathbf{H}_p(\mathbf{S}_\diamond)\mathbf{x}_{t-p} = -\sum_{p=1}^{P}\sum_{k=0}^{K} h_{kp}(\mathbf{S}_\diamond)^k\mathbf{x}_{t-p}. \tag{2.29}$$

The foretasted values $\mathbf{x}_t$ of the PG-VAR model takes now also into account the relationships between features as this is captured by $\mathbf{S}_\diamond$. The amount of parameters is the same for all the three product graph types, PK, and its complexity ($\mathcal{O}(PK|\mathcal{E}_\diamond|)$) is type depended as $|\mathcal{E}\diamond|$ is different for each product type.

PARAMETRIC PRODUCT G-VAR

As explained in Section 2.2.3, the parametric product graph can be used when it is not known which product graph best models the graph process. This can be learned by estimating the parameters s_{ij}. The parametric product graph VAR (PPG-VAR) model is given

by

$$\mathbf{x}_t = -\sum_{p=1}^{P}\sum_{k=0}^{K} h_{kp}\left(\sum_{i,j=0}^{1} s_{ij}\left(\mathbf{S}^i \otimes \mathbf{S}_{\mathscr{F}}^j\right)\right)^k \mathbf{x}_{t-p}. \tag{2.30}$$

This model has the same computational complexity as the PG-VAR model, and also the same number of graph filter coefficients.

GENERALIZED PRODUCT G-VAR

For the last product graph-based VAR model, the generalized product graph filter, which is discussed in Section 2.2.4 is used. We present the generalized product graph VAR (GPG-VAR) model, which is defined as

$$\mathbf{x}_t = -\sum_{p=1}^{P}\sum_{k=0}^{K}\sum_{l=0}^{L} h_{klp}\left(\mathbf{S}^k \otimes \mathbf{S}_{\mathscr{F}}^l\right)\mathbf{x}_{t-p}. \tag{2.31}$$

This model has PKL parameters, which is a an increase compared to the previous discussed graph-based VAR models. The computational complexity of this model is $\mathscr{O}(P(KF|\mathscr{E}|+LN|\mathscr{E}_{\mathscr{F}}|)$, and if L<K this is smaller then the other product graph based models. To illustrate this, the computational complexity of the PG-VAR for the Cartesian product graph type is $\mathscr{O}(PK(F|\mathscr{E}|+N|\mathscr{E}_{\mathscr{F}})|)$.

3

GRAPH-VAR MODELS

This chapter describes our proposed models to model multi-dimensional graph processes. They extend the G-VAR model such that it accounts for the dependencies between features. In Section 3.1, we define the MIMO G-VAR model. Next, in Section 3.2 we propose two models that combine the G-VAR with the PG-VAR and GPG-VAR, respectively. Further, we propose in Section 3.4, two estimators to estimate the graph filter coefficients of the graph-based VAR models. Finally, Section 3.5 describes a method to jointly estimate the graph filter coefficients and feature GSO.

3.1. MIMO G-VAR MODEL

The PG-VAR model makes use of a feature graph to forecast the evolution of multi-dimensional graph signals. The use of this feature graph can be very constrained, as it has the same degrees of freedom (DoF) a G-VAR has to predict one feature. Also, the PG-VAR model relies heavily on the feature graph, which models the relationship between the features. Therefore, we here propose a model that learns the influence features have on each other in a flexible way.

We will first consider the G-VAR model, which does not take into account the effect features have on each other. To forecast an F-dimensional graph signal we can use F separate G-VARs. If the orders of P and K are the same for each feature, this can be written as

$$\mathbf{x}_t = -\sum_{p=1}^{P}\sum_{k=0}^{K-1}\left(\begin{bmatrix} h_{kp}^{(1)} & & \\ & \ddots & \\ & & h_{kp}^{(F)} \end{bmatrix} \otimes \mathbf{S}^k\right)\mathbf{x}_{t-p}, \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^{NF\times 1}$ is a multi-dimensional graph signal as defined in (2.13), and $h_{kp}^{(f)}$ is the graph filter coefficient that corresponds to the f^{th} feature. 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 is changed into a full matrix

$$\mathbf{x}_t = -\sum_{p=1}^{P}\sum_{k=0}^{K-1}\left(\mathbf{H}_{kp}\otimes\mathbf{S}^k\right)\mathbf{x}_{t-p}, \tag{3.2}$$

where the matrices $\mathbf{H}_{kp} \in \mathbb{R}^{F\times F}$ contain the filter coefficients that belong to the F^2 G-VAR models. The multidimensional graph filtering operation in this model corresponds to the MIMO graph filter (2.17), and we will refer to 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 $\mathcal{O}(PKF^2|\mathcal{E}|)$.

To gain some more intuition about the MIMO G-VAR, some special cases will be discussed. First, we will look at the case $K = 1$, i.e. only $\mathbf{S}^0 = \mathbf{I}_N$ is considered. The model can then be rewritten as

$$\mathbf{x}_t = -\sum_{p=1}^{P}\left(\mathbf{H}_p\otimes\mathbf{I}_N\right)\mathbf{x}_{t-p}. \tag{3.3}$$

By reordering the F-dimensional graph signal $\mathbf{x}$ so the feature values corresponding to a node are stacked on top each other, $\tilde{\mathbf{x}} = \left[x^1(1),...,x^f(1),...,x^1(N),...,x^f(N)\right]^T$, this can also be written as

$$\mathbf{x}_t = -\sum_{p=1}^{P}\begin{bmatrix}\mathbf{H}_p & & \\ & \ddots & \\ & & \mathbf{H}_p\end{bmatrix}\tilde{\mathbf{x}}_{t-p}. \tag{3.4}$$

From this, one can see that, for $K = 1$, the MIMO G-VAR is equal to N VAR models applied on the feature values of each node, where the matrices of filter coefficients of each VAR are the same. For higher orders of K, the shifted graph signals are also taken into account.

Further, we can consider the case where the matrix $\mathbf{H}_{kp}$ is a linear scaled version of $\mathbf{S}_{\mathscr{F}}^k$, i.e., the model can be written as

$$\mathbf{x}_t = -\sum_{p=1}^{P}\sum_{k=0}^{K-1}\left(\left(h_{kp}\mathbf{S}_{\mathscr{F}}^k\right)\otimes\mathbf{S}^k\right)\mathbf{x}_{t-p}. \tag{3.5}$$

This shows, that under specific conditions, the Kronecker PG-VAR model is equal to the MIMO G-VAR model.

3.2. COMBINED (G)PG-VAR AND G-VAR MODEL

To forecast multi-dimensional processes over graphs Natali et al. [21] proposed the use of product graphs in graph-based VAR models, leading to the PG-VAR model (2.29) and the GPG-VAR model (2.31). The number of parameters of these models are independent of the number of nodes in both, the graph $\mathscr{G}$ and feature graph $\mathscr{G}_{\mathscr{F}}$. In the case that $k=0$, and for the GPG-VAR also $l=0$ only information of prior values of the feature in $\mathscr{G}_{\mathscr{F}}$ and node in $\mathscr{G}$ itself are taken into account. This can be clearly seen when these terms are written outside of the summation. For example, the PPG-VAR is then written as

$$\begin{aligned}\mathbf{x}_t &= -\left(\sum_{p=1}^{P} h_{0p}\left(\mathbf{S}_F^0\otimes\mathbf{S}^0\right)+\sum_{p=1}^{P}\sum_{k=1}^{K-1}h_{kp}\left(\sum_{i,j=0}^{1}s_{ij}\left(\mathbf{S}_F^i\otimes\mathbf{S}^j\right)\right)^k\right)\mathbf{x}_{t-p}\\ &= -\left(\sum_{p=1}^{P} h_{0p}\left(\mathbf{I}_F\otimes\mathbf{I}_N\right)+\sum_{p=1}^{P}\sum_{k=1}^{K-1}h_{kp}\left(\sum_{i,j=0}^{1}s_{ij}\left(\mathbf{S}_{\mathscr{F}}^i\otimes\mathbf{S}^j\right)\right)^k\right)\mathbf{x}_{t-p}.\end{aligned} \tag{3.6}$$

The summation at $k=0$ represents NF univariate autoregressive models, where all models are constrained to have the same set of parameters h_{0p}.

We now assume that in a multi-dimensional process each feature has similar time-varying characteristics, but that they differ between features. In that case, the parameters h_{0p} restrict the DoF of the product graph-based models too much to predict this kind of multi-dimensional process. Similar to the model defined in (3.1), F independent sets of parameters h_{0p} can be used to give the model more DoF. Again for the PPG-VAR, this can be denoted as

$$\mathbf{x}_t = -\left(\sum_{p=1}^{P}\left(\begin{bmatrix} h_{0p}^{(1)} & & \\ & \ddots & \\ & & h_{0p}^{(F)}\end{bmatrix}\otimes\mathbf{I}_N\right)+\sum_{p=1}^{P}\sum_{k=1}^{K-1}h_{kp}\left(\sum_{i,j=0}^{1}s_{ij}\left(\mathbf{S}_{\mathscr{F}}^i\otimes\mathbf{S}^j\right)\right)^k\right)\mathbf{x}_{t-p}, \tag{3.7}$$

where $h_{0p}^{(f)}$ represent the set of parameters h_{0p} corresponding to the f^{th} feature. This model combines a G-VAR for each feature, with graph filter order $K=1$, with a PPG-VAR model, where the graph filter order of zero is not included. To even further extend the DoF we can generalize the above model to a combination of the G-VAR for each feature and the PG-VAR or GPG-VAR. We will refer to the combined PG-VAR and G-VAR as the

PG-G-VAR model, which is defined as

$$\mathbf{x}_t = -\left(\sum_{p=1}^{P} \sum_{k=0}^{K-1} \left(\begin{bmatrix} h_{kp}^{(1)} & & \\ & \ddots & \\ & & h_{kp}^{(F)} \end{bmatrix} \otimes \mathbf{S}^k \right) + \sum_{p=1}^{P} \sum_{k=0}^{K-1} h_{kp}^{(0)} \mathbf{S}_{\diamond}^k \right) \mathbf{x}_{t-p}, \tag{3.8}$$

and the combined GPG-VAR and G-VAR is referred to as the GPG-G-VAR, which is written as

$$\mathbf{x}_t = -\left(\sum_{p=1}^{P} \sum_{k=0}^{K-1} \left(\begin{bmatrix} h_{kp}^{(1)} & & \\ & \ddots & \\ & & h_{kp}^{(F)} \end{bmatrix} \otimes \mathbf{S}^k \right) + \sum_{p=1}^{P} \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} h_{klp} \left(\mathbf{S}_{\mathscr{F}}^k \otimes \mathbf{S}^l \right) \right) \mathbf{x}_{t-p}. \tag{3.9}$$

Compared to a G-VAR per feature, the number of parameters is increased only by a small amount. 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.

3.3. OVERVIEW OF GRAPH-BASED VAR MODELS

In the previous sections the MIMO G-VAR, PG-G-VAR, and GPG-G-VAR are presented and in Section 2.3.3 the different multi-dimensional graph VARs using product graphs are discussed. The difference in those graph VAR methods is in the way they model the connections between the graph and the multiple features on it. This results in different multi-dimensional graph filters $\mathbf{H}$. Whatever the specific type of VAR, the basic structure of a graph-based VAR is given by,

$$\mathbf{x}_t = -\sum_{p=1}^{P} \mathbf{H}_p \mathbf{x}_{t-p} + \boldsymbol{\epsilon}_t, \tag{3.10}$$

where $\mathbf{x}$ is the graph signal of a single feature in the case of the G-VAR and in the other cases $\mathbf{x}$ is the multi-dimensional graph signal, as defined in (2.13). An overview of the graph filters that are being used in the different graph VAR models, is given in Table 3.1.

3.4. GRAPH FILTER ESTIMATION

In this section two methods to find the graph filter coefficients $h_{k(l)p}$ will be discussed. These methods fit the model on available F-dimensional graph data. An advantage of the proposed methods is that they have a closed-form solution. Further, we will exploit the fact that the graph-based VAR models are all linear with respect to the graph filter coefficients.

3.4.1. MULTIVARIATE LEAST SQUARES ESTIMATOR

The first method is the multivariate least squares (LS) estimator, which is one of the most used methods to estimate VAR coefficients [31, 32, 33]. We assume that, for each variable

G-VAR	$\mathbf{H}_p = \sum_{k=0}^{K-1} h_{kp}\mathbf{S}^k$
PG-VAR	$\mathbf{H}_p = \sum_{k=0}^{K-1} h_{kp}\mathbf{S}_{\diamondsuit}^k$
PPG-VAR	$\mathbf{H}_p = \sum_{k=0}^{K-1} h_{kp}\left(\sum_{i=0}^{1}\sum_{j=0}^{1} s_{ij}\left(\mathbf{S}_{\mathscr{F}}^i \otimes \mathbf{S}^j\right)\right)^k$
GPG-VAR	$\mathbf{H}_p = \sum_{k=0}^{K-1}\sum_{l=0}^{L} h_{klp}\left(\mathbf{S}_{\mathscr{F}}^k \otimes \mathbf{S}^l\right)$
PG-G-VAR	$\mathbf{H}_p = \sum_{k=0}^{K-1}\left(\begin{bmatrix} h_{kp}^{(1)} & & \\ & \ddots & \\ & & h_{kp}^{(F)} \end{bmatrix} \otimes \mathbf{S}^k\right) + \sum_{k=0}^{K-1} h_{kp}^{(0)}\mathbf{S}_{\diamond}^k$
GPG-G-VAR	$\mathbf{H}_p = \sum_{k=0}^{K-1}\left(\begin{bmatrix} h_{kp}^{(1)} & & \\ & \ddots & \\ & & h_{kp}^{(F)} \end{bmatrix} \otimes \mathbf{S}^k\right) + \sum_{k=0}^{K-1}\sum_{l=0}^{L-1} h_{klp}\left(\mathbf{S}_{\mathscr{F}}^k \otimes \mathbf{S}^l\right)$
MIMO G-VAR	$\mathbf{H}_p = \sum_{k=0}^{K-1}\left(\mathbf{H}_{kp} \otimes \mathbf{S}^k\right)$

Table 3.1: Different types of graph filters, where the first model is the one-dimensional graph VAR model (G-VAR). The rest of the presented filters are multi-dimensional product graph-based filters. Presented from the least number of graph filter coefficients to the model with the most degrees of freedom, are the product graph VAR (PG-VAR), parametric product graph VAR (PPG-VAR), generalized product graph VAR (GPG-VAR), combined product graph and graph based VAR (PG-G-VAR), combined generalized product graph and graph VAR (GPG-G-VAR) and the multiple-input-multiple-output graph VAR (MIMO G-VAR), respectively.

in the graph, there are $T+p$ samples available. Let $\mathbf{h}$ represents the vector that contain the unknown filter coefficients $h_{k(l)p}$. The least-squares estimator to find the filter coefficients that bests fit the data is given by the argument that minimizes the sum of squared errors,i.e.,

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h}} \sum_{j}^{T} \left\| \mathbf{x}_j + \sum_{p=1}^{P} \mathbf{H}_p(\mathbf{S})\mathbf{x}_{j-p} \right\|_2^2. \tag{3.11}$$

This minimization has a closed form solution. For the G-VAR the closed form solution can easily be formulated if the model is rewritten into a matrix-vector product. The summations over P and K, can be written out as,

$$\begin{aligned} \sum_{p=1}^{P}\sum_{k=0}^{K} h_{kp}\mathbf{S}^k\mathbf{x}_{t-p} &= \sum_{k=0}^{K} \mathbf{S}^k[\mathbf{x}_{t-1},\cdots,\mathbf{x}_{t-p}] \begin{bmatrix} h_{k1} \\ \vdots \\ h_{kp} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{S}^0[\mathbf{x}_{t-1},\cdots,\mathbf{x}_{t-p}] & \cdots & \mathbf{S}^K[\mathbf{x}_{t-1},\cdots,\mathbf{x}_{t-p}] \end{bmatrix} \mathbf{h}, \end{aligned} \tag{3.12}$$

where $\mathbf{h} = [h_{01}\dots h_{0P}\dots h_{K1}\dots h_{KP}]^T$. In this way, the output of the G-VAR is given by a linear equation of the filter coefficients with the shifted graph signals. Using all $T+p$

samples, we can define

$$\mathbf{A} = \begin{bmatrix} \left(\mathbf{I}_T \otimes \mathbf{S}^0\right) \begin{bmatrix} \mathbf{x}_{P-1} & \cdots & \mathbf{x}_0 \\ \vdots & \dots & \vdots \\ \mathbf{x}_{T+P-1} & \cdots & \mathbf{x}_T \end{bmatrix} & \cdots & \left(\mathbf{I}_T \otimes \mathbf{S}^K\right) \begin{bmatrix} \mathbf{x}_{P-1} & \cdots & \mathbf{x}_0 \\ \vdots & \dots & \vdots \\ \mathbf{x}_{T+P-1} & \cdots & \mathbf{x}_T \end{bmatrix} \end{bmatrix}, \tag{3.13}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{x}_{P-1} \\ \vdots \\ \mathbf{x}_{T+P} \end{bmatrix}$$

Using this notation we can write (3.11) as a classical linear least squares problem,

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h}} \|\mathbf{b} + \mathbf{A}\mathbf{h}\|_2^2. \tag{3.14}$$

Solving this problem leads to the ordinary least squares estimator, which is given by

$$\hat{\mathbf{h}} = -(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = -\mathbf{A}^\dagger\mathbf{b}. \tag{3.15}$$

As all other graph-based VAR models are also linearly dependent on the graph filter coefficients, we can estimate them in a similar way.

3.4.2. YULE-WALKER LEAST SQUARES ESTIMATOR

Another method to estimate a VAR model is the Yule-Walker approach [31]. For a VAR process the Yule-Walker equations are obtained by multiplying both sides of eq. (2.27) with $\mathbf{x}_{t-h}$ and taking the expectation of both sides; resulting for $h > 0$ in:

$$\mathbb{E}[\mathbf{x}_t\mathbf{x}_{t-h}^T] = \mathbb{E}[-\sum_{p=1}^{P} \mathbf{A}_p\mathbf{x}_{t-p}\mathbf{x}_{t-h}^T] \tag{3.16}$$

$$\mathbf{R}_h = -\sum_{p=1}^{P} \mathbf{A}_p\mathbf{R}_{h-p} = \begin{bmatrix} \mathbf{A}_1, \dots, \mathbf{A}_p \end{bmatrix} \begin{bmatrix} \mathbf{R}_{h-1} \\ \vdots \\ \mathbf{R}_{h-p} \end{bmatrix}, \tag{3.17}$$

where $\mathbf{R}_i = \mathbb{E}[\mathbf{x}_t\mathbf{x}_{t-i}^T]$ is the auto-correlation matrix, and i denotes the lag between the signals. Using the Yule-Walker equations, the LS estimator of the VAR model can be derived. This can be done by concatenating the yule-walker equations for $h = 1, ..., P$, resulting in

$$\begin{aligned} [\mathbf{R}_1, \dots, \mathbf{R}_P] &= -\begin{bmatrix} \mathbf{A}_1, \dots, \mathbf{A}_p \end{bmatrix} \begin{bmatrix} \mathbf{R}_0 & \cdots & \mathbf{R}_{P-1} \\ \vdots & \ddots & \vdots \\ \mathbf{R}_{-P+1} & \cdots & \mathbf{R}_0 \end{bmatrix} \\ &= -\mathbf{A}\mathbf{\Psi}, \end{aligned} \tag{3.18}$$

and by multiplying both sides with the inverse of $\mathbf{\Psi}$, which is a square matrix, leading to

$$\mathbf{A} = -\begin{bmatrix} \mathbf{R}_1, \dots, \mathbf{R}_p \end{bmatrix} \mathbf{\Psi}^{-1}. \tag{3.19}$$

When the auto-correlations are estimated by the available data this is equal to the LS estimator. Now, instead of estimating a classical VAR model, we apply this approach to the graph VAR models, as proposed by Isufi et al. in [19]. The graph filter can be substituted for the matrices $\mathbf{A}_p$ in (3.18), which for the G-VAR model is then defined as

$$[\mathbf{R}_1,\dots,\mathbf{R}_P] = -\left[\sum_{k=0}^{K} h_{k1}\mathbf{S}^k,\dots,\sum_{k=0}^{K} h_{kP}\mathbf{S}^k\right]\begin{bmatrix}\mathbf{R}_0 & \cdots & \mathbf{R}_{P-1}\\ \vdots & \ddots & \vdots \\ \mathbf{R}_{-P+1} & \cdots & \mathbf{R}_0\end{bmatrix}. \tag{3.20}$$

As the right-hand side of this equation is linearly dependent on the graph filter coefficients h_{kp}, we can write this as a matrix-vector multiplication. Let us define

$$\begin{aligned}
\boldsymbol{\beta} &= \left[\mathrm{vec}(\mathbf{R}_1)^T,\dots,\mathrm{vec}(\mathbf{R}_p)^T\right]^T,\\
\boldsymbol{\varphi}_i &= \left[\mathrm{vec}(\mathbf{R}_i),\cdots,\mathrm{vec}(\mathbf{R}_{P+i-1})\right],\\
\boldsymbol{\Gamma} &= \left[\left(\mathbf{I}_{NP}\otimes\mathbf{S}^0\right)\boldsymbol{\varphi}_0,\dots,\left(\mathbf{I}_{NP}\otimes\mathbf{S}^0\right)\boldsymbol{\varphi}_{-p+1},\dots,\left(\mathbf{I}_{NP}\otimes\mathbf{S}^k\right)\boldsymbol{\varphi}_0,..,\left(\mathbf{I}_{NP}\otimes\mathbf{S}^k\right)\boldsymbol{\varphi}_{-p+1}\right],\\
\mathbf{h} &= [h_{01}\dots h_{0p}\dots h_{k1}\dots h_{kp}]^T,
\end{aligned} \tag{3.21}$$

where vec$(\cdot)$ stands for the vectorization operation that transforms a matrix into a column vector. We can now write (3.20) as

$$\boldsymbol{\beta} = -\boldsymbol{\Gamma}\mathbf{h}. \tag{3.22}$$

This can't be solved by just multiplying with the inverse of $\boldsymbol{\Gamma}\in\mathbb{R}^{N^2P\times KP}$, as this matrix is not square. Instead, we estimate $\mathbf{h}$ in a least-squares sense. The LS estimator of the Yule-Walker equations is obtained by solving

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h}} \left\|\boldsymbol{\beta}+\boldsymbol{\Gamma}\mathbf{h}\right\|_2^2. \tag{3.23}$$

Finally, the solution of the Yule-Walker LS estimator is expressed as

$$\hat{\mathbf{h}} = -(\boldsymbol{\Gamma}^T\boldsymbol{\Gamma})^{-1}\boldsymbol{\Gamma}^T\boldsymbol{\beta} = -\boldsymbol{\Gamma}^{\dagger}\boldsymbol{\beta}. \tag{3.24}$$

In a similar way, this approach can be applied to the other graph-based VAR methods. Solving the Yule-Walker equations in (3.18) for the VAR model results into the LS estimator. When the same set of Yule-Walker equations are used to estimate the G-VAR parameters, this results in a different solution than given by LS estimator derived in Section 3.4.1. In the rest of the work, we will stick to the LS estimator.

3.5. JOINT ESTIMATION OF GRAPH FILTER COEFFICIENTS AND FEATURE GRAPH

We assume that the network structure that captures a graph process is known or can be built with available data. For example in a sensor network, graphs using spatial information have been proven to work well in forecasting [34, 18]. A simple method to build the feature graph is by determining if features are correlated, e.g., calculating the Pearson

correlation coefficients, and connecting features with a high correlation [35, 36]. This method is useful in finding out if there is a relation between features, but it does not necessarily capture the influence each feature has in predicting connected feature values.

To find the feature graph that best suits the graph-based VAR model, we propose to identify the weights of the GSO by minimizing the forecast error. As the optimal graph filter coefficients are dependent on the used feature GSO we jointly estimate them. For a Kronecker PG-VAR this joint problem is defined as,

$$\{\hat{\mathbf{h}}, \hat{\mathbf{S}}_{\mathscr{F}}\} = \underset{\mathbf{h}, \mathbf{S}_{\mathscr{F}}}{\operatorname{argmin}} \quad \sum_{t=1}^{T} \left\| \mathbf{x}_t - \sum_{p=1}^{P} \sum_{k=0}^{K} h_{kp} (\mathbf{S}_{\mathscr{F}} \otimes \mathbf{S})^k \mathbf{x}_{t-p} \right\|_2^2. \tag{3.25}$$

This is a non-convex problem, due to the polynomials of the feature GSO. As we don't put any constraints on $\mathbf{S}_{\mathscr{F}}$, the estimated weights represent the adjacency matrix of a directed graph including possible self-loops.

To reduce the complexity of computing the joint problem, we follow Natali et al. [37] where $\mathbf{h}$ and $\mathbf{S}_{\mathscr{F}}$ are estimated iteratively using an alternating minimization approach. The pseudo-code that describes our AM approach is given in Algorithm 1. In this algorithm, we use the estimate of $\mathbf{S}_{\mathscr{F}}$ at the $(n-1)$th iteration to find the vector of parameters $\mathbf{h}$, which is obtained by solving

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \quad \sum_{t=1}^{T} \left\| \mathbf{x}_t - \sum_{p=1}^{P} \sum_{k=0}^{K} h_{kp} (\mathbf{S}_{\mathscr{F}} \otimes \mathbf{S})^k \mathbf{x}_{t-p} \right\|_2^2. \tag{3.26}$$

As shown in Section 3.4 this is a linear least-squares problem, which has a closed-form solution as defined in (3.15). We then use the estimated graph-based VAR parameters at the n^{th} iteration to estimate $\mathbf{S}\mathscr{F}$. This estimate is found by minimizing the original objective function with respect to the feature GSO

$$\hat{\mathbf{S}}_{\mathscr{F}} = \underset{\mathbf{S}_{\mathscr{F}}}{\operatorname{argmin}} \quad \sum_{t=1}^{T} \left\| \mathbf{x}_t - \sum_{p=1}^{P} \sum_{k=0}^{K} h_{kp} (\mathbf{S}_{\mathscr{F}} \otimes \mathbf{S})^k \mathbf{x}_{t-p} \right\|_2^2. \tag{3.27}$$

This problem is still non-convex, but it's a lighter problem than (3.25). A non-convex method can be used to obtain $\hat{\mathbf{S}}_{\mathscr{F}}$, as the gradient is not hard to find, we use the sequential quadratic programming (SQP) method; see [37] for details.

Algorithm 1 Joint GF and feature GSO

Require: $S_{\mathscr{F}}^{(0)}, \epsilon > 0$

$n \leftarrow 1$

while not converged **do**

  $\mathbf{h}^{(n)} \leftarrow \arg\min_{\mathbf{h}} f\left(\boldsymbol{h}, \mathbf{S}_{\mathscr{F}}^{(n-1)}\right)$ ▷ See equation (3.26)

  $\mathbf{S}_{\mathscr{F}}^{(n)} \leftarrow \arg\min_{\mathbf{S}} f\left(\boldsymbol{h}^{(n)}, \mathbf{S}_{\mathscr{F}}\right)$ ▷ See equation (3.27)

  Check convergence $(\mathbf{h}^{(n)}, \mathbf{S}_{\mathscr{F}}^{(n)}, \epsilon)$

  $n \leftarrow n+1$

end while

return $\mathbf{h}^{(n)}, \mathbf{S}_{\mathscr{F}}^{(n)}$

4

Numerical Results: Synthetic Data

In this chapter, we evaluate the performance of the graph-based forecasting models discussed in the previous chapters. The evaluations are performed on synthetic multi-dimensional graph data. Section 4.1 discusses the specifics of how the synthetic datasets are generated. In Section 4.2, we present the performed experiments with their numerical results. Finally, we conclude this chapter with a discussion of the results in Section 4.3.

4.1. SYNTHETIC DATA GENERATION

The synthetic data is generated based on a (G)PG-VAR model. To do this we generate two random graphs. First, a random sensor graph $\mathcal{G}$, with N nodes, is constructed using the GSP toolbox [38]. Secondly, a feature graph $\mathcal{G}_{\mathcal{F}}$, with F features, is constructed. The weighted adjacency matrix of this feature graph is generated from a random symmetric matrix, with $\lceil 0.4F^2 \rceil$ non-zero elements, and we set the elements on the diagonal to zero to avoid self-loops. As we use a random symmetric matrix, not all features in the graph might be connected; in that case, a new matrix is generated until we have a connected feature graph. We define the GSO of each graph as the graph Laplacian normalized by its maximum eigenvalue, i.e., $\mathbf{S} = \frac{1}{\lambda_{max}}\mathbf{L}$.

To be sure that the generated signals are stationary, the used model needs to be stable [31]. A VAR model is stable if the eigenvalues of the first-order model have a modulus of less than one [39]. This also holds for graph VAR models, as these are a particular case of the VAR model. All graph VAR models of order P can be rewritten into a model of order 1

$$\begin{pmatrix} \mathbf{x}_t \\ \mathbf{x}_{t-1} \\ \vdots \\ \mathbf{x}_{t-p+1} \end{pmatrix} = \mathbf{A} \begin{pmatrix} \mathbf{x}_{t-1} \\ \mathbf{x}_{t-2} \\ \vdots \\ \mathbf{x}_{t-p} \end{pmatrix} = \begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 & \cdots & \mathbf{H}_p \\ \mathbf{I} & 0 & \cdots & 0 \\ & & & \\ 0 & \ddots & 0 & \vdots \\ \mathbf{0} & 0 & \mathbf{I} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x}_{t-1} \\ \mathbf{x}_{t-2} \\ \vdots \\ \mathbf{x}_{t-p} \end{pmatrix}. \tag{4.1}$$

Now the eigenvalues of $\mathbf{A}$ can be checked to determine if the model of order P is stable.

The graph filter coefficients h_{kp} are randomly generated from a normal distribution $\mathcal{N}(0,1)$, and are scaled such that the graph VAR is stable. The synthetic data samples are stored in a data matrix $\mathbf{X} = [x_1, \ldots, x_T]$, where the graph signals x_t are generated according to (3.10), with random initial samples and $\boldsymbol{\epsilon}_t$ is white Gaussian noise with $\mathcal{N}(\mathbf{0}, \mathbf{I})$. There are $T + 500$ graph signals generated, where the first 500 samples are discarded to reach steady state.

EVALUATION METRIC

To measure the performance we adopt the error metric as defined in [34], which is the average excess prediction risk (AEPR), and take its root.

$$\text{rAEPR} = \sqrt{\mathbb{E}\left[\frac{1}{NF}\left\|\mathbf{x}_t - \sum_{p=1}^{P} \widehat{\mathbf{H}}_p \mathbf{x}_{t-p}\right\|_2^2\right] - \mathbb{E}\left[\frac{1}{NF}\left\|\mathbf{x}_t - \sum_{p=1}^{P} \mathbf{H}_p \mathbf{x}_{t-p}\right\|_2^2\right]}, \tag{4.2}$$

where we recall that $\widehat{\mathbf{H}}$ contains the estimated model and $\mathbf{H}$ is the model used to generate the data. This metric measures the difference between the error in the prediction of $\mathbf{x}_t$ and the variance.

4.2. NUMERICAL RESULTS

4.2.1. ESTIMATORS

First, we compare the proposed estimators from Section 3.4, against the minimum mean square error (MMSE) estimator proposed [19]. Both, the accuracy and the computational time of the estimators are evaluated against the number of training samples. There

are $T = 2000$ data samples in the generated dataset, and they are generated according to a G-VAR model, i.e., there is just one feature, with $p = 2$ and $K = 2$. The number of nodes used to create graph $\mathcal{G}$ is set to $N = 100$. The results are averaged over 25 different generated datasets.

In Figure 4.1, one can see that the performance of the estimators increases when there is more training data. The amount of training data has the most influence on the YW-LS estimator. In contrast, the LS and MMSE estimators already have a low prediction error on a lower amount of training data. Further, the LS and MMSE have the same prediction error, which makes sense as they both minimize the same objective function. In terms of the computational time, there is a clear linear trend visible. The linear trend of the YW-LS and the MMSE are approximately the same, because of the computation of the auto-Pearson correlation matrix. When a more efficient algorithm for calculating the auto-Pearson correlation matrix is used, the computational time can be decreased. In the rest of the experiments, the LS estimator will be used, as this one has the best prediction error as well as the least computational time required.

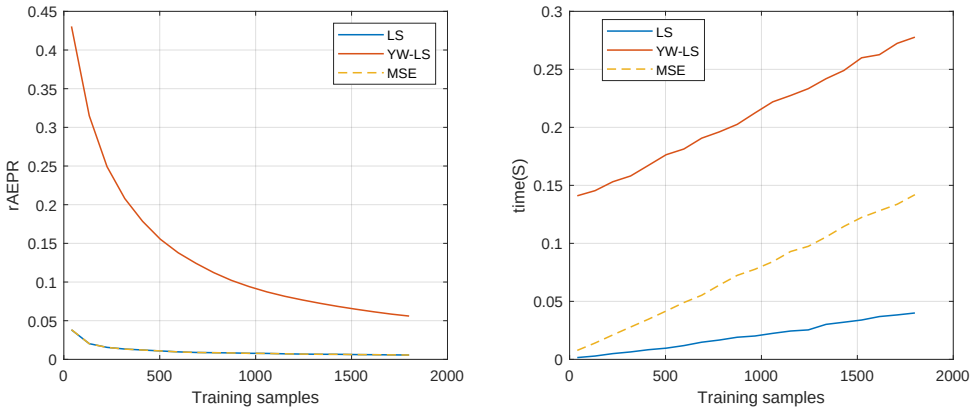


Figure 4.1: Prediction performance of the Least Squares (LS), Yule-Walker Least Squares (YW-LS), and Mean Square Error (MSE) estimator for the G-VAR model (left) and the computational time of each estimator (right) versus the number of training samples.

4.2.2. (G)PG-VAR

In this experiment, the prediction performance is evaluated for synthetic data generated with a Kronecker, Cartesian, Strong, and Generalized PG-VAR model. For each model, we compare it against all other described graph VAR models and also include the MIMO G-VAR model. The graphs $\mathcal{G}$ and $\mathcal{G}_{\mathscr{F}}$ are generated with $N = 50$ and $F = 10$ nodes, respectively. The orders of the graph filters are set to $P = 2$ and $K = 2$. For each dataset, there are $T = 2000$ samples generated. We independently generate 25 of such datasets and again take the averaged rAEPR of the obtained results, which are shown in Figure 4.2. The results show that the GPG-VAR model works well in all cases; only the model used to generate the data has a smaller prediction error. Further, one can see that the Kronecker product graphs don't perform well on the data that is generated through the other models, this is because the Kronecker product graph does not retain any informa-

tion about the edges of graphs $\mathcal{G}$ and $\mathcal{G}_{\mathcal{F}}$. The MIMO G-VAR model needs more training samples to achieve a lower prediction error, this is because the model does not have any prior information about the feature graph.

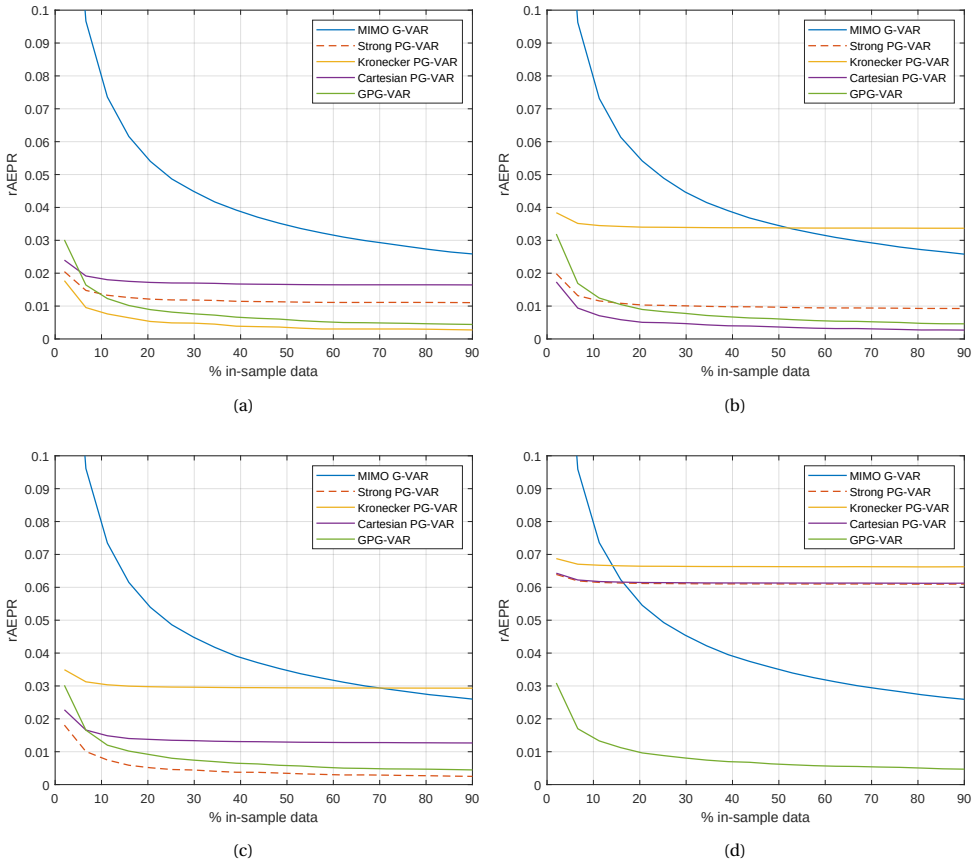


Figure 4.2: Performance comparison of the following models: PG-VAR with different product graph types, GPG-VAR, and the MIMO G-VAR. The synthetic data is generated according to (a) Kronecker PG-VAR model; (b) Cartesian PG-VAR model; (c) Strong PG-VAR model; (d) GPG-VAR model.

4.2.3. PARAMETRIC PRODUCT GRAPH VAR

We now test how effectively the PPG-VAR can identify which product graph is used to generate the synthetic data and compare its performance against the model used to generate the data as well as the GPG-VAR model. This is done by jointly estimating the filter coefficients h_{kp} and parameters s_{ij} that defines the type of product graph [cf. (2.30)]. The same experimental setup is used as in the previous experiment. At the beginning of the estimation process, parameters s_{ij} are randomly chosen from a uniform distribution between (0,1).

In Figure 4.3 the prediction error and the estimated parameters s_{ij} are shown; both results are averaged over 25 different datasets. We see that the prediction error does not smoothly decrease for the PPG-VAR. This is probably due to the fact that the problem is not convex and instead a local minimum is found. However, the prediction error remains almost constant, and for more in-sample data, it has a better performance than the other models. Further, the found parameters do not exactly represent the used product graph, but one can still distinguish which product graph is used.

4.2.4. JOINTLY ESTIMATING THE FEATURE GSO

We now analyze how the Kronecker, Cartesian, Strong, and Generalized PG-VAR models perform when their feature GSO is not known and estimated with the method described in Section 3.5. As the initial point for estimating the feature GSO the Pearson correlation matrix is used. We also compute the prediction performance when this initial point is used as feature GSO. Further, we compare these models against the MIMO G-VAR and G-VAR models. The number of nodes used in the graphs $\mathcal{G}$ and $\mathcal{G}_{\mathcal{F}}$ to generate the synthetic data is $N = 25$ and $F = 7$, respectively. Again, the orders of the graph filters are set to $P = 2$ and $K = 2$. There are 15 datasets generated, and each dataset contains $T = 2000$ samples.

The averaged rAEPR for each type of product graph is shown in Figure 4.4. There are a few interesting things to see in the results. First, when the Pearson correlation matrix is used as GSO, the G-VAR outperforms the model used to generate the data. The G-VAR model and GPG-VAR model, with the Pearson correlation matrix as GSO, show a very similar performance. And lastly, the models where the feature GSO is estimated show a significant increase in the prediction performance.

4.2.5. COMBINED G-VAR AND (G)PG-VAR

In this experiment, the performance of the combined models, described in Section 3.2, is evaluated. The Pearson correlation matrix of the feature data is chosen as the GSO of the feature graph. This is done to simulate a more realistic scenario, as the feature graph is often unknown and has to be estimated. Those combined models are compared with the G-VAR, PG-VAR, GPG-VAR, and MIMO G-VAR models. Synthetic data is generated according to the Kronecker, Cartesian, Strong and Generalized model, and the product graph type used to generate the data is considered known. We use the same settings to generate the data as in the previous experiment.

The results shown in Figure 4.5show that the combined models outperform the (G)PG-VAR model and the G-VAR model. This is as expected as it incorporates both models into one model giving it more flexibility. Further, there is now almost no difference in their performance between the GPG-G-VAR and PG-G-VAR. At last, we notice that the MIMO G-VAR model outperforms all other models, as this learns the relationship between the features and is not fixed as is the case with the product graph-based models.

4.2.6. JOINTLY ESTIMATING THE FEATURE GSO: COMBINED G-VAR AND (G)PG-VAR

In the last experiment on synthetic data we look at the performance of the combined models, PG-G-VAR and GPG-G-VAR, in case we jointly estimate the filter coefficients and

GSO. The same setup is used as in the previous two tests. We consider the MIMO G-VAR, PG-G-VAR with the Pearson correlation matrix as GSO, PG-VAR, and GPG-VAR where the filter coefficients and GSO are also jointly estimated, to compare the combined models with.

In Figure 4.6 the results of this test are depicted. For the PG-VAR models the combined models outperform the model that is used to generate the data, but the GPG-VAR model out-performs them all. When the GPG-VAR is used to generate synthetic data, the combined GPG-G-VAR model eventually outperforms the GPG-VAR model. Further, all models that jointly estimate the GSO perform better than the MIMO G-VAR model.

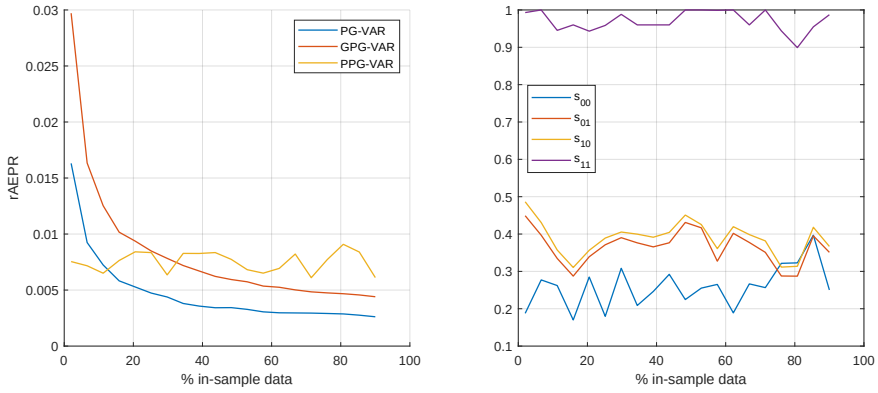
4.3. CONCLUSION

In this chapter, we evaluated the performance of the proposed graph-based models on synthetically generated datasets according to the different product graph-based models. In the first experiment, we showed the performance of the G-VAR for the two proposed estimators and the MSE estimator. From this, we concluded that the LS estimator is preferred as it has the best performance and least computational time. The slope of computational time of the YW-LS and MSE method is very similar, which suggests the increase of computational time is due to the computation of the auto-Pearson correlation matrix. Therefore, a more efficient computation of the auto-Pearson correlation matrix could change our view on which estimator is preferred.

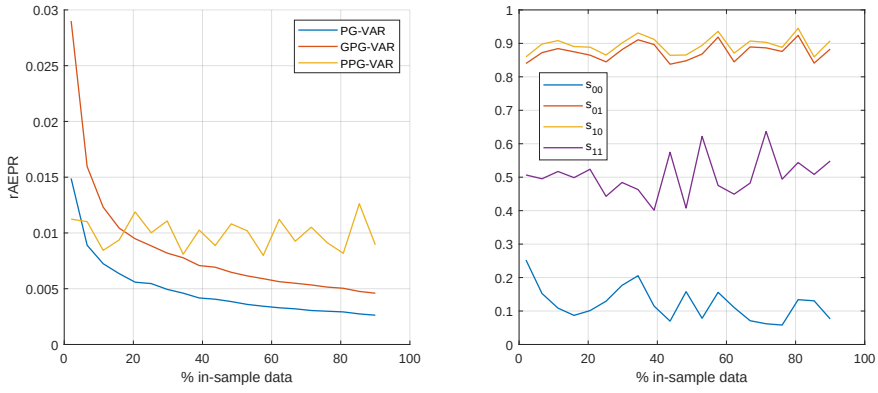
In the following experiment, we investigated how well the product graph-based models perform on data generated according to different types of product graphs. These results showed the effectiveness of the GPG-VAR at capturing all possible product graph types. Further, we showed that the PPG-VAR can also capture all types of product graphs, but the problem of estimating the parameters s_{ij} is non-convex and therefore more challenging. So we suggest using the GPG-VAR instead.

The next experiments focused on the scenario that feature GSO used to generate the data is unknown. First, the Pearson correlation matrix was used as feature GSO to capture the relationship between features. This caused the performance to degrade in such a way that the PG-VAR was outperformed by the G-VAR and the GPG-VAR performed similarly to the G-VAR, while the G-VAR does not take into account any information between features. The combined models did outperform the G-VAR, which shows their ability to incorporate some extra information about the features. The MIMO G-VAR illustrated its effectiveness to incorporate the dependencies between features as it outperformed all other models.

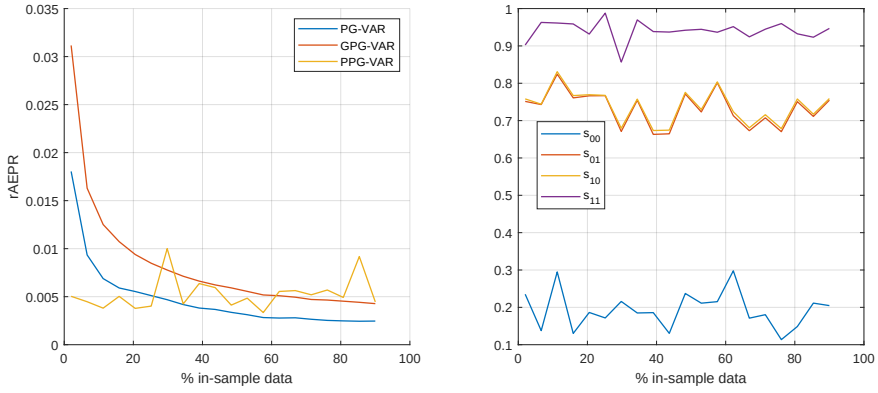
Finally, we experimented with an iterative method that jointly estimates the feature GSO and the graph filter coefficients. The obtained results showed that this method performs well, as the performance of all feature graph-based models improved. They even outperformed the MIMO G-VAR model, with an exception for the Cartesian and strong PG-VAR. Overall, these experiments have shown the importance of estimating the relationship between features and the benefits our proposed models, together with the estimation methods hold.



(a)



(b)



(c)

Figure 4.3: Shown on the left, the prediction error of the PPG-VAR compared to the PG-VAR type used to generate the synthetic data and the GPG-VAR, for data generated according to (a) Kronecker PG-VAR model; (b) Cartesian PG-VAR model; (c) Strong PG-VAR model. The average estimated values of the parameters s_{ij} are depicted on the right.

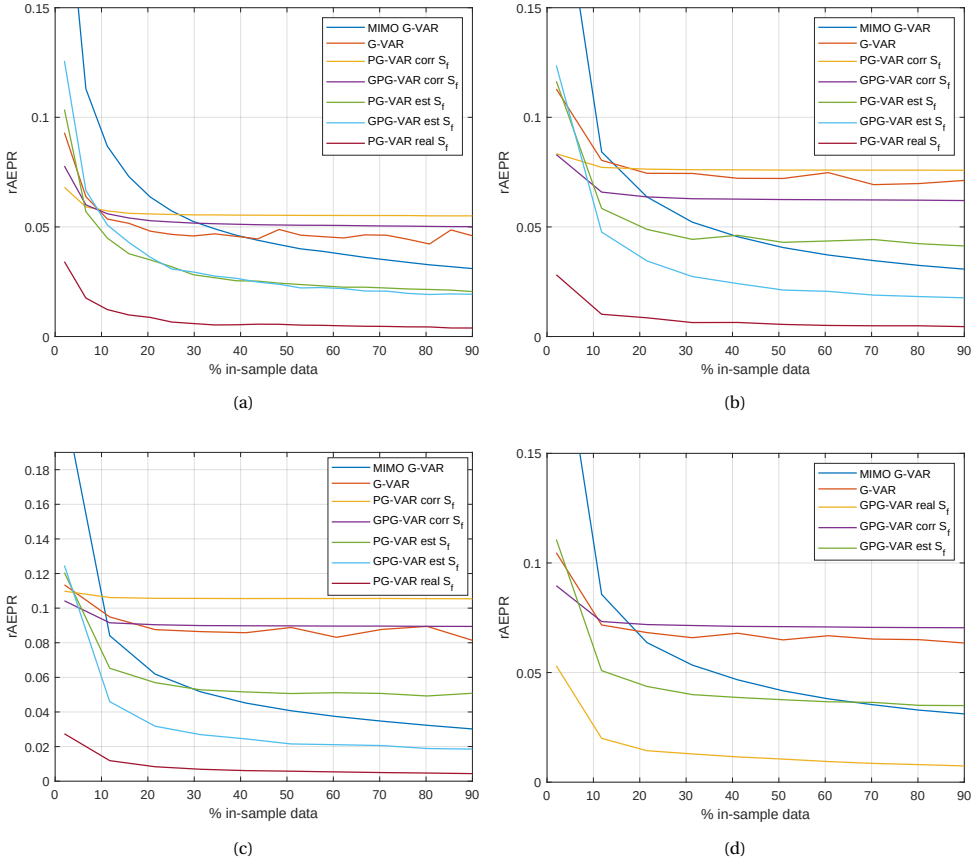


Figure 4.4: The rAEPR versus the amount of training data available. The product graph type to generate the synthetic data is also the product graph type used in the PG-VAR model. In the *real* S_f model the feature GSO that is used to generate the data is given, in the case of *corr* S_f and *est* S_f Pearson correlation matrix and the estimated GSO are used, respectively

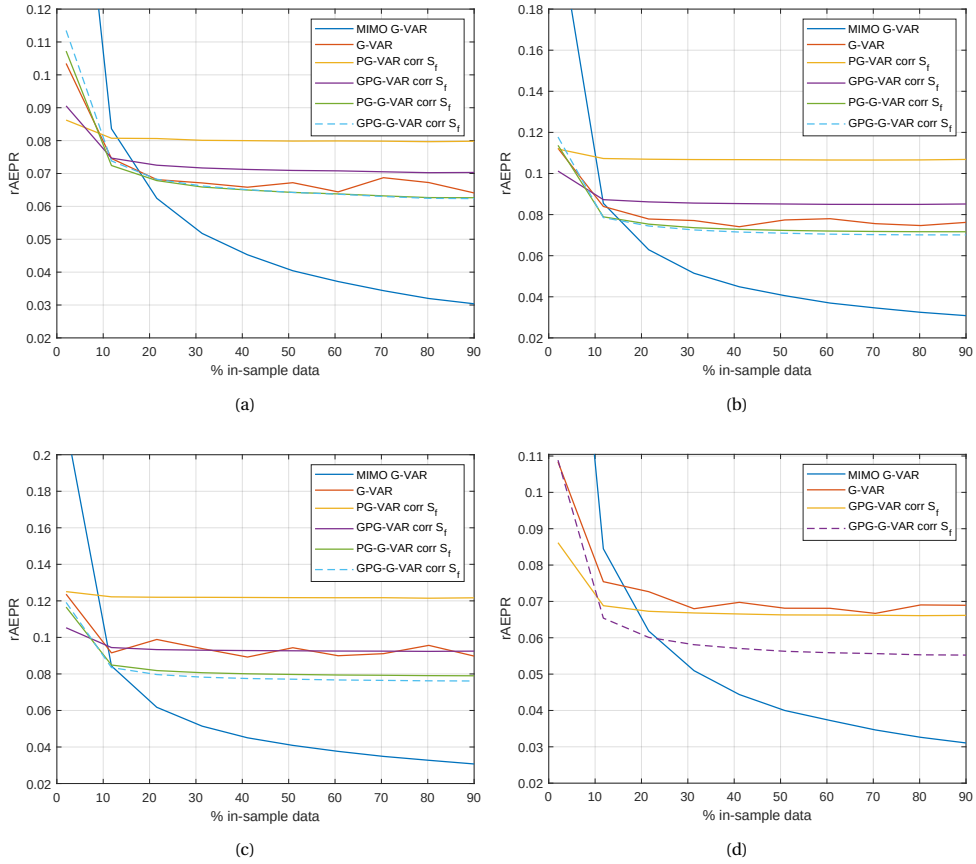


Figure 4.5: The rAEPR of the combined models and all other described graph-based VAR models versus the percentage of in-sample training data, for data generated according to (a) Kronecker PG-VAR model; (b) Cartesian PG-VAR model; (c) Strong PG-VAR model; (d) GPG-VAR model. The product graph type used to estimate is the same as the one used to generate the data. In the case of *corr* S_f the Pearson correlation matrix is used as feature GSO.

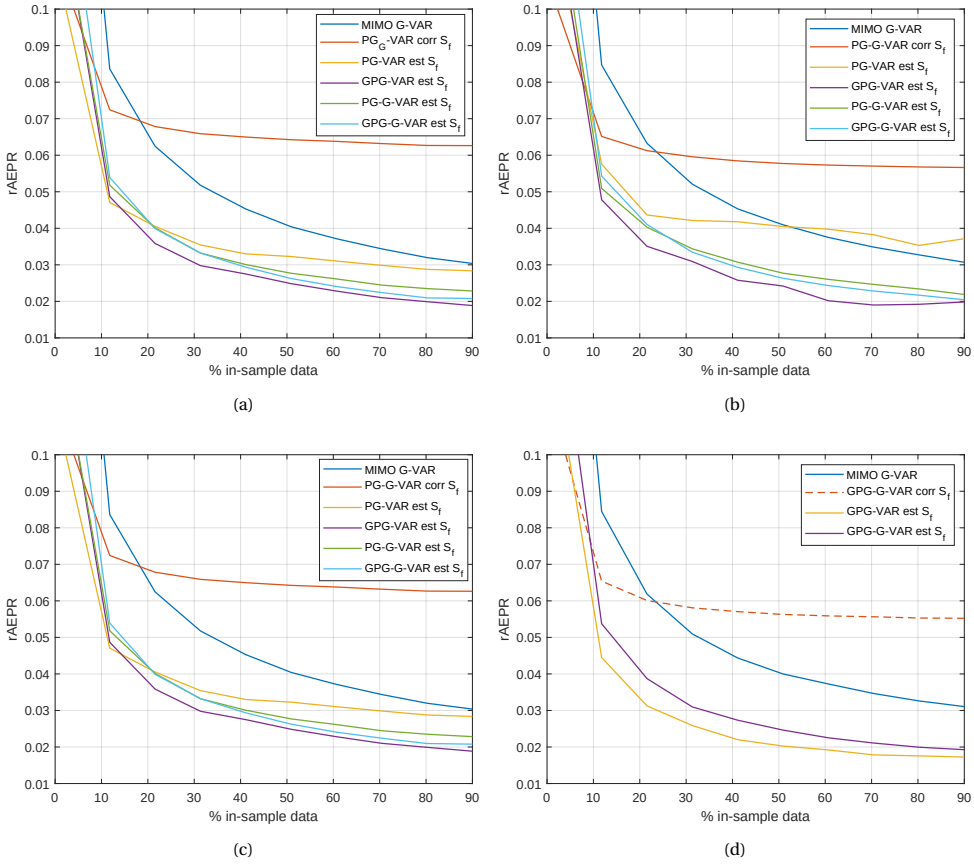


Figure 4.6: The rAEPR of the combined models and all other described graph-based VAR models versus the amount of training data available, for data generated according to (a) Kronecker PG-VAR model; (b) Cartesian PG-VAR model; (c) Strong PG-VAR model; (d) GPG-VAR model. The product graph type used to estimate is the same as the one used to generate the data. In the case of $corr\,S_f$ and $est\,S_f$ the Pearson correlation matrix and jointly estimated feature GSO are used, respectively.

5

Numerical Results: Real-World Data

In this chapter, we evaluate the performance of the graph-based forecasting models on two real-world datasets. Section 5.1 discusses the specifics of both datasets. In Section 5.2, we present the experimental setup used to evaluate the performance of each model. The numerical results are provided in Section 5.3. The concluding remarks of this chapter are presented in Section 5.4.

5.1. DATASETS

5.1.1. WEATHER DATA

The first dataset of real-life measurements comes from Ireland's National Meteorological Service, Met Éireann [40]. It consists of hourly measurements from $N = 25$ weather stations across Ireland from the year 1850 till 2010. At each station, there are measurements of $F = 5$ different features. The type of feature values measured at each station is temperature, wet-bulb temperature, dew point, vapor, and relative humidity. The hourly measurements of each feature at a specific station are shown in Figure 5.1. In the time series, there is a clear seasonal trend visible. Due to missing data, we only consider the samples measured in the last year, which results in $T = 8760$ hourly measurements.

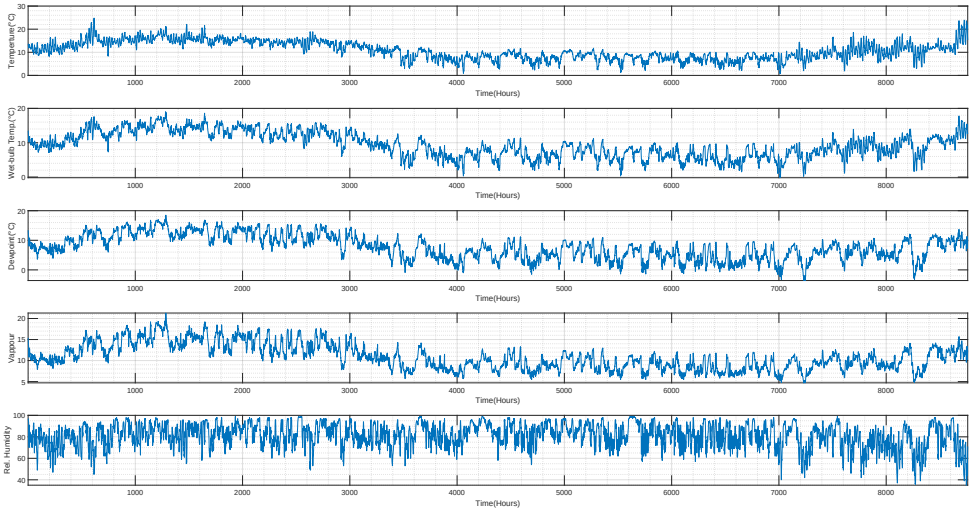


Figure 5.1: Example of the five different time series measured at station number 13, located near Mace Head.

The station graph $\mathcal{G}$ is constructed using a 7−NN method based on the geographical distances, and this graph is shown in Figure 5.2. As in [41, 19], the edge weights w_{ij} are defined by a Gaussian kernel weighting function

$$w_{ij} = \begin{cases} \exp\left(-\frac{d(i,j)^2}{\overline{d}}\right), & \text{if } (v_i, v_j) \in \mathcal{E} \\ 0 & \text{, otherwise} \end{cases}, \tag{5.1}$$

where $d(i,j)$ is the great-circle distance [42] between node v_i and v_j, $\overline{d}$ is the average distance between all stations. The Laplacian matrix, normalized by its maximum eigenvalue, is used as GSO.

There is no prior information on how the features are physically related to each other, but from the available data, these relationships can be learned. There are multiple possible ways to obtain a graph from measurement data, some common methods are given in [35, 36]. We decided to use the Pearson correlation matrix, where the diagonal is set to zero so there are no self-loops, as the adjacency matrix for the feature graph. This resem-

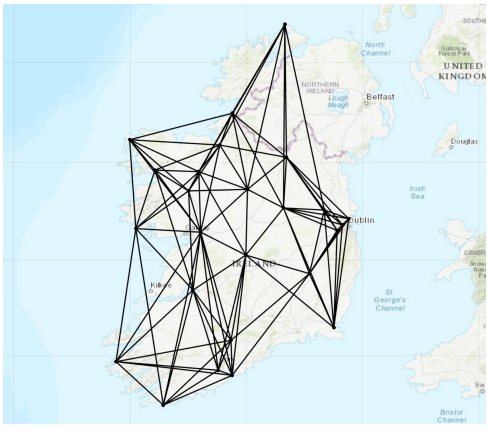


Figure 5.2: 7−NN Graph of the 25 weather stations present in the weather dataset.

bles a fully connected feature graph. As GSO of the feature graph, the graph Laplacian is used, and again we normalize this by its maximum eigenvalue.

5.1.2. AIR-QUALITY DATA

The Air-quality dataset consists of hourly measurements of 6 types of air pollutants (PM2.5, PM10, SO2, NO2, CO, O3) and 6 weather-related variables (temperature, pressure, dew point, rain, wind direction, and wind speed), and are recorded by $N = 12$ air-quality monitoring stations in and around Beijing [43, 44]. The data is recorded between the 1st of March 2013 and 28 February 2017. Due to gaps in the recorded data, we only considered the data from index number 20912 until index number 30830, which results in a set of $T = 9918$ hourly measurements. We don't take the wind direction and rain into account, so there are $F = 10$ features that we consider left. Figure 5.3 shows the considered data at a specific station. In the data, there are some clear seasonal trends visible, especially in the temperature and pressure data.

The spatial graph of the air-quality monitoring stations is constructed according to the $3-NN$ approach, which is discussed in the weather data section. The GSO is defined as the graph Laplacian, which is normalized by its maximum eigenvalue.

To construct the feature graph, a similar method as described in the weather section is used, i.e. we use the Pearson correlation matrix as the basis for the Adjacency matrix. We only keep an edge between feature i and j if this edge weight belongs to the four highest edge weights connected to feature i. Using this method we create a sparse feature graph that only takes into account the connections that have the most influence on each other. Further, there are no self-loops in the feature graph. As feature GSO the graph Laplacian is used, which is also normalized by the largest eigenvalue.

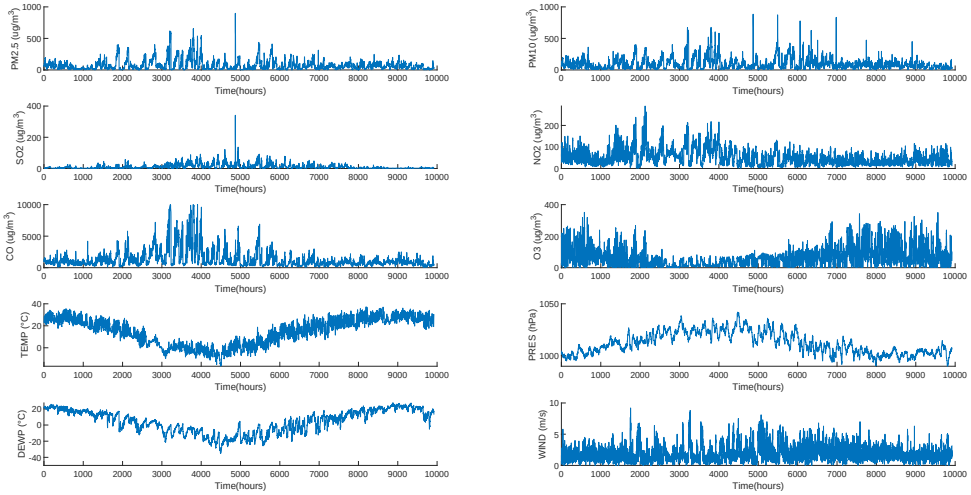


Figure 5.3: The measurements of the 10 different time series, measured at station number 1, located at the National Olympic Sports Center in Beijing.

5.2. EXPERIMENTAL SETUP

A Z-fold sliding window cross-validation setup is used to evaluate the models [45]. This is done as it best fits the real-life scenario, where a model is fitted on data of the previous day's and is used in forecasting the next day's. Further, this type of setup efficiently uses the available data to make a robust evaluation [46, 47]. In the sliding window cross-validation setup, the dataset is split into three parts, an in-sample, out-of-sample, and left-out dataset. The in-sample data is first used to find the optimal hyperparameters and secondly to estimate the filter coefficients. At every new fold, the in-sample and out-of-sample parts slide over the dataset, as shown in Figure 5.4, creating Z in-sample and out-of-sample datasets where the models can be trained and evaluated over. At each fold, the in-sample and out-sample parts are shifted, so we evaluate and train on new data.

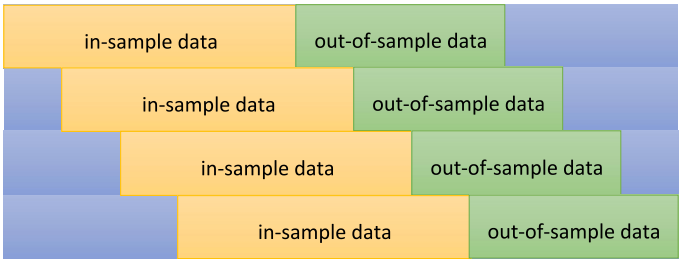


Figure 5.4: Example of how the dataset is split with sliding window cross-validation, with 4 iterations.

Finally, to measure the prediction performance, the averaged prediction error over the out-of-sample data of all Z-folds is taken. The error metric we take is the Root Nor-

malized Mean Square Error (RNMSE). The RNMSE is computed by comparing the true signals $\mathbf{x}_t$ with the predicted signals $\tilde{\mathbf{x}}_t$:

$$\text{RNMSE} = \sqrt{\frac{\sum_{t=1}^{\tau} \|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|_2^2}{\sum_{t=1}^{\tau} \|\mathbf{x}_t\|_2^2}} \tag{5.2}$$

To evaluate the influence of the training size on the prediction performance, the experiments are done with different amounts of samples in the in-sample dataset. To make a fair comparison between the different amount of training sizes, the same out-of-sample data for each segment is taken.

5.2.1. HYPERPARAMETERS

To find the optimal hyperparameters for each model, a grid search is performed [48]. The in-sample data is split 70%/30% into a training and validation set. The models are fitted on the training set with different values for the hyperparameters P, K and if applicable L. Their performance is evaluated on the validation set. We took the following sets of values into consideration: the number of previous signals P taken into account are $P \in \{1,...,5\}$, the order of shifts K are $K \in \{0,...,5\}$ and the order of L are $L \in \{0,...,3\}$. The hyperparameters that yield the lowest RNMSE on the validation set are used to refit the model on the in-sample data.

5.3. NUMERICAL RESULTS

We evaluate our proposed models, the MIMO G-VAR and the combined (G)PG-G-VAR models, and compare their performance with the other discussed models, the (G)PG-VAR, G-VAR, and the VAR model. The Cartesian graph product is considered as the product graph type that models the relations between the station graph and the feature graph.

For all datasets, the number of in-sample data samples varied from 200 to 2000 in steps of 200 samples, and the out-of-sample data consists of 168 hourly measurements, i.e. one week of data. The amount of folds used is 20, and at every fold, all data is shifted with the out-of-sample data size.

In this section only the errors of the models are reported. A visualization regarding the predicted values versus the true values is shown in Appendix A.

WEATHER DATA

Figure 5.5 shows the RNMSE for each model for different amounts of in-sample data. We see that all the graph-based models outperform the conventional VAR model, but also that the VAR model's prediction error decreases the most for more in-sample data. One can expect that, with enough in-sample data, eventually, the VAR will outperform the graph-based models. We further see that, contrary to the experiments done on synthetic data, the MIMO G-VAR has the lowest prediction error in the case there are more than 300 hourly measurements in the in-sample data. The PG-VAR has the worst performance of the graph-based models. Also, the GPG-VAR model has slightly worse performance than applying the G-VAR separately for each feature. As expected from the results on synthetic data, the combined (G)PG-G-VAR models perform better than the G-VAR model.

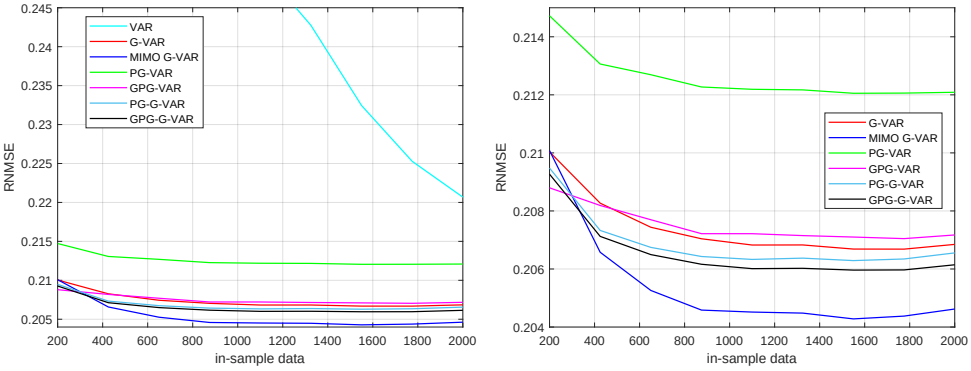


Figure 5.5: RNMSE of the different graph-based VAR models and the conventional VAR model versus the amount of in-sample data. (b) Zoomed in version of (a), where the VAR model is left out.

5

We also evaluate the effect that jointly estimating the filter coefficients and feature GSO, as described in Section 3.5, has on the prediction performance of the product graph-based VAR models. We compare those models with both, the G-VAR and MIMO G-VAR models.

In Figure 5.6 the results are depicted. These results clearly show that jointly estimating the filter coefficients and feature GSO improves the prediction performance. Still, the MIMO G-VAR model has the best performance, but now only after the in-sample data size becomes larger than 600 samples instead of 300 in the case of fixed GSO. The PG-G-VAR and GPG-G-VAR are approaching the MIMO G-VAR the closest, and both perform better than the MIMO G-VAR for low amounts of in-sample data.

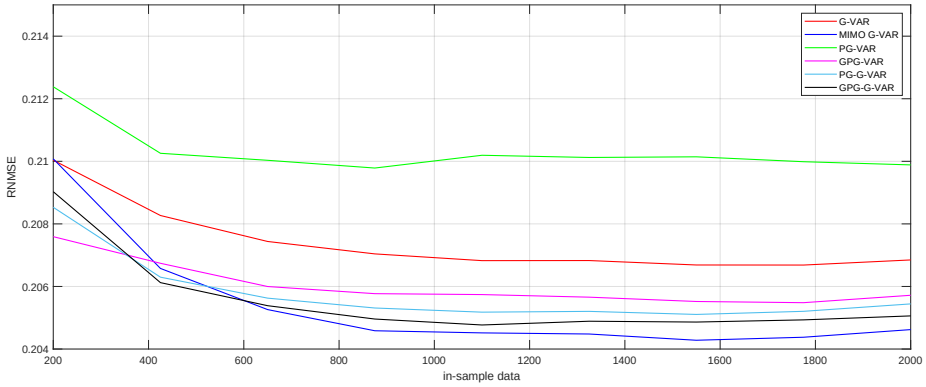


Figure 5.6: RNMSE versus the amount of in-sample data for the different graph-based models, where the feature graphs are jointly estimated with the graph filter coefficients.

AIR-QUALITY DATA

The performance of the different models is shown in Figure 5.7. The VAR model is not depicted in the results as this model did not lead to a stable predictor. We see similar results as with the 'weather' dataset. One significant difference is that GPG-VAR has a bigger performance gap to the G-VAR model than in the 'weather' data scenario, where their performance was almost equal to each other. Another difference is that the performance of all models, except the MIMO G-VAR, does not increase significantly for a larger amount of in-sample data.

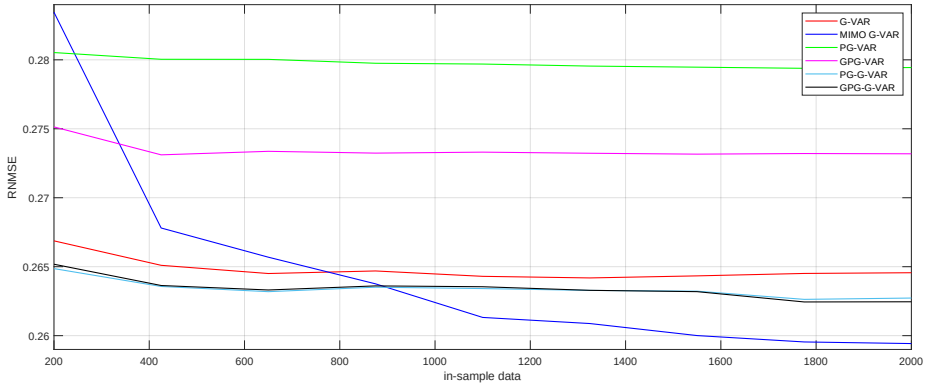


Figure 5.7: RNMSE versus the amount of in-sample data for the different graph-based models.

The performance of product graph-based models, where the feature GSO and filter coefficients are jointly estimated, is shown in Figure 5.8. For comparison, the results of the G-VAR and MIMO G-VAR models are also added. We now see that the performance of the product graph-based models increases when there is more in-sample data available. Only the GPG-VAR model does not improve for more than 400 in-sample data samples. Further, we see that for less than 1100 in-sample the PG-G-VAR model and GPG-G-VAR model outperform the MIMO G-VAR, and for more in-sample data their performances are approximately equal.

In the last experiment, we studied the impact of constraining the support feature GSO to be in the subset of the support of the initial given feature GSO. The used initial feature GSO is defined by the k-NN feature graph, e.g., each feature is connected to k features that are the most correlated to each other. In the previous experiments, the 4-NN feature graph is used. This is compared with the 2-NN and 9-NN feature graph, wherein the 9-NN case the graph is fully connected. The GSO of the 2-NN, 4-NN, and 9-NN feature graph have 100, 56, and 38 number of non-zero elements to estimate, respectively. The influence that these constraints have on the performance is shown in Figure 5.9. We see that the 9-NN graph eventually leads to a slightly better forecasting performance but on low in-sample data, the 2-NN and 4-NN perform slightly better. This suggests that connecting nodes that have the highest correlation adds the most important information to the models.

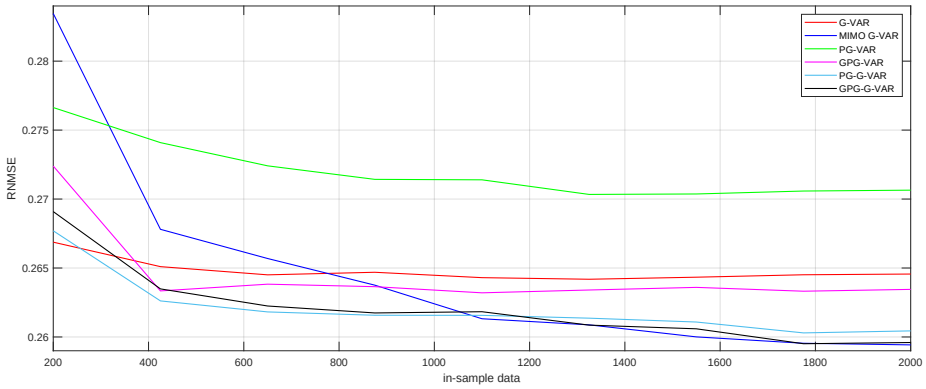


Figure 5.8: RNMSE versus the amount of in-sample data for the different graph-based models, where the feature graphs are jointly estimated with the graph filter coefficients.

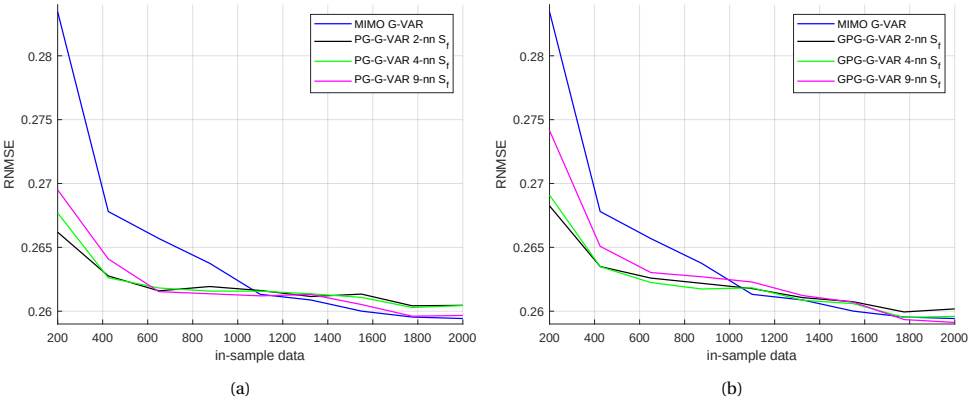


Figure 5.9: RNMSE versus the amount of in-sample data: MIMO G-VAR compared with (a) the PG-G-VAR model and (b) GPG-VAR-model, where the feature graphs are jointly estimated with the graph filter coefficients. The feature graphs are to constrained to the k-NN graph of the feature Pearson correlation matrix.

5.4. CONCLUSION

In this chapter, we investigated whether the proposed models improve the forecasting performance over the state-of-the-art models on real-world datasets. We also evaluate the effect that the amount of in-sample training data has on the performance. We see that the VAR model needs many more in-sample data to come close to the performance of the graph-based models, which could be due to the larger amount of parameters that need to be fitted. However, we also see that performance increases the most for more in-sample data, suggesting that the VAR model will eventually outperform the graph-based models.

Further, we see that the PG-VAR is on both datasets unable to incorporate the relationship between features in such a way that it improves the prediction over the G-VAR model. To a lesser degree, this also is the case for the GPG-VAR model, as even with an

estimated feature GSO the G-VAR has a similar performance on the air-quality data set. Our proposed methods perform on both datasets better than the G-VAR model, suggesting that they use the available information that captures the relations between features in an effective way. However, the MIMO G-VAR model needs more in-sample data to learn those relations.

All models that use a feature graph, show a clear performance improvement when the feature GSO is estimated, which shows the effectiveness of our proposed estimating method. Next, we see that even though we put constraints on the number of edges in the feature graph, the combined models keep a similar performance as the MIMO G-VAR. This suggests that those models could be more efficient in case there are many features.

Finally, we end this section with the remark that the graph-based models have shown to be able to model processes on graphs that are non-stationary over time, as is shown by the air-quality dataset.

6

Conclusion and Future Work

In this concluding chapter, we look back at the carried out research of this thesis and provide recommendations for future work. First, we provide a summary of the thesis in Section 6.1. In Section 6.2, we then answer the research questions posed in the introduction. At last, we present in Section 6.3 suggestions for future work.

6.1. THESIS SUMMARY

In this thesis, we have investigated how we can model the evolution of time-varying multi-dimensional network processes. This is done by experiments on both state-of-the-art product graph-based VAR models and newly proposed models.

In Chapter 1, we introduced the topic of forecasting multi-dimensional network processes, motivated the research, and stated the research questions that we want to answer in this thesis. In Chapter 2, the theoretical background that forms the basis of the research is given. We started with the fundamental basics of graph signal processing, and from there on presented graph filtering, which is the main operation applied in the G-VAR model. Next, multi-dimensional graph signals and state-of-the-art models that incorporate the relationship between different features to improve forecasting were discussed.

In Chapter 3, we proposed three new models to forecast multi-dimensional graph signals. The first model is the MIMO G-VAR model and does not take into account any prior information on how the available features are related. Instead, the MIMO G-VAR learns the dependencies between features. Further, we showed that the product graph-based models can be quite restrictive. To overcome this issue the PG-G-VAR and GPG-G-VAR models were proposed, which both combine a product graph-based VAR and the G-VAR. We also provided methods to estimate the graph filter parameters and discussed an iterative method to jointly estimate the feature GSO together with the graph filter parameters.

Through extensive experiments, we have compared our proposed models with the current state-of-the-art models on both synthetic and real-world data in Chapter 4 and Chapter 5, respectively. The results of these experiments showed that our proposed models perform better than the currently available state-of-the-art linear graph-based models.

6.2. ANSWERS TO THE RESEARCH QUESTIONS

In this section, we provide answers based on the work presented in this thesis to the research questions posed in the introduction.

RQ1: *"How to incorporate the information between features of multi-dimensional graph signals into a model to enhance their prediction?"*

To answer this question, we first discussed state-of-the-art models in Section 2.3.3, which incorporate the connection between features utilizing product graphs. A drawback of these models is that they can be too restrictive to improve prediction. To improve on these models, we introduced, in Chapter 3, three new models, which are extensions of state-of-the-art models, to enhance their prediction performance. First, in Section 3.1 the MIMO G-VAR model is proposed, which predicts the following feature values as a linear combination of G-VAR models for each feature. This model has the most enhanced degrees of freedom, thus giving the model more flexibility to approximate processes but also having more parameters to estimate. Secondly, in Section 3.2 we proposed to combine the G-VAR with a product graph-based VAR model into one model. These models

use the G-VAR to forecast each feature separately and use the PG-VAR or GPG-VAR model to add extra information contained by the feature GSO. The combined models are more efficient than the MIMO G-VAR model, in case the amount of features is large and the feature graph is sparse.

RQ2: *"How to learn the model parameters that capture the influence between features in multi-dimensional graph signal?"*

This research question is addressed in the last two sections of Chapter 3. Here we first provide two estimators to estimate the graph filter coefficients in the graph-based VAR models, and secondly, we show an iterative method that learns the feature GSO. As the MIMO G-VAR model only uses graph filter coefficients to incorporate the relationship between features, only the LS estimators described in Section 3.4 are necessary to learn those relationships. The other discussed models use the feature GSO to capture the influence between features. It is crucial to define a feature GSO that fits these models well to enhance forecasting. Our proposed iterative method, described in Section 3.5, jointly estimates the graph filter coefficients and feature GSO, and can thus be applied for that purpose.

RQ3: *"How well do the proposed architectures perform the task of forecasting multi-dimensional graph processes?"*

The last research question is addressed by the experiments performed in Chapters 4 and 5. In Chapter 4, the proposed models are evaluated on synthetic data and are compared with state-of-the-art models. Those results showed us the importance of learning the dependencies between features so it best suits the used model. In Chapter 5, we evaluated the prediction performance of our proposed models on two real-world datasets. These experiments showed that all graph-based models outperform the VAR model on the weather dataset and that on the air-quality dataset, the VAR model can't estimate a stable model, as some features are non-stationary. This emphasizes the advantage of using the graph structure. On both datasets, the graph-based models showed similar results. On the air-quality dataset, it is shown that PG-VAR and GPG-VAR models are outperformed by the G-VAR model, which suggests those approaches are not suitable to exploit the relation between features. Finally, we conclude with the observation that our proposed models outperform the G-VAR model on both datasets. This suggests that they are effective approaches in incorporating information between features to enhance the forecasting of multi-dimensional graph signals.

6.3. FUTURE WORK

We conclude with a list of possible directions for future research that extends on the work presented in this thesis:

- **Using different initial feature GSOs**

  The estimation of the feature GSO with the joint graph filter and feature GSO method is a non-convex problem. Therefore, it depends on the initial chosen feature GSO if a global optimum or local optimum is found. In this thesis, there is only one initial feature GSO used, based on the correlation matrix. Methods to define a graph based on data are proposed [35, 36], which can be used as a basis for initial feature GSOs. Another approach to finding suitable initial feature GSOs can be based on the GSO candidate generation method [37], where the GSO candidates are iteratively generated according to available data on a model for an increasing graph filter order K.

- **Constraining the estimation of the feature GSO**

  In the minimization problem (3.27) to estimate the feature GSO there is one constraint, which is that the support of the estimated feature GSO is a subset of the support of the initial feature GSO. This constraint has as many parameters to estimate as there are non-zero elements in the initial feature GSO. For example, when the initial feature GSO is the graph Laplacian the estimated feature GSO can represent an undirected graph with self-loops. Constraints could restrict the graph, corresponding to the feature GSO, to be undirected or include no self-loops. Constraints for the feature GSO could be that it is defined as an adjacency matrix or graph Laplacian.

- **Feature variant product graph filters**

  Throughout this thesis, the graph filter defined in (2.10) is used as the basis of all models. All nodes apply the same graph filter coefficient h_k to the kth shifted graph signal. This can thus be regarded as a node-invariant graph filter. A generalized graph filter is proposed in [49], which is node-variant. This graph filter is defined as

  $$\mathbf{H}_{\text{nv}} := \sum_{k=0}^{K-1} \text{diag}\left(\mathbf{h}_k\right) \mathbf{S}^k, \tag{6.1}$$

  where the $\mathbf{h}_k \in \mathbb{R}^N$ contains the node dependent graph filter coefficients.

  We showed in Section 3.2 that the node-invariant product graph filter could be restrictive when modeling multi-dimensional graph processes. Furthermore, the number of nodes in a product graph between $\mathcal{G}$ and $\mathcal{G}_{\mathcal{F}}$ is FN, where we assume that $N \gg F$. Therefore, one could consider a feature-variant product graph filter, with more flexibility than the node-invariant product graph filter and still a limited number of graph filter coefficients dependent on the number of features. This can be defined for the Kronecker product graph as

  $$\mathbf{H}_{\text{fv}} := \sum_{k=0}^{K-1} \left(\text{diag}\left(\mathbf{h}_k\right) \mathbf{S}_{\mathcal{F}}^k\right) \otimes \mathbf{S}^k, \tag{6.2}$$

  where the $\mathbf{h}_k \in \mathbb{R}^F$ contains the feature dependent graph filter coefficients. This feature-variant product graph filter applies the same filter coefficient to each feature of a shifted multi-dimensional graph signal.

- CONSTRAINED EDGE-FEATURE VARIANT PRODUCT GRAPH FILTERS

  In [27], a constrained edge-variant graph filter is proposed, which is a generalization of the node-variant graph filter. Each node in the constrained edge-variant graph filter weights the $K-1$ shifted graph signal differently for each neighbor. This graph filter is defined as

$$\mathbf{H}_{\text{c-ev}} := \sum_{k=1}^{K} (\mathbf{\Phi}_k \odot \mathbf{S})\, \mathbf{S}^{k-1} + \mathbf{\Phi}_0, \tag{6.3}$$

  where $\mathbf{\Phi}_k \in \mathbb{R}^{N \times N}$ is the matrix that contains the edge weights, which has the same support as $\mathbf{S}$, and $\mathbf{\Phi}_0$ is a diagonal weight matrix.

  When applying a product graph to model multi-dimensional graph processes, the number of edges grows at least linearly with the number of edges in the feature graph. To reduce the number of parameters in a constrained edge-variant product graph filter, we propose to apply the edge-variant graph filter only to the feature GSO. This results in a constrained edge-feature-variant product graph filter, which is defined for the Kronecker product graph as

$$\mathbf{H}_{\text{c-efv}} := \mathbf{\Phi}_0 \otimes \mathbf{I}_N + \sum_{k=1}^{K} \left((\mathbf{\Phi}_k \odot \mathbf{S}_{\mathscr{F}})\, \mathbf{S}_{\mathscr{F}}^{k-1} \otimes \mathbf{S}^k \right), \tag{6.4}$$

  where now $\mathbf{\Phi}_k \in \mathbb{R}^{F \times F}$ is the matrix that contains the feature edge weights, which has the same support as $\mathbf{S}_{\mathscr{F}}$, and $\mathbf{\Phi}_0$ is a diagonal weight matrix. The numerical experiments showed that estimating $\mathbf{S}_{\mathscr{F}}$ leads to increased forecasting performance, but with the cost that a non-convex problem has to be solved. The constrained edge-feature-variant graph filter is linear with respect to the filter coefficients, which makes them easier to estimate, while also adding flexibility to put weights on the edges of the feature graph.

- NON-LINEAR MULTI-DIMENSIONAL GRAPH MODELS

  In this thesis, the research has only been focused on linear forecasting models. Graph convolutional neural networks (GCNN) generalize convolutional neural networks to the graph domain [50, 9]. The GCNN framework can be used to extend the discussed linear models to non-linear models. A graph perceptron can be seen as the most basic form of a GCNN and applies an elementwise nonlinearity to a filtered graph signal, i.e.,

$$\mathbf{y} = \sigma(\mathbf{H}\mathbf{x}), \tag{6.5}$$

  where $\sigma(\cdot)$ is a non-linear activation function, such as ReLU or Softmax, and $\mathbf{H} \in \mathbb{R}^{N \times N}$ is a matrix containing the graph filter.

  Like the graph perceptron, one could extend the linear graph-based forecasting models to non-linear models. This is done by applying the elementwise nonlinearity to each shifted graph signal in the graph-based VAR models. For example, this means that the non-linear G-VAR model is expressed as

$$\mathbf{x}_t = \sum_{p=1}^{P} \sigma\left(\sum_{k=0}^{K-1} h_{kp} \mathbf{S}^k \mathbf{x}_{t-p} \right). \tag{6.6}$$

Similarly, other graph-based VAR models can be transformed into non-linear models.

- EXTERNAL INPUTS

  A more challenging problem would be the scenario where the evolution of a graph process is also dependent on external inputs. For example, in a brain network, where nodes represent electrodes that measure electroencephalography (EEG) signals, visual or sound effects could be used as external stimuli. If the model of how the brain network behaves under specific external inputs is known, this can be used to detect abnormalities in the brain.

BIBLIOGRAPHY

[1] Monson H. Hayes. *Statistical digital signal processing and modeling*. John Wiley and Sons, 1996.

[2] Ireneusz Jabłoński. "Graph Signal Processing in Applications to Sensor Networks, Smart Grids, and Smart Cities". In: *IEEE Sensors Journal* 17.23 (2017), pp. 7659–7666. DOI: 10.1109/JSEN.2017.2733767.

[3] YAO MA and JILIANG TANG. *DEEP LEARNING ON GRAPHS*. CAMBRIDGE UNIV PRESS, 2021.

[4] Ali Dehghantanha. "Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, Github, and More , by Matthew A. Russell". In: *Journal of Information Privacy and Security* 11 (Apr. 2015), pp. 137–138. DOI: 10.1080/15536548.2015.1046287.

[5] Dmitri Goldenberg. "Social Network Analysis: From Graph Theory to Applications with Python". In: *CoRR* abs/2102.10014 (2021). arXiv: 2102.10014. URL: https://arxiv.org/abs/2102.10014.

[6] Ulrich Stelzl et al. "A Human Protein-Protein Interaction Network: A Resource for Annotating the Proteome". In: *Cell* 122.6 (2005), pp. 957–968. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2005.08.029. URL: https://www.sciencedirect.com/science/article/pii/S0092867405008664.

[7] Jean-François Rual et al. "Towards a proteome-scale map of the human protein–protein interaction network". In: *Nature* 437.7062 (2005), pp. 1173–1178.

[8] Abir De et al. *Learning and Forecasting Opinion Dynamics in Social Networks*. 2016. arXiv: 1506.05474 [cs.SI].

[9] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32 (2019), pp. 4–24.

[10] Antonio Ortega et al. "Graph Signal Processing: Overview, Challenges, and Applications". In: *Proceedings of the IEEE* 106 (May 2018), pp. 808–828. DOI: 10.1109/JPROC.2018.2820126.

[11] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge University Press, 2017.

[12] Aliaksei Sandryhaila and José M. F. Moura. "Discrete Signal Processing on Graphs". In: *IEEE Transactions on Signal Processing* 61.7 (2013), pp. 1644–1656. DOI: 10.1109/TSP.2013.2238935.

[13] David Shuman et al. "The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains". In: *IEEE Signal Processing Magazine* 30 (Oct. 2012). DOI: 10.1109/MSP.2012.2235192.

[14] Siheng Chen et al. “Signal denoising on graphs via graph filtering”. In: *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (2014). DOI: 10.1109/globalsip.2014.7032244.

[15] Elvin Isufi et al. “Observing and tracking bandlimited graph processes from sampled measurements”. In: *Signal Processing* 177 (Aug. 2020), p. 107749. DOI: 10.1016/j.sigpro.2020.107749.

[16] Andreas Loukas, Elvin Isufifi, and Nathanaël Perraudin. “Predicting the evolution of stationary graph signals”. In: Oct. 2017, pp. 60–64. DOI: 10.1109/ACSSC.2017.8335136.

[17] Daniel Romero, Vassilis Ioannidis, and G.B. Giannakis. “Kernel-based Reconstruction of Space-time Functions on Dynamic Graphs”. In: *IEEE Journal of Selected Topics in Signal Processing* PP (Dec. 2016). DOI: 10.1109/JSTSP.2017.2726976.

[18] Paolo Di Lorenzo et al. “Adaptive Graph Signal Processing: Algorithms and Optimal Sampling Strategies”. In: *IEEE Transactions on Signal Processing* 66.13 (2018), pp. 3584–3598. DOI: 10.1109/TSP.2018.2835384.

[19] Elvin Isufi et al. “Forecasting Time Series With VARMA Recursions on Graphs”. In: *IEEE Transactions on Signal Processing* 67.18 (2019), pp. 4870–4885. DOI: 10.1109/TSP.2019.2929930.

[20] Aliaksei Sandryhaila and Jose M.F. Moura. “Big Data Analysis with Signal Processing on Graphs: Representation and processing of massive data sets with irregular structure”. In: *IEEE Signal Processing Magazine* 31.5 (2014), pp. 80–90. DOI: 10.1109/MSP.2014.2329213.

[21] 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)*. 2020, pp. 5575–5579. DOI: 10.1109/ICASSP40776.2020.9053522.

[22] Chung Fan R K. *Spectral graph theory*. Published for the Conference Board of the Mathematical Sciences by the American Mathematical Society, cc1997, 1997.

[23] Markus Puschel and JosÉ M. F. Moura. “Algebraic Signal Processing Theory: Foundation and 1-D Time”. In: *IEEE Transactions on Signal Processing* 56.8 (2008), pp. 3572–3585. DOI: 10.1109/TSP.2008.925261.

[24] Adnan Gavili and Xiao-Ping Zhang. “On the Shift Operator, Graph Frequency, and Optimal Filtering in Graph Signal Processing”. In: *IEEE Transactions on Signal Processing* 65.23 (2017), pp. 6303–6318. DOI: 10.1109/TSP.2017.2752689.

[25] Aliaksei Sandryhaila and José M. F. Moura. “Discrete Signal Processing on Graphs: Frequency Analysis”. In: *IEEE Transactions on Signal Processing* 62.12 (2014), pp. 3042–3054. DOI: 10.1109/TSP.2014.2321121.

[26] Ljubisa Stankovic et al. “Understanding the Basis of Graph Signal Processing via an Intuitive Example-Driven Approach [Lecture Notes]”. In: *IEEE Signal Processing Magazine* 36.6 (2019), pp. 133–145. DOI: 10.1109/MSP.2019.2929832.

[27] E. Isufi. “Graph-time signal processing: Filtering and sampling strategies”. PhD thesis. 2019. DOI: https://doi.org/10.4233/uuid:e52cc182-457c-4687-baee-d0f72af36950.

[28] Fernando Gama et al. “MIMO Graph Filters for Convolutional Neural Networks”. In: *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 2018, pp. 1–5. DOI: 10.1109/SPAWC.2018.8445934.

[29] Richard Hammack, Wilfried Imrich, and Sand Klavžar. “Handbook of Product Graphs”. In: (Jan. 2011).

[30] Ljubisa Stankovic, Milos Dakovic, and Ervin Sejdic. “Introduction to Graph Signal Processing”. In: Jan. 2019, pp. 3–108. ISBN: 978-3-030-03573-0. DOI: 10.1007/978-3-030-03574-7_1.

[31] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer, 2005.

[32] Peter J. Brockwell and Richard A. Davis. *Introduction to time series and forecasting*. Springer, 2016.

[33] Eric Zivot and Jiahui Wang. *Modeling financial time series with S-PLUS*. Springer, 2006.

[34] Jonathan Mei and Jose Moura. “Signal Processing on Graphs: Causal Modeling of Unstructured Data”. In: *IEEE Transactions on Signal Processing* PP (Dec. 2016), pp. 1–1. DOI: 10.1109/TSP.2016.2634543.

[35] Gonzalo Mateos et al. “Connecting the Dots: Identifying Network Structure via Graph Signal Processing”. In: (Oct. 2018).

[36] Xiaowen Dong et al. “Learning Graphs From Data: A Signal Representation Perspective”. In: *IEEE Signal Processing Magazine* 36 (May 2019), pp. 44–63. DOI: 10.1109/MSP.2018.2887284.

[37] Alberto Natali, Mario Coutino, and Geert Leus. “Topology-Aware Joint Graph Filter and Edge Weight Identification for Network Processes”. In: (July 2020).

[38] Nathanaël Perraudin et al. “GSPBOX: A toolbox for signal processing on graphs”. In: *ArXiv e-prints* (Aug. 2014). arXiv: 1408.5781 [cs.IT].

[39] “Vector Autoregressive Models for Multivariate Time Series”. In: *Modeling Financial Time Series with S-PLUS®*. New York, NY: Springer New York, 2006, pp. 385–429. ISBN: 978-0-387-32348-0. DOI: 10.1007/978-0-387-32348-0_11. URL: https://doi.org/10.1007/978-0-387-32348-0_11.

[40] *RAINFALL TIME SERIES FROM 1850-2010 FOR IRELAND*. URL: https://www.met.ie/climate/available-data/long-term-data-sets. (accessed: 30.06.2021).

[41] Marcelo Spelta and Wallace Martins. “Online Temperature Estimation using Graph Signals”. In: Sept. 2018. DOI: 10.14209/sbrt.2018.164.

[42] Carl Carter. “Great circle distances”. In: *SiRF White Paper* (2002).

[43] Shuyi Zhang et al. “Cautionary tales on air-quality improvement in Beijing”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473 (2017).

[44] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data.

[45] Vitor Cerqueira, Luís Torgo, and Igor Mozetič. “Evaluating time series forecasting models: an empirical study on performance estimation methods”. In: *Machine Learning* 109 (Nov. 2020), pp. 1–32. DOI: 10.1007/s10994-020-05910-7.

[46] Christoph Bergmeir, Rob Hyndman, and Bonsoo Koo. “A note on the validity of cross-validation for evaluating autoregressive time series prediction”. In: *Computational Statistics Data Analysis* 120 (Nov. 2017). DOI: 10.1016/j.csda.2017.11.003.

[47] Christoph Bergmeir and José Benítez. “On the use of cross-validation for time series predictor evaluation”. In: *Information Sciences* 191 (May 2012), pp. 192–213. DOI: 10.1016/j.ins.2011.12.028.

[48] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[49] Santiago Segarra, Antonio G. Marques, and Alejandro Ribeiro. “Optimal Graph-Filter Design and Applications to Distributed Linear Network Operators”. In: *IEEE Transactions on Signal Processing* 65.15 (Aug. 2017), pp. 4117–4131. ISSN: 1941-0476. DOI: 10.1109/TSP.2017.2703660.

[50] Fernando Gama et al. “Graphs, Convolutions, and Neural Networks: From Graph Filters to Graph Neural Networks”. In: *IEEE Signal Processing Magazine* 37.6 (Nov. 2020), pp. 128–138. ISSN: 1558-0792. DOI: 10.1109/msp.2020.3016143. URL: http://dx.doi.org/10.1109/MSP.2020.3016143.

A

Real-World Data: Supplementary Material

A

In Section 5.3, we showed the prediction error of the discussed forecasting models on two real-world datasets. To illustrate how these error values relate to the forecasting capability, the predicted values are plotted against the ground truth values.

For the Air-quality dataset, this is depicted in Figure A.1 and A.2, for the PG-VAR and MIMO G-VAR models, respectively. In Table A.1, the RNMSE is shown for the plotted models. The predictions of the VAR, PG-VAR, and MIMO G-VAR models on the weather dataset are presented in Figure A.3 and A.4. In Table A.2, the RNMSE is shown for the plotted models. In both tables the RNMSE for the 1-lagged signal is included, i.e., the values of previous time steps are taken as predicted values.

	1-lagged	PG-VAR	MIMO G-VAR
RNMSE	0.3722	0.3623	0.3318

Table A.1: Prediction errors of the Air-quality dataset at 1st fold and for 1325 number of in-sample data samples

	1-lagged	VAR	PG-VAR	MIMO G-VAR
RNMSE	0.2520	0.2466	0.2189	0.2132

Table A.2: Prediction errors of the Wheather dataset at 1st fold and for 1325 number of in-sample data samples

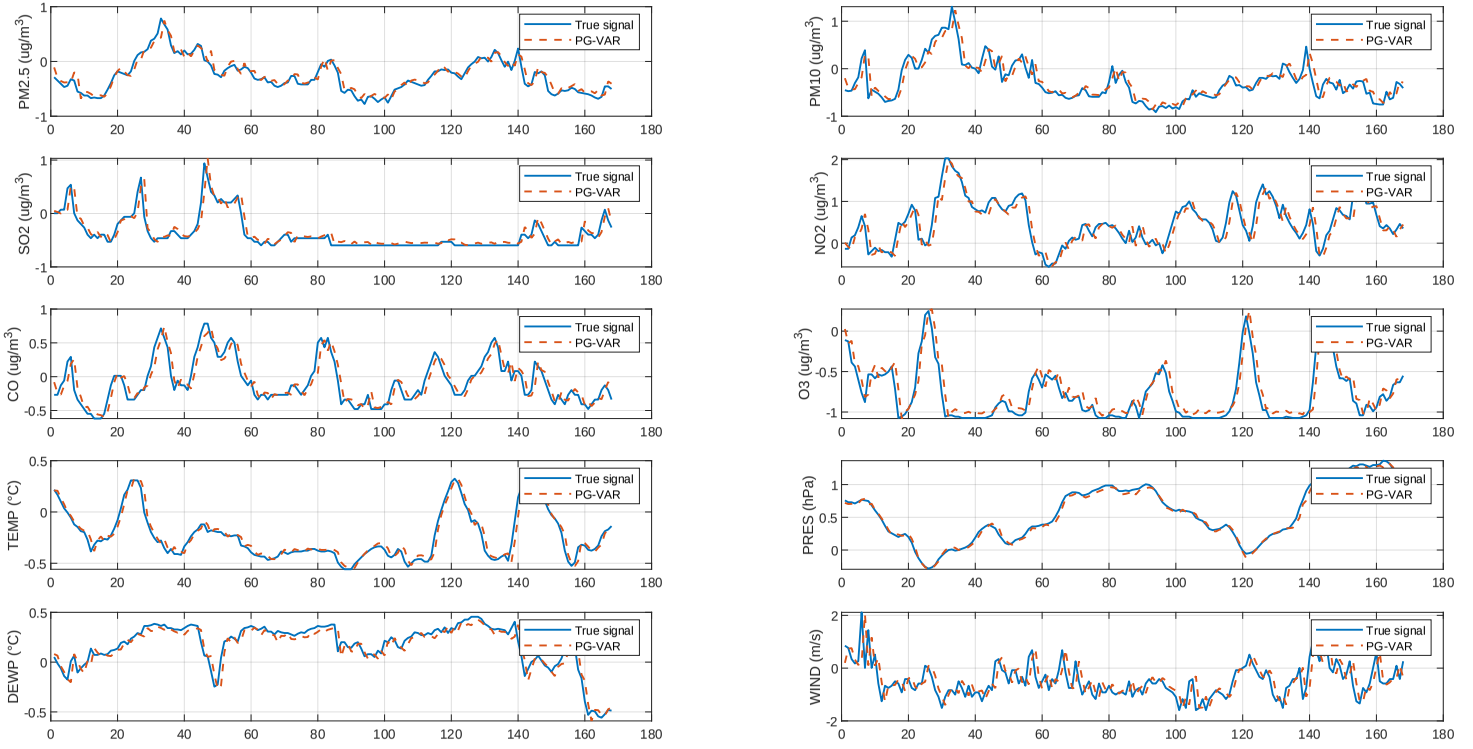


Figure A.1: Prediction values of the PG-VAR model on the Air-quality dataset at the 1st fold and for 1325 number of in-sample data samples. Data is from station number 1, located at the National Olympic Sports Center in Beijing.

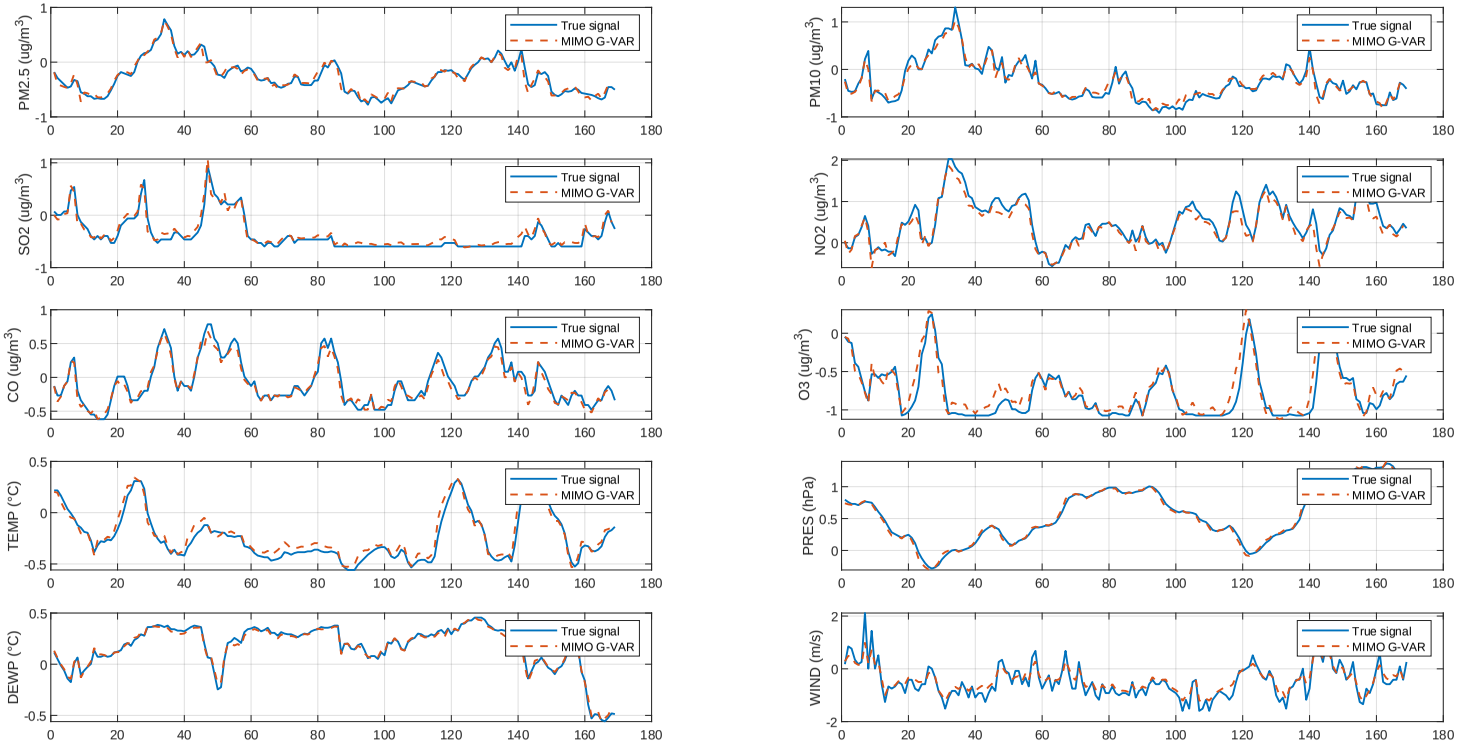


Figure A.2: Prediction values of the MIMO G-VAR model on the Air-quality dataset at the 1st fold and for 1325 number of in-sample data samples. Data is from station number 1, located at the National Olympic Sports Center in Beijing.

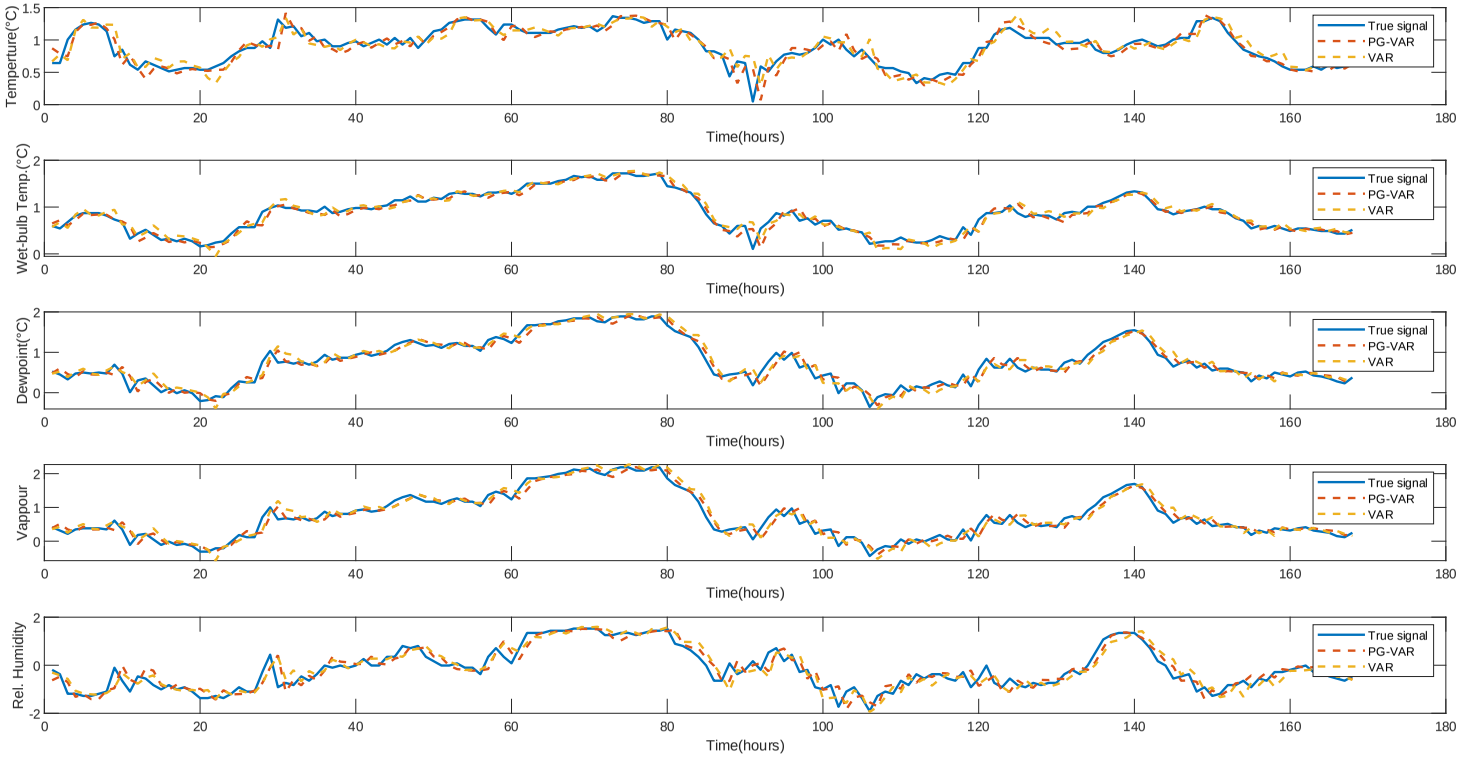


Figure A.3: Prediction values of the VAR and PG-VAR model on the weather dataset at the 1st fold and for 875 number of in-sample data samples. Data is from station number 13, located near Mace Head, Ireland

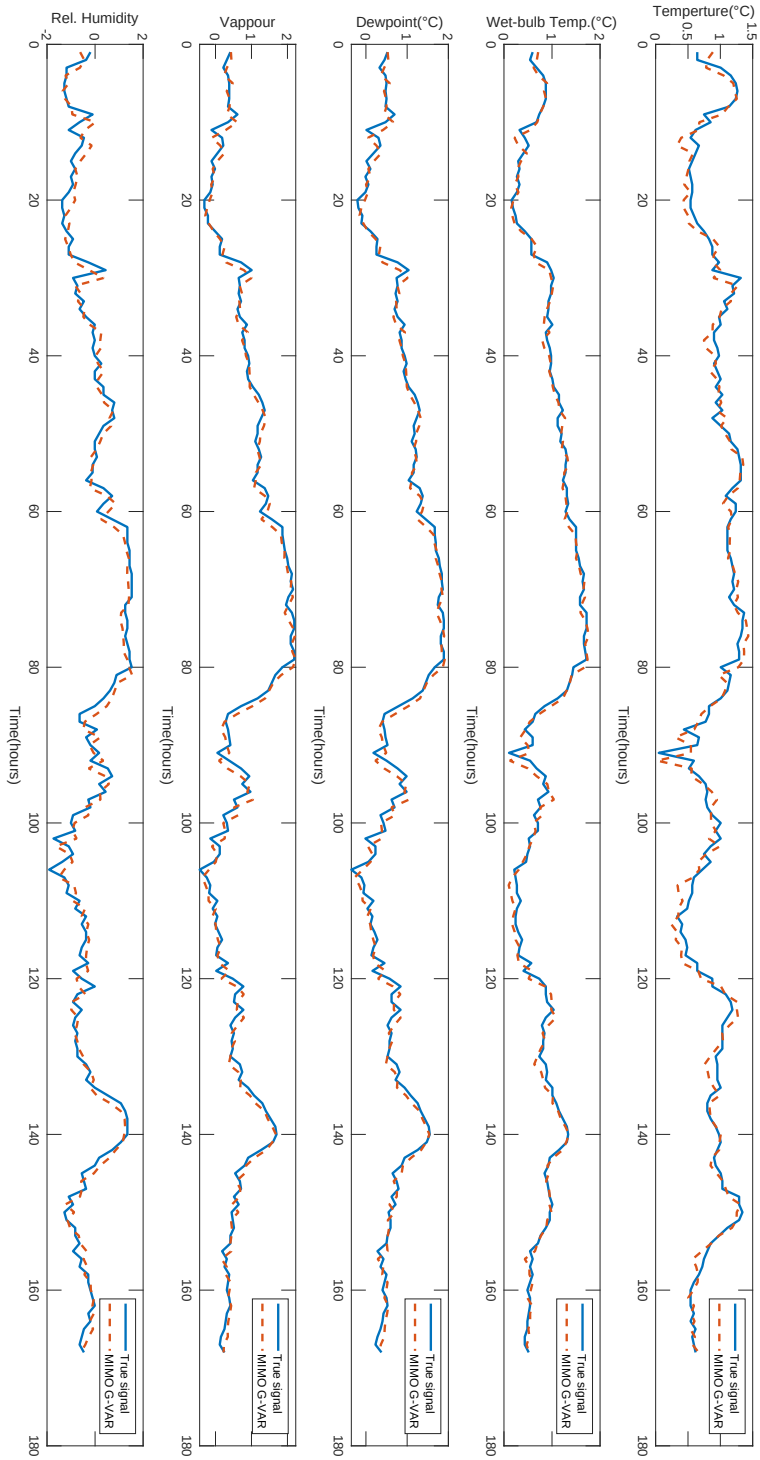


Figure A.4: Prediction values of the MIMO G-VAR model on the Air-quality dataset at the 1st fold and for 875 number of in-sample data samples. Data is from station number 13, located near Mace Head, Ireland