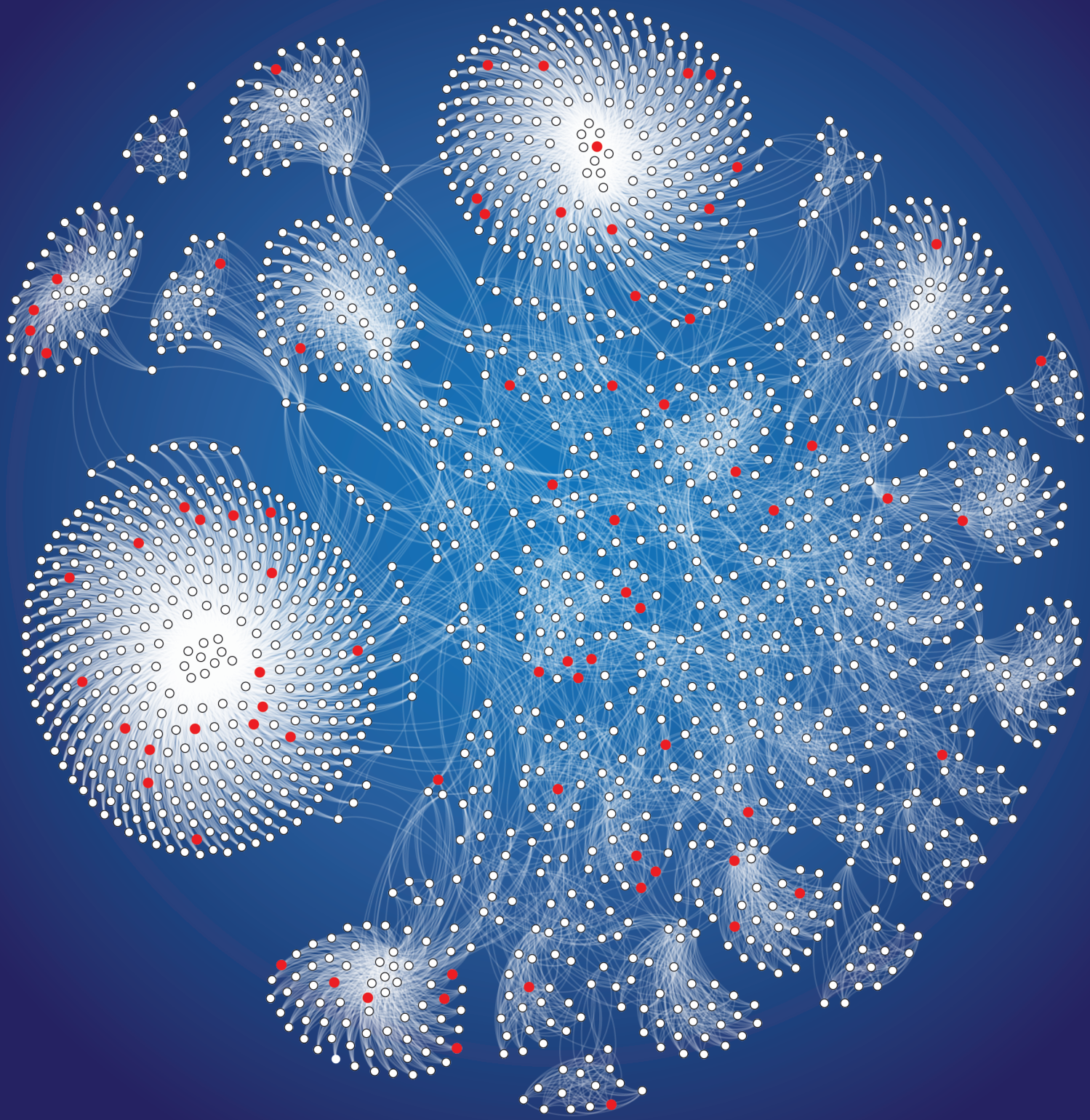
# Multidomain Graph Signal Processing

## Learning and Sampling



TU Delft

Guillermo Ortiz Jiménez

# Multidomain Graph Signal Processing
## Learning and Sampling

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL ENGINEERING

by

Guillermo Ortiz Jiménez
born in Madrid, Spain

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

**Delft University of Technology**

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Multidomain Graph Signal Processing"** by **Guillermo Ortiz Jiménez** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 23 August 2018

Chairman: _____
prof.dr.ir. G. Leus

Advisors: _____
prof.dr.ir. G. Leus

_____
dr.ir. S.P. Chepuri

Committee Members: _____
dr.ir. R. Hendriks

_____
dr.ir. D. Tax

# Abstract

In this era of data deluge, we are overwhelmed with massive volumes of extremely complex datasets. Data generated today is complex because it lacks a clear geometric structure, comes in great volumes, and it often contains information from multiple domains. In this thesis, we address these issues and propose two theoretical frameworks to handle such multidomain dataset.

To begin with, we extend the recently developed geometric deep learning framework to multidomain graph signals, e.g., time-varying signals, defining a new type of convolutional layer that will allow us to deal with graph signals defined on top of several domains, e.g., electroencephalograms or traffic networks. After discussing its properties and motivating its use, we show how this operation can be efficiently implemented to run on a GPU and demonstrate its generalization abilities on a synthetic dataset.

Next, we consider the problem of designing sparse sampling strategies for multidomain signals, which can be represented using tensors. To keep the framework general, we do not restrict ourselves to multidomain signals defined on irregular domains. Nonetheless, this particularization to multidomain graph signals is also presented. To do so, we leverage the multidomain structure of tensor signals and propose to acquire samples using a Kronecker-structured sensing function, thereby circumventing the curse of dimensionality. For designing such sensing functions, we develop several low-complexity greedy algorithms based on submodular optimization methods that compute near-optimal sampling sets. To validate the developed theory, we present several numerical examples, ranging from multi-antenna communications to graph signal processing.

# Acknowledgments

First and foremost, I would like to express my gratitude to all the people without whom this thesis would not have been possible.

First, to my daily supervisor, Dr. Sundeep Chepuri, whose commitment and passion to research, as well as his invaluable advice have made me reflect continuously on what I was doing, and have immensely improved the quality of my work. Yet, I am especially grateful to his personal guidance and mentorship, which extend far beyond professionalism, and have been a great inspiration to aim higher in my career. Thank you!

I would also like to thank Prof. Geert Leus, for his timely supervision, and for having given me the freedom to always explore new ideas during my thesis. His door has always been open, and I have greatly learnt from his constructive criticism.

Working at CAS has been a real pleasure, and that is thanks to all its members. A special mention should go to Mario, whose collaboration has been key in the realization of this thesis, and with whom I have greatly enjoyed working. ¡Gracias! Also, I would like to thank Lucas and Bas, for having made all those necessary coffee breaks so pleasant, and for having shared so many hours with me in the MSc. room.

Finally, I would like to thank the rest of my friends and family, scattered around the world, for their love and words of encouragement. My achievements, and who I am today would not be possible without your kind support.

Guillermo Ortiz Jiménez
Delft, The Netherlands
23 August 2018

*Solo el misterio nos hace*
*vivir. Solo el misterio.*
- Federico García Lorca

*Only mystery makes us live.*
*Only mystery.*
- Federico García Lorca

# Contents

# List of Figures

# List of Tables

# Nomenclature

**Mathematical objects**

| | |
|---|---|
| $x$ | Scalar |
| $\mathbf{x}$ | Vector |
| $\mathbf{X}$ | Matrix |
| $\boldsymbol{\mathcal{X}}$ | Tensor |
| $\mathcal{X}$ | Set |
| $\mathcal{G}$ | Graph |

**Set theory**

| | |
|---|---|
| $\mathbb{R}$ | Set of real numbers |
| $\mathbb{C}$ | Set of complex numbers |
| $2^{\mathcal{X}}$ | Power set of $\mathcal{X}$, i.e., set of all subsets of $\mathcal{X}$ |
| $\varnothing$ | Empty set |
| $\cap$ | Intersection |
| $\cup$ | Union |
| $\subset$ | Strict subset |
| $\subseteq$ | Subset |
| $\times$ | Cartesian product |
| $\lvert\mathcal{X}\rvert$ | Cardinality of $\mathcal{X}$ |
| $\mathcal{M}$ | Matroid |
| $\mathcal{I}$ | Independent set |

**Graph signal processing**

| | |
|---|---|
| $\mathbf{L}$ | Graph Laplacian |
| $\mathbf{S}$ | Graph shift operator |
| $\square$ | Graph Cartesian product |
| $\diamond$ | General graph product |
| $*$ | Time or graph convolution (inferred from context) |
| $*_{\mathcal{G}}$ | Graph convolution |
| $*_{\mathcal{T}}$ | Time convolution |
| $\mathcal{F}$ | Time or Graph Fourier transform (inferred from context) |
| $\mathcal{F}_{\mathcal{G}}$ | Graph Fourier transform |
| $\mathcal{F}_{\mathcal{T}}$ | Standard discrete time Fourier transform |

### Linear algebra

| | |
|---|---|
| $\mathbf{I}$ | Identity matrix |
| $\mathbf{1}$ | All-ones vector |
| $\otimes$ | Kronecker product |
| $\odot$ | Khatri-Rao product |
| $\circ$ | Hadamard product |
| $\oplus$ | Kronecker sum |
| $\bullet_n$ | $n$-mode tensor-matrix product |
| $\bullet_n^m$ | Tensor contraction between the $n$th mode and the $m$th mode of two tensors |
| $(\cdot)^*$ | Matrix conjugate, without transpose |
| $(\cdot)^\dagger$ | Moore-Penrose pseudoinverse of a matrix |
| $(\cdot)^H$ | Matrix conjugate transpose |
| $(\cdot)^T$ | Matrix transpose |
| $(\cdot)^{\circ n}$ | $n$th element-wise power of a matrix |
| $\langle \cdot, \cdot \rangle$ | Inner product between elements of Euclidean space |
| $\left\|\cdot\right\|_2$ | $\ell_2$-norm of a vector |
| $\left\|\cdot\right\|_F$ | Frobenius-norm of a matrix or tensor |
| $\det\{\cdot\}$ | Determinant of a matrix |
| $\lambda_{\max}\{\cdot\}$ | Maximum eigenvalue of a matrix |
| $\mathrm{tr}\{\cdot\}$ | Trace of a matrix |

### Other symbols

| | |
|---|---|
| $\mathcal{O}(\cdot)$ | Big O notation |
| $\rho(\cdot)$ | Point-wise nonlinearity |
| $T_k(\cdot)$ | Chebyshev polynomial of order $k$ |
| $\mathrm{Conv}\{\mathbf{x}\}$ | Toeplitz matrix with shifted copies of vector $\mathbf{x}$ on its rows |
| $\log\{\cdot\}$ | Natural logarithm |
| $\mathrm{diag}\{\mathbf{x}\}$ | Diagonal matrix with $\mathbf{x}$ in its main diagonal. |
| $\mathbb{E}\{\cdot\}$ | Expectation operator |
| $\preceq$ | Element-wise inequality between vectors |

# 1 Introduction

In this era of data deluge, we are submerged under massive volumes of extremely complex dataset, such as sensor network readings, brain activity records, opinions in social media, or high-resolution 3D point clouds. Signal processing has traditionally been the tool we used to extract meaningful information from noisy data; but data today is more intricate than ever: it lacks a clear physical model, comes in great volumes, and it is often defined over multiple irregular domains.

Take, for instance, the problem of predicting the volume of traffic in a city in the next hour, given data of the traffic evolution recorded by a sensor network scattered across the city. It is clear that in a modern busy city the size of this dataset would be tremendous. Furthermore, even though we might have an intuition on how traffic evolves during a day, or a week, finding an explicit model that describes the sophisticated interactions between different neighborhoods is a rather challenging task. Finally, this data resides on two domains: a temporal one, and spatial one with an irregular geometric structure that is best represented by a graph of street connections. Classical signal processing techniques cannot deal with these peculiarities, so we need to find better tools to tackle them.

The emerging fields of deep learning [1] and graph signal processing (GSP) [2] have recently attracted a lot of attention as potential tools to overcome these challenges. On the one hand, deep learning, a subfield of machine learning, tries to recognize patterns in data without having access to a given model. The main difference with machine learning being that it does so by exploiting the geometric structure of the data support. On the other hand, GSP focuses on the extension of traditional signal processing techniques developed for signals living on regular domains such as timelines or images, to signals that reside on irregular domains with a network structure. The combination of these two fields has lead to the creation of geometric deep learning [3], a new theoretical framework for discovering patterns in graph signals. In the first part of this thesis, we will explore an open issue in this field: how to deal with multidomain graph signals, such as the ones in the aforementioned traffic prediction problem. We refer to this extension as *multidomain geometric deep learning*.

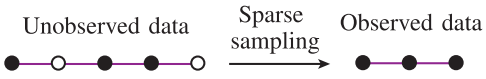
In many engineering and scientific applications, we frequently encounter large volumes of multisensor data defined over multiple domains that are complex in nature. For example, in wireless communications, received data per user may be indexed in space, time, and frequency. Similarly, in hyperspectral imaging, a scene measured in different wavelengths contains information from the three-dimensional spatial domain as well as the spectral domain. And also, when dealing with network data or point clouds, often multidimensional time-varying signals are defined on each node in the network. To process such multisensor datasets, *higher-order tensors* or *multiway arrays* have been proven to be extremely useful.

In practice, however, due to limited access to sensing resources, economical con-

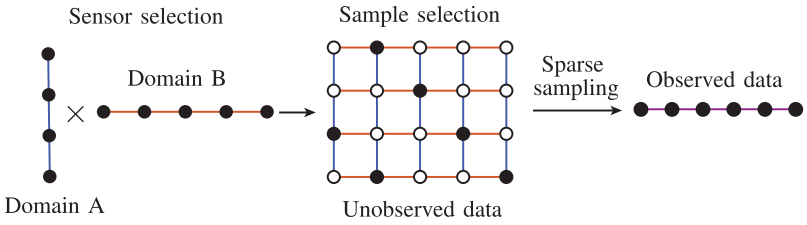
straints, or physical limitations, it is often not possible to measure such multidomain signals using every combination of sensors related to different domains. To cope with such issues, in the second part of this thesis, we propose *sparse sampling* techniques to acquire multisensor data that can be represented using tensors.

Sparse samplers can be designed to select a subset of measurements (e.g., spatial or temporal samples as illustrated in Fig. 1.1a) such that a desired inference performance is achieved. This problem is referred to as sparse sampling or sensor selection [4]. An example of this is field estimation, in which the measured field is related to the source signal of interest through a linear model (i.e., the measured field lives in a known subspace). To infer the source signal, a linear inverse problem is solved. In a resource-constrained and noisy environment, since many measurements cannot be taken, it is crucial to carefully select the best subset of measurements. The problem of choosing the best subset of samples from a large pool of measurements is combinatorial in nature, and extremely hard to solve, even for small-sized problems. Thus, most of the research efforts on this topic have been focused on finding suboptimal sampling strategies that yield good approximations of the optimal solution.

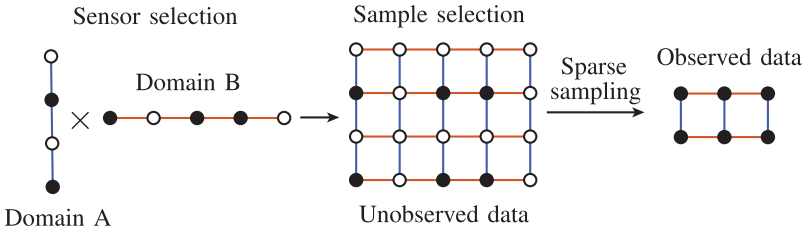
For signals defined over multiple domains, the dimensionality of the measurements grows much faster. An illustration of this *curse of dimensionality* is provided in Fig. 1.1b, wherein the measurements now have to be systematically selected from even a larger pool of measurements. Typically used suboptimal sensor selection strategies




(a) Single domain sparse sampling




(b) Unstructured multidomain sparse sampling




(c) Kronecker-structured multidomain sparse sampling

Figure 1.1: Different sparse sampling schemes. Black (white) dots represent selected (unselected) measurement locations. Blue and red lines determine different domain directions, and a purple line means that data has a single-domain structure.

are not useful anymore because their complexity is too high; or simply because they need to store very large matrices that do not fit in memory. Usually, acquiring arbitrary samples from a multidomain signal, requires that sensors are placed densely on every domain, which greatly increases the infrastructure costs. Hence, we propose an efficient *Kronecker-structured* sparse sampling strategy for gathering multidomain signals that overcomes the aforementioned issues. In Kronecker-structured sparse sampling, instead of choosing a subset of measurements from all possible combined domain locations (as in Fig. 1.1b), we propose to choose a subset of sensing locations from each domain and then combine them to obtain multidimensional observations; see an illustration of such a Kronecker-structured sparse sampling in Fig 1.1c. We will see later that taking this approach will allow us to define computationally efficient design algorithms that are useful in big data scenarios.

## Research statements

In this M.Sc. thesis we answer the following two main research questions:

**Learning**

*How to generalize graph convolutional neural networks to account for the multidomain structure in product graph signals?*

**Sampling**

*How to choose a subset of sampling locations from each domain of a multidomain (possibly irregular) signal so that its reconstruction has minimum error?*

## Thesis outline

This M.Sc. thesis is structured into two parts, namely, learning and sampling. As we saw before, the first one deals with the extension of deep learning to multidomain graph signals, and the second one with the design of sparse samplers for inverse problems with multidomain signals. The chapter distribution of the remainder of this thesis is as follows:

- **Part I: Learning**
  - Chapter 2: We briefly introduce the theory of deep learning and graph signal processing. We also review the state-of-the-art in geometric deep learning.
  - Chapter 3: We propose a new type of graph convolutional layer that exploits correlations between the multiple graph domains.
  - Chapter 4: We evaluate the performance of such a layer on a synthetic dataset where we prove its robustness against adversarial noise and limited access to training data.

- **Part II: Sampling**
  - Chapter 5: We summarize the notation, and mathematical theory of tensor algebra and discrete optimization that will be extensively used througout this part.
  - Chapter 6: We formally define the sparse tensor sampling problem as a discrete optimization problem and review the state-of-the art in single domain sparse sampling.
  - Chapter 7: We propose two greedy algorithms based on submodular optimization theory and one algorithm based on convex optimization to design the sampling set in case the multilinear decomposition of the target tensor signal has a dense core.
  - Chapter 8: We propose two algorithms (one based on the submodularity of the objective function, and another based on its convexity) to design the sampling set in case the multilinear decomposition of the target tensor signal has a hyperdiagonal core.
  - Chapter 9: We evaluate the performance of the proposed algorithms for different amounts of compression using synthetic data. We also test the applicability of our framework to solve problems with real data in graph signal processing, recommendation systems, and MIMO communications.
- **Conclusions:** We conclude the thesis summarizing the main contributions of both parts and with a compilation of research directions that may be interesting to pursue in the future.

## Publications

The research conducted during this M.Sc. thesis has lead to the submission of the following two publications:

- G. Ortiz-Jiménez, M. Coutino, S.P. Chepuri, and G. Leus, "Sparse Sampling for Inverse Problems with Tensors". Submitted to *IEEE Transactions on Signal Processing* Jun. 2018. Available: https://arxiv.org/abs/1806.10976
- G. Ortiz-Jiménez, M. Coutino, S.P. Chepuri, and G. Leus, "Sampling and Reconstruction of Signals on Product Graphs". Submitted to the *2018 6th IEEE Global Conference on Signal and Information Processing* Jun. 2018. Available: https://arxiv.org/abs/1807.00145

# Part I
# Learning

# Background 2

## 2.1 Deep learning

Deep learning [1, 5] refers to a subset of concepts and tools in machine learning that tries to solve supervised, unsupervised, and reinforcement learning tasks using neural networks. In contrast to classical methods, deep learning proposes solving the whole machine learning pipeline (feature extraction, feature selection, and inference) with a common framework of multilayer architectures and first-order optimization methods.

Convolutional neural networks (CNNs) are clearly the most popular tool among all deep learning techniques, mainly due to their success in computer vision [6, 7] and natural language processing [8] tasks. As most neural networks, CNNs are formed by sequentially stacking a set of layers composed of a linear mapping and a point-wise non-linearity as illustrated in Fig. 2.1. However, in CNNs the linear mapping generally takes the form of a convolution operator whose kernel parameters are learnt during training. The concatenation of many such layers, intercalated with subsampling stages referred to as pooling, can generate a very wide class of universal function approximators that have been shown to perform very well on many machine learning tasks. Besides, having a multilayer architecture allows solving optimization problems using simple first-order methods and the backpropagation algorithm [5]: an efficient way to perform gradient descent iterations in a neural network.

The reason for the success of CNNs is still an open issue which has been especially hard to settle due to the black box nature of most deep learning methods. The most accepted explanation argues that the power in CNNs comes from the fact that they leverage the stationarity and stability to local translations of the problems they try

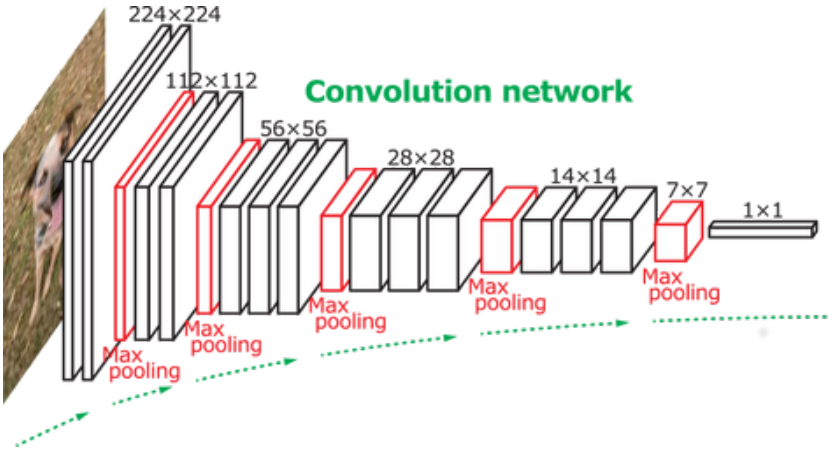



Figure 2.1: Example of a convolutional neural network used to classify images. Taken from [7].

to solve [1, 5, 9]. Whereas, their fixed number of parameters, i.e., independent to the input size, makes them amenable to train even in high dimensional scenarios. Especially, thanks to the advent of graphic processing units (GPUs) for general purpose programming in recent years.

Due to space limitations, we will not dwell on the details of how modern deep learning architectures work. For an in-depth description of such models, and review of the recent developments in deep learning we refer the reader to [5].

## 2.2 Graph signal processing

Graph signal processing aims to extend basic concepts of classical signal processing developed for signals defined on Euclidean domains to signals that reside on irregular domains with a network structure [2]. Mathematically, a graph signal $\mathbf{x} \in \mathbb{R}^N$ consists of a collection of $N$ values that can be associated to the nodes of a known graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with a vertex set $\mathcal{V}$, and an edge set $\mathcal{E}$ that reveals the connections between the nodes. For the sake of exposition, we will focus on undirected graphs.

Using the graph structure, we can construct an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ that stores the strength of the connection between nodes $i$ and $j$ in its $(i, j)$th and $(j, i)$th entries, $[\mathbf{A}]_{i,j} = [\mathbf{A}]_{j,i}$. The degree of the $i$th node of a graph is defined as $d_i = \sum_{j=1}^{N} [\mathbf{A}]_{i,j}$. Related to the adjacency matrix, we can also define an alternative matrix representation known as the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A} \in \mathbb{R}^{N \times N}$, where $\mathbf{D} = \text{diag}\{d_1, \ldots, d_N\} \in \mathbb{R}^{N \times N}$. Both the adjacency matrix and the graph Laplacian belong to the class of matrices that represent a valid *graph-shift operator* (GSO) [2], i.e., a matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ with a sparsity pattern defined by the graph connectivity.

Since for undirected graphs $\mathbf{S}$ is symmetric, it admits an eigenvalue decomposition

$$\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^H = [\mathbf{u}_1 \cdots \mathbf{u}_N]\text{diag}\{\lambda_1, \ldots, \lambda_N\}[\mathbf{u}_1 \cdots \mathbf{u}_N]^H, \tag{2.1}$$

where the eigenvectors $\{\mathbf{u}_i\}_{i=1}^N$ and the eigenvalues $\{\lambda_i\}_{i=1}^R$ provide a notion of frequency for the graph setting [2]. If working with directed graphs, one can simply replace (2.1) by a Jordan decomposition $\mathbf{S} = \mathbf{U}\mathbf{J}\mathbf{U}^{-1}$ [10].

The vectors $\{\mathbf{u}_i\}_{i=1}^N$ provide a Fourier-like basis for graph signals, allowing to decompose any signal $\mathbf{x}$ into its spectral components $\mathbf{x}_\text{f} = \mathcal{F}\{\mathbf{x}\} = \mathbf{U}^H\mathbf{x}$, an operation known as the graph Fourier transform (GFT). In this sense, we say that a graph signal $\mathbf{x}$ is bandlimited [2] when $\mathbf{x}_\text{f}$ has $K < N$ non-zero entries.

### 2.2.1 Graph filters

In classical signal processing, filtering refers to the process of amplifying or attenuating different frequencies of a signal [11]. Exploiting the GFT frequency representation, one can perform an analogous operation on graph signals [2]. Indeed, filtering a graph signal $\mathbf{x}$ with a graph filter $H(\lambda)$ can be done as

$$\mathbf{y} = \mathbf{U}\,\text{diag}\{H(\lambda_1), \ldots, H(\lambda_N)\}\mathbf{U}^H\mathbf{x} = \mathcal{F}^{-1}\left\{\text{diag}\{H(\lambda_1), \ldots, H(\lambda_N)\}\mathcal{F}\{\mathbf{x}\}\right\}. \tag{2.2}$$

Nevertheless, opposed to what happens in classical signal processing, we do not have a fast implementation of the GFT, yet. For this reason, filtering a signal using (2.2)

can be very inefficient. The most common way to circumvent this problem is designing and implementing graph filters in the vertex domain [2], defining the finite impulse response (FIR) version of graph filters as a linear combination of powers of the GSO which exploit the sparsity pattern in $\mathbf{S}$.

**Definition 2.1** (FIR graph filter)**.** *Let $\mathbf{x} \in \mathbb{C}^N$ be a signal defined on a graph $\mathcal{G}$ equipped with a GSO $\mathbf{S} \in \mathbb{C}^{N \times N}$, and $\mathbf{h} \in \mathbb{C}^K$ be a vector that stores the $K$ coefficients of a graph FIR filter. Filtering $\mathbf{x}$ with $\mathbf{h}$ in the vertex domain can be performed doing*

$$\mathbf{y} = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x}. \tag{2.3}$$

*The universal spectral response of such a filter is*

$$H(\lambda) = \sum_{k=0}^{K-1} h_k \lambda^k. \tag{2.4}$$

As we will see, FIR graph filters can represent a vast class of graph filters which can be implemented using a $K$-hop message passing distributed algorithm. Furthermore, as we will see, they are key in the definition of efficient graph convolutional neural networks.

### 2.2.2 Product graphs

Oftentimes, large scale graphs appear as a product of several smaller graphs. For example, time series on a sensor network, can be factorized using a cycle or path graph to represent time, and a spatial mesh to represent the network [12]. In genomics, the graph that relates the different phenotypes of a population is a product of the graphs that relate the different character phenotypes [13]. And, the movie ratings on a platform like Netflix can be viewed as signals living on the product of the social network of the users and the graph of relations among movies [14]. More formally, we say that a graph is a product graph when its node set can be decomposed as a Cartesian product of the nodes of two smaller factor graphs, and its edges are related with a known connection to the edges of the factors [15].

As the number of nodes within each factor increases, the total number of nodes in the product graph grows exponentially. When this happens, the tools that were designed to process data on small networks start to fail. In literature, this issue is commonly known as the *curse of dimensionality*. To circumvent this issue, some authors have proposed exploiting the product structure of large graphs to parameterize graph signals with a reduced dimensionality [16].

Let $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ be two[1] undirected graphs with $N_1$ and $N_2$ vertices, respectively. The product graph [15], of $\mathcal{G}_1$ and $\mathcal{G}_2$ denoted with the symbol $\diamond$, is the graph given by

$$\mathcal{G}_\diamond = \mathcal{G}_1 \diamond \mathcal{G}_2 = (\mathcal{V}_\diamond, \mathcal{E}_\diamond),$$

[1]For the sake of exposition, we restrict ourselves to the product of two graphs. Extensions to higher-order product graphs are possible, although they require the use of tensor product formulation. Hence, we postpone this extension to Part II of this thesis.

where $\mathcal{V}_\diamond = \mathcal{V}_1 \times \mathcal{V}_2$ denotes the Cartesian product of the vertex sets, and $\mathcal{E}_\diamond$ defines a valid edge set for $\mathcal{V}_\diamond$ according to the rules of the graph product. Depending on the set of rules that determine $\mathcal{E}_\diamond$, three different product graphs are usually defined: the Cartesian product, the Kronecker product, and the strong product [cf. Fig. 2.2]. For details of these rules we refer the reader to [15]. The eigenvalue decomposition of the graph-shift operator of a product graph, denoted by $\mathbf{S}_\diamond$, is related to the eigenvalue decompositions of its factors through [16]

$$\mathbf{S}_\diamond = \mathbf{U}_\diamond \mathbf{\Lambda}_\diamond \mathbf{U}_\diamond^H = (\mathbf{U}_1 \otimes \mathbf{U}_2)\mathbf{\Lambda}_\diamond (\mathbf{U}_1 \otimes \mathbf{U}_2)^H,$$

where $\otimes$ denotes the Kronecker product between matrices, $\mathbf{U}_1$ and $\mathbf{U}_2$ are the eigenvectors of the graph-shift operators for $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively, and $\mathbf{\Lambda}_\diamond$ is some diagonal matrix that depends on $\mathcal{G}_1$, $\mathcal{G}_2$ and the type of product.

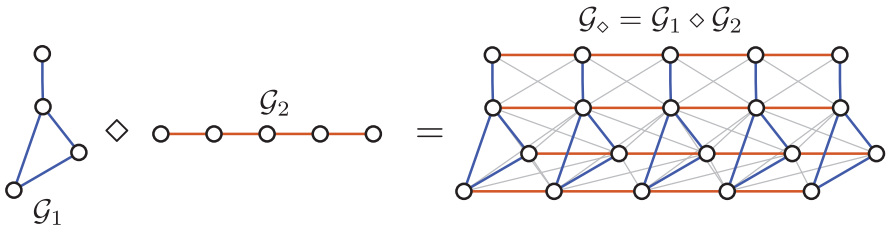



Figure 2.2: Illustration of the product between two graphs. $\diamond$ represents either a Cartesian (only colored edges), a Kronecker (only gray edges) or a strong product (all edges) between graphs.

**Example 2.1** (Graph Cartesian product)**.** *Throughout the thesis we will make special use of the Cartesian product of graphs, denoted by the symbol* $\square$*. This product is characterized by an edge set* $\mathcal{E}_\square$ *that contains an edge between two vertices in the product graph if there is an edge between the vertices in the factor graphs [cf. Fig. 2.2]; and a GSO that satisfies* $\mathbf{S}_\square = \mathbf{S}_1 \oplus \mathbf{S}_2 = \mathbf{S}_1 \otimes \mathbf{I}_2 + \mathbf{I}_1 \otimes \mathbf{S}_2$*, and hence* $\mathbf{\Lambda}_\square = \mathbf{\Lambda}_1 \oplus \mathbf{\Lambda}_2$*.*

Because every node in a product graph can be indexed using a pair of vertices of the factor graphs, we can rearrange any product graph signal $\mathbf{x} \in \mathbb{R}^{N_1 N_2}$ in a matrix form $\mathbf{X} \in \mathbb{R}^{N_2 \times N_1}$ such that $[\mathbf{x}]_{i+(j-1)N_2} = [\mathbf{X}]_{i,j}$. The spectral decomposition of $\mathbf{x}$ then takes the form

$$\mathbf{x} = \mathcal{F}^{-1}\{\mathbf{x}_\mathrm{f}\} = (\mathbf{U}_1 \otimes \mathbf{U}_2)\mathbf{x}_\mathrm{f} \iff \mathbf{X} = \mathcal{F}^{-1}\{\mathbf{X}_\mathrm{f}\} = \mathbf{U}_2 \mathbf{X}_\mathrm{f} \mathbf{U}_1^T. \tag{2.5}$$

Using this formulation, we can say that a product graph signal $\mathbf{X}$ is bandlimited when it is simultaneously bandlimited in both domains, i.e., with a sparse $\mathbf{X}_\mathrm{f} \in \mathbb{R}^{N_2 \times N_1}$ having $K_1 < N_1$ columns and $K_2 < N_2$ rows different than zero with known support. In such cases, $\mathbf{x}$ admits a low-dimensional representation as

$$\mathbf{x} = (\tilde{\mathbf{U}}_1 \otimes \tilde{\mathbf{U}}_2)\tilde{\mathbf{x}}_\mathrm{f} \iff \mathbf{X} = \tilde{\mathbf{U}}_2 \tilde{\mathbf{X}}_\mathrm{f} \tilde{\mathbf{U}}_1^T, \tag{2.6}$$

where $\tilde{\mathbf{U}}_1$ and $\tilde{\mathbf{U}}_2$ are obtained by removing the columns of $\mathbf{U}_1$ and $\mathbf{U}_2$ corresponding to the indices of the rows and columns of $\mathbf{X}_\mathrm{f}$ that are zero, respectively; and $\tilde{\mathbf{X}}_\mathrm{f}$ and $\tilde{\mathbf{x}}_\mathrm{f}$ are the non-zero spectral components in $\mathbf{X}_\mathrm{f}$ and $\mathbf{x}_\mathrm{f}$.

### 2.2.3 Time-varying graph signal processing

With the rapid development of GSP in the past years, many researchers have shift their focus to time-varying signals and processes that reside on graphs [12, 17–19].

These works model time-varying graph signals as signals living on the Cartesian product of an arbitrary graph $\mathcal{G}$ and a directed cycle graph $\mathcal{T}$, that represents time. This way, every sample of a time-varying graph process is indexed using a node of the cycle graph [cf. Fig. 2.3].

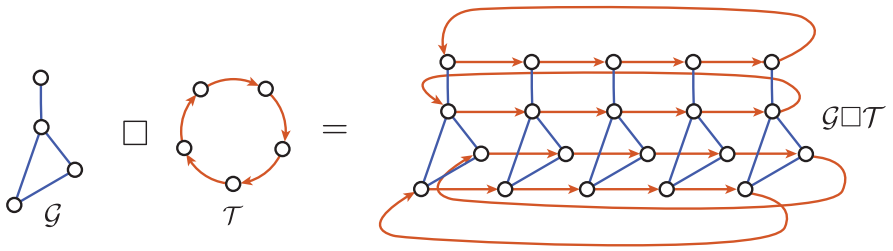



Figure 2.3: Support of time-varying graph periodic signal.

The joint time-vertex Fourier transform of a time-varying graph signal $\mathbf{X} \in \mathbb{R}^{N\times T}$ is defined as the Fourier transform of the product graph signal

$$\mathbf{X}_{\mathrm{f}} = \mathcal{F}_{\mathcal{G},\mathcal{T}}\{\mathbf{X}\} = \mathbf{U}_{\mathcal{G}}^{H}\mathbf{X}\mathbf{U}_{\mathcal{T}}^{*},$$

where $\mathbf{U}_{\mathcal{G}}$ represents the GFT eigenbase of $\mathcal{G}$, and $\mathbf{U}_{\mathcal{T}}$ the GFT eigenbase of $\mathcal{T}$, which takes the form of a discrete Fourier transform (DFT) matrix.

Likewise, we can also define the Fourier transforms with respect to each particular domain, i.e., either vertex or time, as

$$\mathbf{X}_{\mathrm{f}_{\mathcal{G}}} = \mathcal{F}_{\mathcal{G}}\{\mathbf{X}\} = \mathbf{U}_{\mathcal{G}}^{H}\mathbf{X},$$
$$\mathbf{X}_{\mathrm{f}_{\mathcal{T}}} = \mathcal{F}_{\mathcal{T}}\{\mathbf{X}\} = \mathbf{X}\mathbf{U}_{\mathcal{T}}^{*},$$

where the underlying joint spectral structure is not exploited.

## 2.3 Geometric deep learning

The term geometric deep learning was recently coined in [3] to refer to the recent developments in the deep learning field that allow the generalization of the popular convolutional neural networks to non-Euclidean domains. This topic has attracted a lot of attention among researchers from different fields that are now trying to mimic the extreme success that convolutional neural networks had on the computer vision community in the past few years. Nevertheless, this heterogenous mixture of research lines has produced a very diverse set of competing ideas on how to generalize convolutions to graphs and manifolds.

As a consequence of the lack of shift invariance, vector space structure, or a common system of coordinates in non-Euclidean domains there is no clear way to define a convolution operation for graphs [3]. Depending on the field of origin, different researchers have come up with different analogies for this operation. In particular, most

of the work has been done in trying to achieve generalizations that have the following properties (proposed in [3]):

- Locality in the vertex domain, i.e., information is combined taking into account the neighborhood of each node.
- Fixed number of parameters with respect to the input size.
- Low computational complexity.
- Ability to generalize to different graphs using the same trained coefficients.

In the following we briefly review the different approaches that have been published on this matter.

### 2.3.1 Spectral convolutional layer

The first convolutional layer definition was proposed by Bruna et al. in [20] who translated the well-known convolution theorem to non-Euclidean domains.

**Definition 2.2** (Spectral convolution [20])**.** *Let* $\mathbf{f} \in \mathbb{C}^N$ *and* $\mathbf{g} \in \mathbb{C}^N$ *denote two signals defined on the vertices of a graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*, with* $|\mathcal{V}| = N$*, then we can define a graph convolution operator* $*_{\mathcal{G}}$ *as*

$$\mathbf{f} *_{\mathcal{G}} \mathbf{g} = \mathcal{F}_{\mathcal{G}}^{-1} \{\mathbf{f}_{\mathrm{f}} \circ \mathbf{g}_{\mathrm{f}}\} = \mathbf{U} \left( (\mathbf{U}^H \mathbf{f}) \circ (\mathbf{U}^H \mathbf{g}) \right), \tag{2.7}$$

*where* $\mathbf{U}$ *denotes the matrix containing the eigenvectors of the GSO, and* $\circ$ *represents the Hadamard, or element-wise product between vectors.*

Using this definition, Bruna created the following convolutional layer:

**Definition 2.3** (Spectral convolutional layer [20])**.** *Let* $\mathbf{x}^{(i)} \in \mathbb{C}^N$ *be a graph signal defined on* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *that represents the input of the* $i$*-th layer of a graph convolutional neural network (GCNN). We define a spectral convolutional layer as*

$$\mathbf{x}^{(i+1)} = \rho \left( \mathbf{h} *_{\mathcal{G}} \mathbf{x}^{(i)} \right) = \rho \left( \mathbf{U} \left( \mathbf{h}_{\mathrm{f}} \circ \mathbf{U}^H \mathbf{x}^{(i)} \right) \right), \tag{2.8}$$

*where* $\mathbf{h}_{\mathrm{f}} = \mathbf{U}^H \mathbf{h}$ *is a set of weights to learn during training and* $\rho(\cdot)$ *a point-wise non-linearity, e.g. sigmoid, rectified linear unit, hyperbolic tangent, etc. [5].*

Even though Definition 2.2 was the breakthrough that triggered a series of very important discoveries in the field, (2.8) does not satisfy any of the properties desired for a convolutional layer. In particular, such a layer requires $\mathcal{O}(N)$ parameters, has a high computational complexity due to the eigenvalue decomposition of the GSO (i.e. $\mathcal{O}(N^2)$), is in principle non-local in the vertex domain, and does not generalize to inputs defined on different graphs, as its parameters depend on the specific eigendecomposition of a particular GSO.

To circumvent some of these problems, namely, the $\mathcal{O}(N)$ scaling of the number of parameters and the non-locality in the vertex domain, Bruna proposed in [20] an alternative version of (2.8) based on a cubic spline parameterization of the spectral filters. This is, it defined a convolutional layer of the form:

**Definition 2.4** (Smooth spectral convolutional layer [20]). *Let $\mathbf{B} \in \mathbb{C}^{N\times K}$ be a fixed cubic spline kernel and $\boldsymbol{\alpha} \in \mathbb{C}^K$ a fixed-size vector of spline coefficients to learn, we define a smooth spectral convolutional layer as*

$$\mathbf{x}^{(i+1)} = \rho\left(\mathbf{h} *_{\mathcal{G}} \mathbf{x}^{(i)}\right) = \rho\left(\mathbf{U}\left(\mathbf{B}\boldsymbol{\alpha} \circ \mathbf{U}^T\mathbf{x}^{(i)}\right)\right). \tag{2.9}$$

Indeed, this architecture has a fixed number of $K$ parameters. Furthermore, Bruna claims that, in analogy with the classical Fourier transform, the Parseval identity ensures the spatial localization, i.e., a limited spatial support, of the graph filters due to their smoothness in the frequency domain[2]. However, besides this analogy, this claim is not supported by any proof. A GCNN with this architecture has successfully been used to solve the community detection problem on social networks [21].

### 2.3.2 Spectrum-free convolutional layer

Despite the clear advantages of (2.9) over (2.8), this convolutional layer still has a great computational complexity and it depends on a particular Laplacian. Therefore several authors tried to overcome these issues by performing the convolution in the vertex domain. In particular, they focused on the use of FIR vertex-domain graph filters [2] as the framework for their definitions of convolutional layers.

This was first proposed by Deferrad et al. in [22], where they defined a convolutional layer using FIR filtering of Chebyshev polynomials of the graph Laplacian.

**Definition 2.5** (Chebyshev convolutional layer [22]). *Let $\mathbf{L}$ be any graph Laplacian with eigenvalues $\lambda_n \in [0, \lambda_{\max}]$, and let $T_k(\lambda)$ denote a Chebyshev polynomial basis of order $k$. Then, we can define a Chebyshev convolutional layer as*

$$\mathbf{x}^{(i+1)} = \rho\left(\mathbf{H}(\tilde{\mathbf{L}})\mathbf{x}^{(i)}\right) = \rho\left(\sum_{k=0}^{K-1} h_k T_k(\tilde{\mathbf{L}})\,\mathbf{x}^{(i)}\right), \tag{2.10}$$

*where $\{h_k\}_{k=0}^{K-1}$ are $K$ free parameters that need to be learnt during training and*

$$\tilde{\mathbf{L}} = \frac{2}{\lambda_{\max}}\mathbf{L} - \mathbf{I}.$$

With this definition, the problems present in (2.9) are solved: On the one hand, (2.10) does not require to compute a computationally expensive eigendecomposition of the graph Laplacian, as filtering is performed directly on the vertex domain. And on the other hand, the recursive nature of the Chebyshev polynomials allows for a fast implementation of (2.10) that relies on efficient sparse matrix-vector multiplications with complexity $\mathcal{O}(N)$ .

Furthermore, the fact that applying a function to a diagonalizable matrix is the same as applying the function to each of its eigenvalues allows us to write

$$T_k(\tilde{\mathbf{L}}) = \mathbf{U}T_k(\tilde{\mathbf{\Lambda}})\mathbf{U}^H = \mathbf{U}\begin{bmatrix} T_k(\tilde{\lambda}_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & T_k(\tilde{\lambda}_N) \end{bmatrix}\mathbf{U}^H,$$

[2]Signals parameterized by cubic splines are indeed smooth.

which implies that we can also write (2.10) as a universal filter on the eigenvalues of a general graph Laplacian

$$H(\lambda) = \sum_{k=0}^{K-1} h_k T_k \left( \frac{2}{\lambda_{\max}} \lambda - 1 \right),$$

thus, meaning that this type of filter generalizes to inputs of different Laplacians. Nevertheless, in a typical neural network several convolutional layers like the one in (2.10) would be stacked to form a deep sequence of non-linear operations, and there is no evidence guaranteeing that the overall response of a multilayer GCNN behaves the same way with input signals that come from different graphs. This has in fact lead to the speculation [3] that convolutional layers of the form (2.10) might behave poorly when used with different graphs. Anyway, Chebyshev GCNNs have successfully been applied to tackle different problems in recommender systems [23], natural language processing [24], and neuroscience [25].

The Chebyshev convolutional layer is not, however, the only convolutional layer defined on the basis of an FIR graph filter. Other authors have also proposed their own versions that, in principle, work better in different scenarios. Such is the case of the node-varying convolutional layer defined by Gama et al. in [26], which allows to define a different set of weights per node, therefore, approximating a more general class of linear functions; or the first-order FIR convolution of Kipf et al. [27, 28] who claimed that a filter order of $K = 2$ is enough to cope with real datasets on different tasks.

### 2.3.3 Spatial domain convolutional layer

The last type of convolutional layers do not make use of the spectral intuition behind the convolution theorem [29–31]. Instead, they propose to mimic the filter matching behavior of classic CNNs and define a local template that is passed at each point of the domain and computing the correlation of the signal with the template. Nevertheless, since there is no clear way to define a patch-extracting operator on graphs, many different implementations of this idea have been proposed, most of them trying to reduce the computational burden of extracting the domain patches, and defining a new set of generalized coordinates that can be used across different domains. In this thesis, however, we will not focus on this approach, and leave the generalization of this type of convolutional layer to time-varying signals for future work.

### 2.3.4 Time-varying geometric deep learning

The ability to generalize GCNNs to time-varying signals is still an open area of research, as it has not yet attracted a lot of attention from the geometric deep learning community. Most of the research effort has focused on combining some type of GCNNs with recursive neural network (RNNs) architectures [5] which nowadays are the state-of-the-art models for natural language processing and deep learning applied to acoustic signals.

These networks have been applied to solve problems in computer graphics [32], natural language processing [33] and traffic forecasting [34]; but lack a general architecture

and are far from reaching a framework consensus. In fact, to our knowledge, we are the first to propose an extension of the convolutional layer to jointly process time-varying graph signals which might serve as the building block for a general treatment of time-varying graph processes.

# 3 Problem modeling

In this chapter, we define a natural extension of the spectrum-free graph convolutional neural networks that allows us to generalize the theory of geometric deep learning to product graph signals. In particular, we will define a new type of 2D-FIR graph filters that allow to jointly filter graph signals on multiple irregular domains, and to build localized convolutional layers that have a fixed number of training parameters and a low computational complexity. For the case when one of the domains is time, we will see that this new type of filters matches the most general form of the time-vertex FIR graph filters defined by Isufi et al. [17].

## 3.1 2D-FIR graph convolutional layer

Let $\mathbf{X} \in \mathbb{C}^{N_2 \times N_1}$ be a product graph signal defined on the product of $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$. Recall that all convolutional layers in Section 2.3 define graph convolutions on column signals. Hence, we can extend these definitions to perform a joint convolution on the rows and columns of $\mathbf{X}$. We can filter the rows of $\mathbf{X}$ with a 1D-FIR filter by filtering the columns of the transpose of $\mathbf{X}$ as

$$\mathbf{Y} = \left( \sum_{m=0}^{M-1} h_m \mathbf{S}_1^m \mathbf{X}^T \right)^T = \sum_{m=0}^{M-1} h_m \mathbf{X} \mathbf{S}_1^m, \tag{3.1}$$

where $\{h_m\}_{m=0}^{M-1}$ is a set of 1D-FIR coefficients. Combining a row and a column convolution we can jointly filter a multidomain graph signal using the following 2D-FIR filter.

**Definition 3.1** (2D-FIR graph filter)**.** *Let $\{h_{k,m}\}$ for $k = 0, \dots, K-1$ and $m = 0, \dots, M-1$ ($\mathbf{H} \in \mathbb{C}^{K \times M}$ in matrix notation) be a set of 2D-FIR filter coefficients. We can jointly filter $\mathbf{X}$ in its two vertex domains by computing*

$$\mathbf{Y} = \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} h_{k,m} \mathbf{S}_2^k \mathbf{X} \mathbf{S}_1^m. \tag{3.2}$$

*The spectral response of such a filter is*

$$H(\lambda_1, \lambda_2) = \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} h_{k,m} \lambda_1^m \lambda_2^k. \tag{3.3}$$

As happened with the unidimensional FIR graph filter, the 2D-FIR graph filter is the perfect candidate to define a multidomain graph convolutional neural network (2D-GCNN). They are parameterized by a fixed number of $K \times M$ parameters, regardless

of the number of nodes in each domain. They are local in both domains, since they inherent the locality properties of the unidimensional graph convolutions. And they have a universal time-vertex spectral response that is independent of the GSO.

For these reasons, we propose the following convolutional layer to process multidomain graph signals.

**Definition 3.2** (2D-GCNN)**.** *We define a multidomain graph convolutional layer as*

$$\mathbf{X}^{(i+1)} = \rho\left(\sum_{k=0}^{K-1} h_{k,m} T_k(\tilde{\mathbf{L}}_2)\mathbf{X}^{(i)} T_m(\tilde{\mathbf{L}}_1)\right), \tag{3.4}$$

*where* $\{h_{k,m}\}$ *for* $k = 0, \dots, K-1$ *and* $m = 0, \dots, M-1$ *are a set of trainable coefficients.*

## 3.2 Time-vertex graph convolutional layer

For the special case when one of the two domains is time, our proposed 2D-FIR graph filter is reduced to the time-vertex FIR filter proposed by Isufi et al. [17]. In particular, their implementation takes advantage of the special structure of time to define filters that do not require computing powers of the temporal GSO.

Let $\mathbf{x}_t \in \mathbb{C}^N$ be the $t$-th time sample of a finite support temporal graph signal defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and let us create a matrix $\mathbf{X} \in \mathbb{C}^{N \times T}$ that stores the $T$ samples of $\mathbf{x}_t$ as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \dots & \mathbf{x}_{T-1} \end{bmatrix}.$$

We can write the temporal convolution of such a signal $\mathbf{x}_t$ and a temporal FIR filter $h_t$ of order $M$ ($\mathbf{h} \in \mathbb{C}^M$ in vector notation) as

$$\mathbf{y}_t = \mathbf{x}_t *_{\mathcal{T}} h_t = \sum_{m=0}^{M-1} h_m \mathbf{x}_{t-m}. \tag{3.5}$$

Assuming zero-padding, we can also write the convolution using matrix notation as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \dots & \mathbf{y}_{T-1} \end{bmatrix} = \mathbf{X} \begin{bmatrix} h_0 & \dots & h_{M-1} & 0 & & \dots & 0 \\ 0 & h_0 & \dots & h_{M-1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 & h_0 & \dots & h_{M-1} \\ \vdots & \ddots & \ddots & & & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 & h_0 \end{bmatrix} = \mathbf{X}\underline{\mathbf{H}}, \tag{3.6}$$

where $\underline{\mathbf{H}} = \text{Conv}\{\mathbf{h}\} \in \mathbb{R}^{T \times T}$ is a Toeplitz matrix where each row is a zero-padded shifted copy of the filter coefficients $\mathbf{h} = [h_0, \dots, h_{M-1}]^T$. Hence, the convolution is performed on each row of $\mathbf{X}$ individually. At this point, we see that (3.6) is simply a particularization of (3.1) where $\mathbf{S}_1$ is a shift matrix.

As we did in (3.2) we can combine a 1D-FIR graph filter with this formulation to define a time-vertex FIR graph filter.

**Definition 3.3** (Time-vertex FIR filter [17]). *We can jointly filter* $\mathbf{X}$ *in the time and vertex domain by computing*

$$\mathbf{y}_t = \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} h_{k,m} \mathbf{S}^k \mathbf{x}_{t-m}, \tag{3.7}$$

*or alternatively in matrix notation*

$$\mathbf{Y} = \sum_{k=0}^{K-1} \mathbf{S}^k \mathbf{X} \underline{\mathbf{H}}_k, \tag{3.8}$$

*where* $\underline{\mathbf{H}}_k = \mathrm{Conv}\{\mathbf{h}_k\}$*, being* $\mathbf{h}_k = [h_{k,0}, \ldots, h_{k,M-1}]^T$ *a set of filter coefficients for* $k = 0, \ldots, K-1$*. The spectral response of such a filter can be obtained by taking the joint time-vertex Fourier transform of* (3.7)

$$H(\omega, \lambda) = \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} h_{k,m} \lambda^k e^{-j\omega m}. \tag{3.9}$$

Since in the rest of the learning part we will focus on time-varying graph signals, we present the particularization of the 2D-GCNN to this type of signals.

**Definition 3.4** (TV-GCNN). *We define a time-varying graph convolutional layer as*

$$\mathbf{X}^{(i+1)} = \rho\left( \sum_{k=0}^{K-1} T_k(\tilde{\mathbf{L}}) \mathbf{X}^{(i)} \underline{\mathbf{H}}_k \right), \tag{3.10}$$

*where* $\underline{\mathbf{H}}_k = \mathrm{Conv}\{\mathbf{h}_k\}$*, being* $\mathbf{h}_k = [h_{k,0}, \ldots, h_{k,M-1}]^T$ *a set of trainable coefficients for* $k = 0, \ldots, K-1$*.*

Finally, to prove the versatility of this type of filters, Fig. 3.1 shows a collection of spectral responses obtained by randomly setting the coefficients of a 2D-FIR graph filter ($K = 3$ and $M = 3$). The diversity of spectral patterns in this picture proves that the 2D-FIR graph filters can implement a wide range of frequency responses which can focus on very specific bands of the spectrum.

In the next chapter, we will see how we can implement this kind of layer on a GPU. Thus, allowing us to train very deep TV-GCNNs.

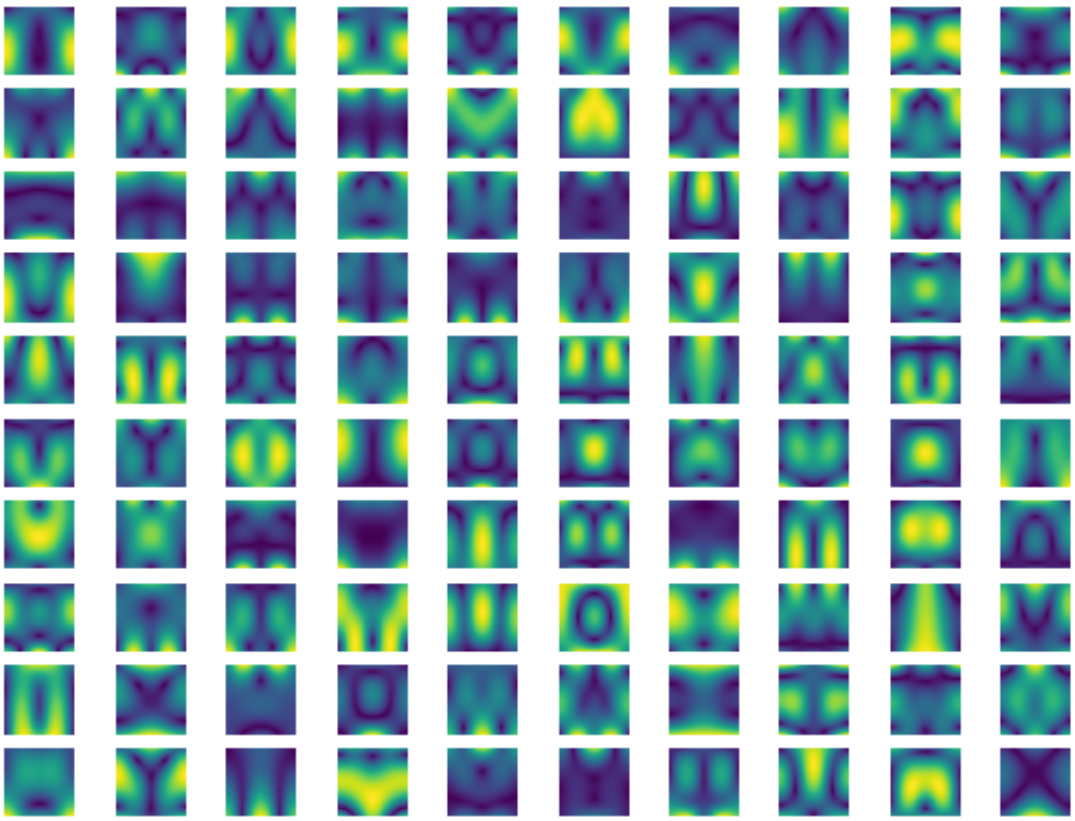


Figure 3.1: Collection of spectral time-vertex responses of 2D-FIR graph filters with random coefficients ($K = 3$ and $M = 3$). The horizontal direction represents the symmetric (real coefficients) temporal frequencies, and the vertical direction the graph frequencies.

# Implementation and numerical results

# 4

One of the greatest breakthroughs in the recent history of deep learning has been the development of open source libraries, such as Tensorflow [35], that allow to build, train and test complex neural network architectures using a very high-level API. In this sense, most modern deep learning libraries have been optimized to run on GPUs: the highly parallelizable computations needed to train a CNN can be run efficiently on machines equipped with GPUs. As a result, computational complexity in deep learning refers more to the suitability of implementation on a GPU using some of the standard libraries, than to the classic theory of algorithmic complexity.

Bearing this in mind, in this section, we address the complexity of our proposed TV-GCNN and study an efficient way to parameterize this type of layer such that it is amenable to be implemented using deep learning libraries. In particular, since most deep learning libraries use a computational framework based on highly parallelizable tensor product operations[1], we will propose an alternative formulation of (3.10) based on tensor computations.

Next, we will evaluate the performance of our proposed TV-GCNN when solving graph signal classification tasks. To do so, we will compare the classification accuracy of two deep GCNNs; a 1D-GCNN with convolutions based on the Chebyshev convolutional layer[2] [22], and one based on our proposed time-varying graph convolutional layer (TV-GCNN); on a synthetic dataset we built to assess the generalization abilities of such networks, as well as their robustness against noise.

## 4.1 Tensorization and GPU implementation

In classical image CNNs, each layer's input is treated as a 3D tensor made out of a set of images (or feature maps) that contain an abstract representation of the input. In the first layer, these feature maps correspond to the three color channels of an image, i.e., red, green and blue. A convolutional layer, thus, takes a 3D tensor as input of dimensions width $\times$ height $\times$ channels ($W \times H \times C$) and outputs another 3D tensor of size width $\times$ height $\times$ features ($W \times H \times F$). Hence, in each layer there are $F$ filters of size $K \times M \times C$ that convolve a $K \times M$ kernel with each of the $C$ channels and sum the result to give one of the $F$ feature maps. Thus, every layer has $K \times M \times C \times F$ trainable parameters that can be stored in a tensor $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{K \times M \times C \times F}$.

The same happens in TV-GCNNs: Each layer takes a 3D tensor $\boldsymbol{\mathcal{X}}^{(i)} \in \mathbb{R}^{N \times T \times C}$ representing a time-varying graph signal with $N$ vertices, $T$ snapshots and $C$ features,

[1] This is analogous to Matlab's computational framework based on matrices rather than loops. Since matrix-vector products are internally optimized by the Matlab interpreter, a proficient Matlab programmer would generally exploit this data structures in its code.

[2] The GCNNs based on Chebyshev convolutional layers have shown to perform best on a wide range of applications, and have the most sound theoretical foundation.

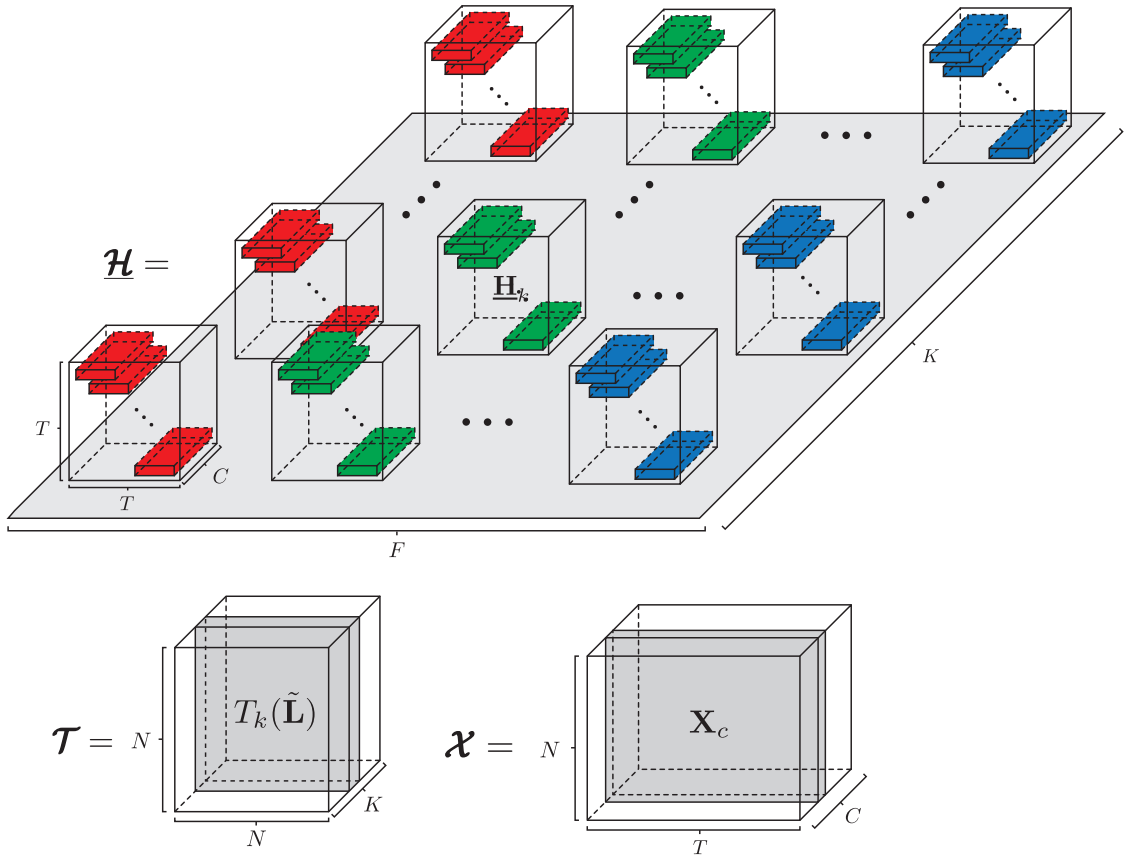



Figure 4.1: Diagram representing the proposed tensorization.

e.g., meteorological records consist of time variations of wind speed, humidity, and temperature across different cities which can be connected by a graph; and outputs another 3D tensor $\boldsymbol{\mathcal{X}}^{(i+1)} \in \mathbb{R}^{N\times T\times F}$. Hence, each layer has a set of $F$ parallel filters with $K \times M \times C$ parameters that are convolved using a 2D-FIR graph filter of order $K$ in the vertex domain, and order $M$ in the time domain with each of the $C$ channels of the input. Later, the convolved channel images are summed up to form each of the $F$ feature maps. Thus, every layer has $K \times M \times C \times F$ trainable parameters that can be stored in a tensor $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{K\times M\times C\times F}$.

Taking this into account, we propose the following tensorization of (3.10).

**Definition 4.1** (Tensor TV-GCNN)**.** *Let $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{K\times N\times N}$ be a tensor formed by stacking the individual terms of a Chebyshev polynomial of order $K$ of the graph Laplacian, such that $\boldsymbol{\mathcal{T}}(k,:,:) = T_k(\tilde{\mathbf{L}})$, and let $\underline{\boldsymbol{\mathcal{H}}} \in \mathbb{R}^{K\times T\times T\times C\times F}$ be a tensor formed by arranging the trainable parameters $\boldsymbol{\mathcal{H}}$ such that $\underline{\boldsymbol{\mathcal{H}}}(k,:,:,c,f) = \mathrm{Conv}\left\{\boldsymbol{\mathcal{H}}(k,:,c,f)\right\}$ [cf. Fig. 4.1]. The tensorized version of the time-varying graph convolutional layer for an input with $C$ channels and an output of $F$ feature maps is*

$$\boldsymbol{\mathcal{X}}^{(i+1)} = \rho\left(\left[\boldsymbol{\mathcal{T}} \bullet_1^3 \boldsymbol{\mathcal{X}}^{(i)}\right] \bullet_{3,1,4}^{3,1,4} \underline{\boldsymbol{\mathcal{H}}}\right), \tag{4.1}$$

*where $\bullet_m^n$ represents the generalized tensor contraction operation [36], consisting of the application of an inner product to the $n$th mode of the left hand side tensor with the $m$th*

*mode of the right hand side tensor. Stacked indices, represent sequential applications of the generalized contraction.*

Tensor contractions are a basic operation in most deep learning libraries[3], and as such, they are highly optimized for an efficient performance on GPUs. Since (3.10) does mainly consist of sequential applications of this operation, we can guarantee that this implementation of TV-GCNN can be run efficiently on most deep learning platforms.

**Remark.** *One can also further optimize* (3.10) *by, instead of performing the second tensor contraction with* $\underline{\boldsymbol{\mathcal{H}}}$ *corresponding to the convolution in time, using the built-in 1D convolution operator contained in most deep learning libraries. The computational improvement that this approach might have depends on the specific platform implementation, though.*

## 4.2 Construction of the synthetic dataset

As we mentioned in the introduction, we are interested in classifying signals that live on a multidomain support composed of a spatial domain (graph) and a temporal domain (cycle). Hence, our synthetic dataset consists of a set of signals that show different types of spatio-temporal correlations. In particular, we propose to generate different classes of time-varying graph signals using the model shown in Fig. 4.2.

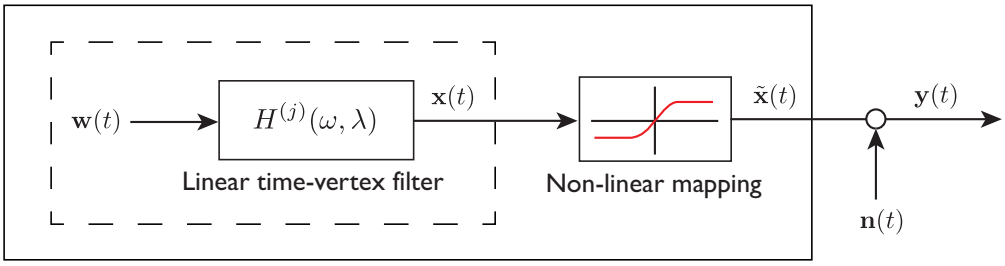



Figure 4.2: Generative model of synthetic time-varying graph signals.

The generative model works as follows: For a given graph $\mathcal{G}$, we first filter (in the joint time-graph spectral domain) a white Gaussian spatio-temporal signal $\mathbf{w}(t)$ by a class-specific linear filter whose response is chosen to model a certain pattern of spatio-temporal correlations. Then, we pass the resulting signal $\mathbf{x}(t)$ through a non-linear point-wise mapping that spreads the energy of $\mathbf{x}(t)$ to different frequencies. Finally, we add additive white Gaussian noise $\mathbf{n}(t) \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ to obtain the final signal $\mathbf{y}(t)$. This model, allows us to generate a synthetic dataset with $J$ classes by simply defining $J$ different filters $\{H^{(j)}(\omega, \lambda)\}_{j=1}^{J}$ in the time-graph spectral domain. Hence, the target inference task is the classification of the statistics of $J$ types of time-varying graph processes.

To generate the dataset in our experiments we use a community graph[4] with

[3]For instance, `tf.tensordot` or `tf.einsum` in Tensorflow [35].

[4]This type of random graph is characterized by having a strong community structure with very connected clusters and a few sparse connections between the communities. In our experiments it was generated using PyGSP [37].

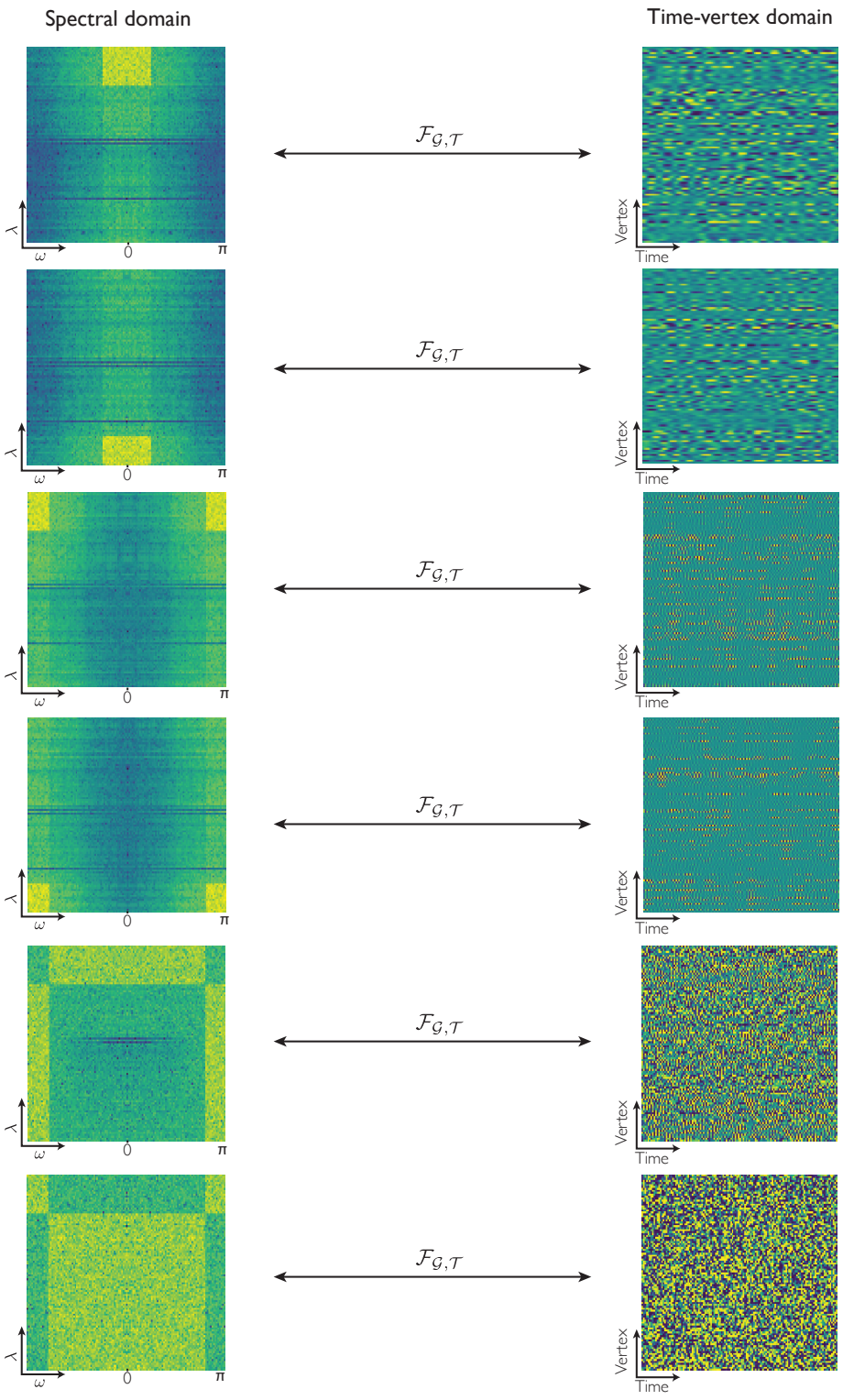



Figure 4.3: Time-varying graph process realization examples for each of the $J = 6$ classes used in our experiments. The GCNN receives the time-vertex representation.

| Type of layer | Layer parameters |
|---|---|
| TV-FIR (ReLU) | K=3, M=3, C=1, F=8 |
| Max-pooling | Time = 4, Vertex = 2 |
| TV-FIR (ReLU) | K=3, M=3, C=8, F=16 |
| Max-pooling | Time = 4, Vertex = 2 |
| TV-FIR (ReLU) | K=3, M=3, C=16, F=32 |
| Max-pooling | Time = 4, Vertex = 2 |
| Fully connected (softmax) | F = 6 |
| #Parameters=11,214 | |

Table 4.1: List of parameters involved in the architecture of the TV-GCNN.

$N = 100$ nodes to model the spatial support of our signal and produced time-varying realizations of $T = 128$ frames, i.e., each input sample has 12,800 features. We also define $J = 6$ different classes of time-vertex filters [cf. Fig. 4.3], and used the hyperbolic tangent function as point-wise non-linearity. An example of signal realizations for the six classes in the spectral and time-vertex domains for the noiseless case is shown in Fig. 4.3. It is important to highlight, that in the noiseless case due to the effect of the hyperbolic tangent mapping, the maximum absolute amplitude of the signal is clamped to 1.

## 4.3 Architecture definition

The architectures of the GCNNs that we compare in our experiments are described in Tables 4.1 and 4.2. As we can see, both networks share the same overall structure: they are composed of three graph convolutional layers, activated by a rectified linear unit (ReLU), and one fully connected layer, activated by a softmax operation [5]. The main differences in the architectures are the choices for the graph convolution operators and the order of the graph filters, which is modified so that the number of parameters in both networks is of the same order.

In both cases, the pooling layers [5] are implemented using the Graclus algorithm [38] as proposed in [22]. However, in the TV-GCNN, a max-pooling stage is also introduced for the temporal dimension since we also perform convolutions on this domain. Each pooling layer is followed by a batch normalization step [5], a standard trick in deep learning that has shown to improve the convergence speed of the training algorithms. This is normalizing the output of each hidden layer to zero mean and unit variance using online estimates of the mean and variance of the training set.

Training is performed by minimizing the cross entropy between the one-hot encoded labels and the output of the fully connected layer using an Adam optimizer [39]: a stochastic-gradient-descent-based optimization algorithm that adapts the learning rate depending on the dynamics of the optimization. The initial learning rate is set to $10^{-4}$.

| Type of layer | Layer parameters |
|---|---|
| 1D-FIR (ReLU) | K=4, C=1, F=8 |
| Max-pooling | Vertex = 2 |
| 1D-FIR (ReLU) | K=4, C=8, F=16 |
| Max-pooling | Vertex = 2 |
| 1D-FIR (ReLU) | K=4, C=16, F=32 |
| Max-pooling | Vertex = 2 |
| Fully connected (softmax) | F = 6 |
| #Parameters=9,734 | |

Table 4.2: List of parameters involved in the architecture of the 1D-GCNN.

We train all experiments for 10 epochs, i.e., we perform 10 passes through the whole training set.

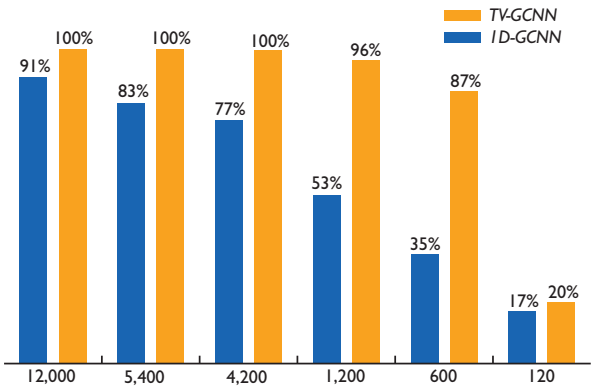
All architectural design decisions are taking in order to optimize the performance of both networks on a validation set consisting of 2,400 samples (400 samples per class). The deep learning library used for this implementation is Tensorflow [35], and, in particular, we extend the code from [22] to implement the Chebyshev graph convolutional layers and graph pooling layers. The code used for this part of the thesis is available in https://gitlab.com/gortizji/tv-graph-cnn. Here, you can find an efficient implementation of the 2D-FIR graph convolutional layer using the tensorization techniques explained in Section 4.1, as well as basic functions to build a synthetic dataset and test it using the proposed GCNN architectures.

## 4.4 Performance evaluation

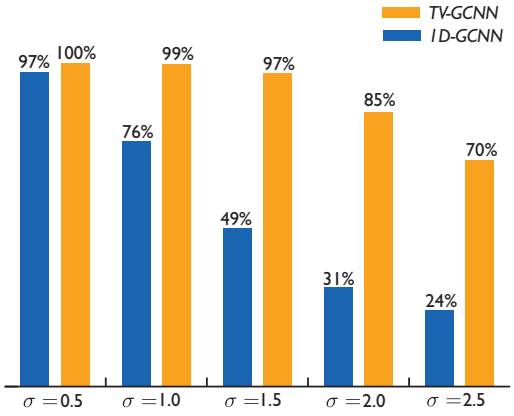
To compare the performance of these architectures, namely, 1D-GCNN and TV-GCNN, we evaluate the accuracy of both networks trained on our synthetic dataset and under different conditions.

First, we evaluate the generalization abilities of the GCNNs with respect to the training data available. To do so, we train both networks with different numbers of samples and test their accuracy on a new test set of 2,400 random samples per experiment that are equally distributed among classes. The noise level is set to $\sigma = 0.75$ for all experiments. Every experiment is repeated 6 times for each training size and the results are averaged to obtain an estimate of the generalization errors of both networks. As we can see in Fig. 4.4a, when there is abundant training data available both networks perform quite well, even though the 1D-GCNN scores a bit lower. However, as we decrease the training set size we can clearly see how the performance of the 1D-GCNN is rapidly deteriorated, while the TV-GCNN maintains its performance. The performance gap between the two networks peaks when training with 100 samples per class reaching an accuracy difference of more than 50 percentage points. Only when there is as little

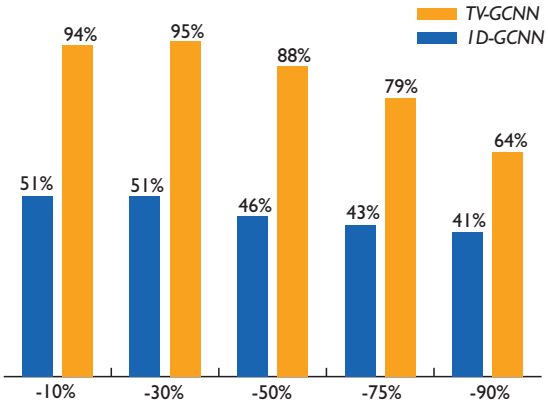
as 20 samples per class for training does the TV-GCNN fail to generalize to new and unseen data. Thus, highlighting that exploiting the joint correlations between space and time in a structured manner leads to much better generalization abilities.




(a) Test accuracy vs training set size.




(b) Test accuracy vs noise level.




(c) Test accuracy vs missing data.

Figure 4.4: Comparison of the generalization ability of the 1D-GCNN and the TV-GCNN for different training set sizes and noise levels.

In the second round of experiments, we test the robustness of both networks to noisy data. For this reason, we fix the training set size to 12,000 and train both GCNNs for different values of $\sigma$. Again, we repeated every experiment 6 times for each value of $\sigma$ and averaged the results to estimate the test accuracy of both architectures. In Fig. 4.4b, we see how the TV-GCNN is much more robust against additive noise than the 1D-GCNN. Even for very negative SNRs (recall that signals are clamped between $\pm 1$ after the hyperbolic tangent) we can see that the TV-GCNN is able to classify the statistics of the time-varying signals with very high accuracy. The maximum performance gap between the two networks, achieved when the SNR is very low, reaches again the 50 percentage points.

Finally, in the third round of experiments, we test the robustness of both networks against missing data. To do so, we fix the training set size to 1,200 and $\sigma$ to 0.75 and train and evaluate both networks randomly setting a percentage of the sample entries

to 0. Fig. 4.4c shows the results of these experiments (averaged from 6 different runs). In this case, even for very small percentages of removed entries the TV-GCNN doubles the accuracy of the 1D-GCNN, achieving very good performance even when removing more than half of the sample entries.

In light of these results, and reminding that the only difference between both networks is the choice of the convolutional layer, we can claim that the use of the TV-GCNN has the potential to extend the success of GCNNs to time-varying graph signals.

# Part II
# Sampling

# 5 Background

In Part II, we focus on the reconstruction of graph signals that reside on the vertices of a product graph by just observing a small subset of its vertices. In particular, we propose using a structured sampling scheme with which we select a sparse subset of nodes from each factor, thereby observing the signal at a few specific nodes of the product graph. This approach contrasts with traditional graph signal processing methods which do not take into account any underlying graph factorization when designing the sampling set [40–45]. When the underlying graph factorization is not accounted for, the complexity of designing the sampling set scales with the total number of vertices in the graph, and therefore, the applicability of such methods to large graphs is very limited.

Our proposed scheme circumvents this issue by reducing the original product search space into the union of two much smaller spaces. Hence, avoiding the curse of dimensionality. An illustration of this is shown in Fig. 5.1, where we see that selecting nodes from the factors reduces the possible candidate locations from 20 to 9. In essence, the aim is to reconstruct a signal on the product graph (rightmost in Fig. 5.1) by observing a subset of nodes of the factor graphs (on the left of Fig. 5.1) that generate the product graph.

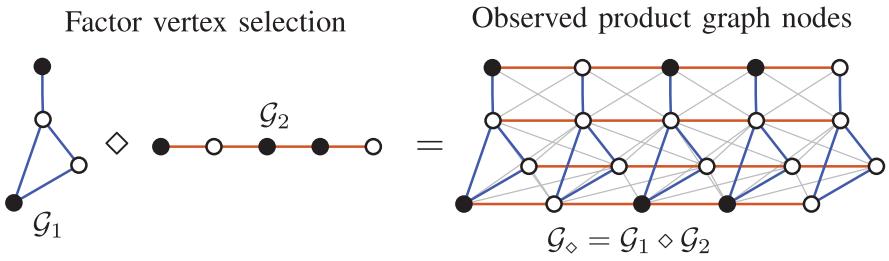



Figure 5.1: Illustration of the proposed sampling scheme for a product of two graphs. The black (white) dots represent the selected (unselected) vertices.

Nevertheless, we will not restrict ourselves only to multidomain signals defined on irregular domains, since, as we will see, the multidomain sparse sensing framework can be kept more general. Nonetheless, this particularization to multidomain graph signals will also be presented in Chapter 9 when we showcase our framework on some real dataset. In general, we will consider multidomain signals which can be represented using tensors and a multilinear system model.

In this chapter, we introduce the notation that will be used throughout Part II, as well as some preliminary notions of tensor algebra, multilinear systems and submodular optimization.

## 5.1 Notation

In Part II, we will denote product of variables/sets using a tilde superscript, i.e., $\tilde{N} = \prod_{i=1}^{R} N_i$, or $\tilde{\mathcal{N}}_i = \mathcal{N}_1 \times \cdots \times \mathcal{N}_R$; and drop the tilde to denote sums (unions) of the same variables/sets, i.e., $N = \sum_{i=1}^{R} N_i$, or $\mathcal{N} = \bigcup_{i=1}^{R} \mathcal{N}_i$.

Some properties of the Kronecker, and the Khatri-Rao products that will appear throughout the chapters are (see [46] for their derivations):

- $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$
- $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \odot \mathbf{D}) = \mathbf{AC} \odot \mathbf{BD}$
- $(\mathbf{A} \odot \mathbf{B})^H(\mathbf{A} \odot \mathbf{B}) = \mathbf{A}^H\mathbf{A} \circ \mathbf{B}^H\mathbf{B}$
- $(\mathbf{A} \otimes \mathbf{B})^\dagger = \mathbf{A}^\dagger \otimes \mathbf{B}^\dagger$
- $(\mathbf{A} \odot \mathbf{B})^\dagger = (\mathbf{A}^H\mathbf{A} \circ \mathbf{B}^H\mathbf{B})^\dagger(\mathbf{A} \odot \mathbf{B})^H$.

## 5.2 Tensors

A tensor $\boldsymbol{\mathcal{X}} \in \mathbb{C}^{N_1 \times \cdots \times N_R}$ of order $R$ can be viewed as a discretized multidomain signal, with each of its entries indexed over $R$ different domains.

Using multilinear algebra two tensors $\boldsymbol{\mathcal{X}} \in \mathbb{C}^{N_1 \times \cdots \times N_R}$ and $\boldsymbol{\mathcal{G}} \in \mathbb{C}^{K_1 \times \cdots \times K_R}$ may be related by a multilinear system of equations whenever they have a linear relationship between each of its corresponding domains, as depicted in Fig. 5.2a. That is,

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}} \bullet_1 \mathbf{U}_1 \bullet_2 \cdots \bullet_R \mathbf{U}_R, \tag{5.1}$$

where $\{\mathbf{U}_i \in \mathbb{C}^{N_i \times K_i}\}_{i=1}^{R}$ represents a set of matrices that capture the relationship between the $i$th domain of $\boldsymbol{\mathcal{X}}$ and the so-called *core tensor* $\boldsymbol{\mathcal{G}}$, and $\bullet_i$ represents the $i$th mode product between a tensor and a matrix [47].

An alternative representation of (5.1) can be obtained by vectorizing (5.1), yielding

$$\mathbf{x} = (\mathbf{U}_1 \otimes \cdots \otimes \mathbf{U}_R)\,\mathbf{g}, \tag{5.2}$$

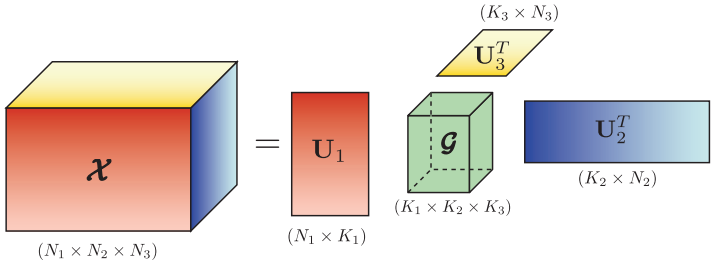
with $\mathbf{x} = \text{vec}(\boldsymbol{\mathcal{X}}) = \boldsymbol{\mathcal{X}}(:) \in \mathbb{C}^{\tilde{N}}$; $\tilde{N} = \prod_{i=1}^{R} N_i$, and $\mathbf{g} = \text{vec}(\boldsymbol{\mathcal{G}}) \in \mathbb{C}^{\tilde{K}}$; $\tilde{K} = \prod_{i=1}^{R} K_i$.

When the core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{C}^{K_c \times \cdots \times K_c}$ is hyperdiagonal (as depicted in Fig. 5.2b), (5.2) simplifies to

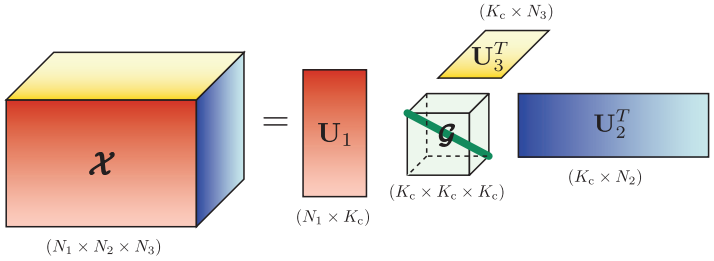
$$\mathbf{x} = (\mathbf{U}_1 \odot \cdots \odot \mathbf{U}_R)\,\mathbf{g} \tag{5.3}$$

with $\mathbf{g}$ collecting the main diagonal entries of $\boldsymbol{\mathcal{G}}$. Note that $\mathbf{g}$ has different meanings in (5.2) and (5.3), which can always be inferred from the context.

Such a multilinear system is commonly seen with $R = 2$ and $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}} \bullet_1 \mathbf{U}_1 \bullet_2 \mathbf{U}_2 = \mathbf{U}_2\boldsymbol{\mathcal{G}}\mathbf{U}_1^T$, for instance, in image processing when relating an image to its 2-dimensional Fourier transform with $\boldsymbol{\mathcal{G}}$ being the spatial Fourier transform of $\boldsymbol{\mathcal{X}}$, and $\mathbf{U}_1$ and $\mathbf{U}_2$ being inverse Fourier matrices related to the row and column spaces of the image, respectively. When dealing with Fourier matrices (more generally, Vandermonde matrices) with $\mathbf{U}_1 = \mathbf{U}_2$ and a diagonal tensor core, $\boldsymbol{\mathcal{X}}$ will be a Toeplitz covariance matrix, for which the sampling sets may be designed using sparse covariance sensing [48, 49].




(a) Dense core




(b) Diagonal core

Figure 5.2: Graphic representation of multilinear system of equations for $R = 3$. Colors represent arbitrary values.

## 5.3 Submodular optimization

Submodularity is a notion based on the law of diminishing returns that is useful to obtain heuristic algorithms with near-optimality guarantees for cardinality-constrained discrete optimization problems. More precisely, submodularity is formally defined as follows.

**Definition 5.1** (Submodular function)**.** *A set function $f : 2^{\mathcal{N}} \to \mathbb{R}$ defined over the subsets of $\mathcal{N}$ is submodular if it satisfies that for every $\mathcal{X} \subseteq \mathcal{N}$, and $x, y \in \mathcal{N} \setminus \mathcal{X}$ we have*

$$f(\mathcal{X} \cup \{x\}) - f(\mathcal{X}) \geq f(\mathcal{X} \cup \{x, y\}) - f(\mathcal{X} \cup \{y\}).$$

*A function $f$ is said to be supermodular if $-f$ is a submodular function.*

Besides submodularity, many near-optimality theorems in discrete optimization require that functions are also monotone non-decreasing, and normalized.

**Definition 5.2** (Monotonicity)**.** *A set function $f : 2^{\mathcal{N}} \to \mathbb{R}$ is monotone non-decreasing if for every $\mathcal{X} \subseteq \mathcal{N}$*

$$f(\mathcal{X} \cup \{x\}) \geq f(\mathcal{X}) \quad \forall x \in \mathcal{N} \setminus \mathcal{X}$$

**Definition 5.3** (Normalization)**.** *A set function $f : 2^{\mathcal{N}} \to \mathbb{R}$ is normalized if $f(\varnothing) = 0$.*

When a set function $f$ is submodular, monotone non-decreasing, and normalized, then one can claim that the solution of a greedy maximization algorithm (summarized in Algorithm 1) has a multiplicative near-optimality guarantee for the cardinality-constrained maximization problem

$$\underset{\mathcal{X} \subseteq \mathcal{N}}{\text{maximize}} \quad f(\mathcal{X}) \quad \text{subject to} \quad |\mathcal{X}| = K. \tag{5.4}$$

**Algorithm 1** Greedy maximization algorithm

**Require:** $\mathcal{X} = \varnothing$, $K$

1: **for** $k \leftarrow 1$ **to** $K$
2: $\quad s^* \leftarrow \arg\max_{s \notin \mathcal{X}} f(\mathcal{X} \cup \{s\})$
3: $\quad \mathcal{X} \leftarrow \mathcal{X} \cup \{s^*\}$
4: **end**
5: **return** $\mathcal{X}$

In particular, Neumhauser [50] proved the following theorem.

**Theorem 5.1** (Near-optimal maximization of submodular function subject to a cardinality constraint [50])**.** *Let $f : 2^{\mathcal{N}} \to \mathbb{R}$ be a monotone non-decreasing, normalized, submodular function, and let $f(\mathcal{X}^*)$ denote the optimal solution of* (5.4)*. Furthermore, let $\mathcal{X}_{greedy}$ be the solution obtained by Algorithm 1. Then*

$$f(\mathcal{X}_{greedy}) \geq \left(1 - \frac{1}{e}\right) f(\mathcal{X}^*),$$

*where $e$ is Euler's number.*

Many discrete optimization problems can be formulated using (5.4) and meet the conditions of Theorem 5.1. These include most greedy methods used in single domain sparse sampling [51–53], and the $D$-optimal formulation of the sparse tensor sensing problem. Nevertheless, as we will see later, many other problems, such as the frame-potential-based sparse tensor sensing problem cannot be formulated as (5.4), since they require imposing extra constraints on top of the restriction on the total cardinality.

In submodular optimization, matroids are generally used to impose constraints on an optimization, since they define a class of problems that we can efficiently solve near-optimally. A matroid is a mathematical structure that generalizes the concept of linear independence in algebra to sets and can be used to define many types of constraints [54], e.g. the ones in (6.10). Formally, a matroid is defined as follows.

**Definition 5.4** (Matroid)**.** *A finite matroid $\mathcal{M}$ is a pair $(\mathcal{N}, \mathcal{I})$, where $\mathcal{N}$ is a finite set (also called the ground set) and $\mathcal{I}$ is a family of subsets of $\mathcal{N}$ (called the independent sets) that satisfies the following properties:*

1. *The empty set is independent, i.e., $\varnothing \in \mathcal{I}$.*
2. *For every $\mathcal{X} \subseteq \mathcal{Y} \subseteq \mathcal{N}$, if $\mathcal{Y} \in \mathcal{I}$, then $\mathcal{X} \in \mathcal{I}$.*
3. *For every $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{N}$ such that $|\mathcal{Y}| > |\mathcal{X}|$ and $\mathcal{X}, \mathcal{Y} \in \mathcal{I}$ there exists one $x \in \mathcal{Y} \setminus \mathcal{X}$ such that $\mathcal{X} \cup \{x\} \in \mathcal{I}$.*

In this thesis, we will deal with the following types of matroids.

**Example 5.1** (Uniform matroid)**.** *The subsets of $\mathcal{N}$ with at most $K$ elements form a uniform matroid $\mathcal{M}_{\mathrm{u}} = (\mathcal{N}, \mathcal{I}_{\mathrm{u}})$ with $\mathcal{I}_{\mathrm{u}} = \{\mathcal{X} \subseteq \mathcal{N} : |\mathcal{X}| \leq K\}$.*

**Algorithm 2** Greedy maximization subject to $T$ matroid constraints

**Require:** $\mathcal{X} = \varnothing$, $K$, $\{\mathcal{I}_i\}_{i=1}^{T}$
1: **for** $k \leftarrow 1$ **to** $K$
2: $\quad s^* = \arg\max_{s \notin \mathcal{X}} \{f(\mathcal{X} \cup \{s\}) : \mathcal{X} \cup \{s\} \in \bigcap_{i=1}^{T} \mathcal{I}_i\}$
3: $\quad \mathcal{X} \leftarrow \mathcal{X} \cup \{s^*\}$
4: **end**
5: **return** $\mathcal{X}$

**Example 5.2** (Partition matroid)**.** *If* $\{\mathcal{N}_i\}_{i=1}^{R}$ *form a disjoint partition of* $\mathcal{N} = \bigcup_{i=1}^{R} \mathcal{N}_i$ *then* $\mathcal{M}_{\mathrm{p}} = (\mathcal{N}, \mathcal{I}_{\mathrm{p}})$ *with* $\mathcal{I}_{\mathrm{p}} = \{\mathcal{X} \subseteq \mathcal{N} : |\mathcal{X} \cap \mathcal{N}_i| \leq K_i \quad i = 1, \ldots, R\}$ *defines a partition matroid.*

**Example 5.3** (Partition-truncated matroid)**.** *The intersection of a uniform matroid* $\mathcal{M}_{\mathrm{u}} = (\mathcal{N}, \mathcal{I}_{\mathrm{u}})$ *and a partition matroid* $\mathcal{M}_{\mathrm{p}} = (\mathcal{N}, \mathcal{I}_{\mathrm{p}})$ *defines a partition-truncated matroid* $\mathcal{M}_{\mathrm{t}} = (\mathcal{N}, \mathcal{I}_{\mathrm{p}} \cap \mathcal{I}_{\mathrm{u}})$.

As happened with the cardinality-constrained submodular optimization problem, the matroid constrained submodular optimization problem

$$\underset{\mathcal{X} \subseteq \mathcal{N}}{\text{maximize}} \quad f(\mathcal{X}) \quad \text{subject to} \quad \mathcal{X} \in \bigcap_{i=1}^{T} \mathcal{I}_i \tag{5.5}$$

can also be solved near-optimally using Algorithm 2. This result is formally stated in the following theorem.

**Theorem 5.2** (Near-optimal maximization of submodular function subject to a matroid constraint [55])**.** *Let* $f : 2^{\mathcal{N}} \to \mathbb{R}$ *be a monotone non-decreasing, normalized, submodular set function, and* $\{\mathcal{M}_i = (\mathcal{N}, \mathcal{I}_i)\}_{i=1}^{T}$ *be a set of matroids defined over* $\mathcal{N}$*. Furthermore, let* $f(\mathcal{X}^*)$ *denote the optimal solution of* (5.5)*, and let* $\mathcal{X}_{greedy}$ *be the solution obtained by Algorithm 2. Then*

$$f(\mathcal{X}_{greedy}) \geq \frac{1}{T+1} f(\mathcal{X}^*).$$

# 6 Problem modeling

We are concerned with the design of optimal sampling strategies for an $R$th order tensor signal $\boldsymbol{\mathcal{X}} \in \mathbb{C}^{N_1 \times \cdots \times N_R}$, which admits a multilinear parameterization in terms of a core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{C}^{K_1 \times \cdots \times K_R}$ (dense or diagonal) of smaller dimensionality. We assume that the set of system matrices $\{\mathbf{U}_i\}_{i=1}^R$ are perfectly known, and that each of them is tall, i.e., $N_i > K_i$ for $i = 1, \ldots, R$, and has full column rank.

Sparse sampling a tensor $\boldsymbol{\mathcal{X}}$ is equivalent to selecting entries of $\mathbf{x} = \text{vec}(\boldsymbol{\mathcal{X}})$. Let $\tilde{\mathcal{N}}$ denote the set of indices of $\mathbf{x}$. Then, a particular sample selection is determined by a subset of selected indices $\mathcal{L}_{\text{un}} \subseteq \tilde{\mathcal{N}}$ such that $|\mathcal{L}_{\text{un}}| = L_{\text{un}}$ (subscript "un" denotes unstructured). This way, we can denote the process of sampling $\boldsymbol{\mathcal{X}}$ as a multiplication of $\mathbf{x}$ by a selection matrix $\boldsymbol{\Theta}(\mathcal{L}_{\text{un}}) \in \{0,1\}^{L_{\text{un}} \times \tilde{N}}$ such that

$$\mathbf{y} = \boldsymbol{\Theta}(\mathcal{L}_{\text{un}})\mathbf{x} = \boldsymbol{\Theta}(\mathcal{L}_{\text{un}})(\mathbf{U}_1 \otimes \cdots \otimes \mathbf{U}_R)\mathbf{g}, \tag{6.1}$$

for a dense core [cf. (5.2)], and

$$\mathbf{y} = \boldsymbol{\Theta}(\mathcal{L}_{\text{un}})\mathbf{x} = \boldsymbol{\Theta}(\mathcal{L}_{\text{un}})(\mathbf{U}_1 \odot \cdots \odot \mathbf{U}_R)\mathbf{g}, \tag{6.2}$$

for a diagonal core [cf. (5.3)]. Here, $\mathbf{y}$ is a vector containing the $L_{\text{un}}$ selected entries of $\mathbf{x}$ indexed by the set $\mathcal{L}_{\text{un}}$.

For each case, if $\boldsymbol{\Theta}(\mathcal{L}_{\text{un}})(\mathbf{U}_1 \otimes \cdots \otimes \mathbf{U}_R)$ and $\boldsymbol{\Theta}(\mathcal{L}_{\text{un}})(\mathbf{U}_1 \odot \cdots \odot \mathbf{U}_R)$ have full column rank, then knowing $\mathbf{y}$ allows to retrieve a unique least squares solution, $\hat{\mathbf{g}}$, as

$$\hat{\mathbf{g}} = \left[\boldsymbol{\Theta}(\mathcal{L}_{\text{un}})(\mathbf{U}_1 \otimes \cdots \otimes \mathbf{U}_R)\right]^{\dagger} \mathbf{y}, \tag{6.3}$$

or

$$\hat{\mathbf{g}} = \left[\boldsymbol{\Theta}(\mathcal{L}_{\text{un}})(\mathbf{U}_1 \odot \cdots \odot \mathbf{U}_R)\right]^{\dagger} \mathbf{y}, \tag{6.4}$$

depending on whether $\boldsymbol{\mathcal{G}}$ is dense or hyperdiagonal. Next, we estimate $\boldsymbol{\mathcal{X}}$ using either (5.2) or (5.3).

In many applications, such as transmitter-receiver placement in multiple input multiple output (MIMO) radar, it is not possible to perform sparse sampling in an unstructured manner by ignoring the underlying domains. For these applications, some unstructured sparse sample selections generally require using a dense sensor selection in each domain (as shown in Fig. 1.1b), which produces a significant increase in hardware cost. Also, there is no particular structure in (6.3) and (6.4) that may be exploited to compute the pseudo-inverses, thus leading to a high computational cost to estimate $\mathbf{x}$. Finally, in the multidomain case, the dimensionality grows rather fast making it difficult to store the matrix $(\mathbf{U}_1 \otimes \cdots \otimes \mathbf{U}_R)$ or $(\mathbf{U}_1 \odot \cdots \odot \mathbf{U}_R)$ to perform row subset selection. For all these reasons, we will constrain ourselves to the case where the sampling matrix has a compatible Kronecker structure. In particular, we define a new sampling matrix

$$\boldsymbol{\Phi}(\mathcal{L}) \coloneqq \boldsymbol{\Phi}_1(\mathcal{L}_1) \otimes \cdots \otimes \boldsymbol{\Phi}_R(\mathcal{L}_R), \tag{6.5}$$

where each $\boldsymbol{\Phi}_i(\mathcal{L}_i)$ represents a selection matrix for the $i$th factor of $\boldsymbol{\mathcal{X}}$, $\mathcal{L}_i \subseteq \mathcal{N}_i$ is the set of selected row indices from the matrix $\mathbf{U}_i$ for $i = 1, \ldots, R$, and $\mathcal{L} = \bigcup_{i=1}^{R} \mathcal{L}_i$ and $\mathcal{L}_i \cap \mathcal{L}_j = \varnothing$ for $i \neq j$.

We will use the notation $|\mathcal{L}_i| = L_i$ and $|\mathcal{L}| = \sum_{i=1}^{R} L_i = L$ to denote the number of selected sensors per domain and the total number of selected sensors, respectively; whereas $\tilde{\mathcal{L}} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_R$ and $\tilde{L} = |\tilde{\mathcal{L}}| = \prod_{i=1}^{R} L_i$ denote the set of sample indices and the total number of samples acquired with the above Kronecker-structured sampler. In order to simplify the notation, whenever it will be clear, we will drop the explicit dependency of $\boldsymbol{\Phi}_i(\mathcal{L}_i)$ on the set of selected rows $\mathcal{L}_i$, from now on, and simply use $\boldsymbol{\Phi}_i$.

Imposing a Kronecker structure on the sampling scheme means that sampling can be performed independently for each domain. In the *dense core tensor* case [cf. (5.2)], we have

$$\begin{aligned} \mathbf{y} &= (\boldsymbol{\Phi}_1 \otimes \cdots \otimes \boldsymbol{\Phi}_R)\,(\mathbf{U}_1 \otimes \cdots \otimes \mathbf{U}_R)\,\mathbf{g} \\ &= (\boldsymbol{\Phi}_1\mathbf{U}_1 \otimes \cdots \otimes \boldsymbol{\Phi}_R\mathbf{U}_R)\,\mathbf{g} = \boldsymbol{\Psi}(\mathcal{L})\mathbf{g}, \end{aligned} \tag{6.6}$$

whereas in the *diagonal core tensor* case [cf. (5.3)], we have

$$\begin{aligned} \mathbf{y} &= (\boldsymbol{\Phi}_1 \otimes \cdots \otimes \boldsymbol{\Phi}_R)\,(\mathbf{U}_1 \odot \cdots \odot \mathbf{U}_R)\,\mathbf{g} \\ &= (\boldsymbol{\Phi}_1\mathbf{U}_1 \odot \cdots \odot \boldsymbol{\Phi}_R\mathbf{U}_R)\,\mathbf{g} = \boldsymbol{\Psi}(\mathcal{L})\mathbf{g}. \end{aligned} \tag{6.7}$$

As in the unstructured case, whenever (6.6) or (6.7) are overdetermined, using least squares, we can estimate the core $\hat{\mathbf{g}} = \boldsymbol{\Psi}^{\dagger}(\mathcal{L})\mathbf{y}$ as

$$\hat{\mathbf{g}} = \left[(\boldsymbol{\Phi}_1\mathbf{U}_1)^{\dagger} \otimes \cdots \otimes (\boldsymbol{\Phi}_R\mathbf{U}_R)^{\dagger}\right]\mathbf{y}, \tag{6.8}$$

or

$$\hat{\mathbf{g}} = \left[(\boldsymbol{\Phi}_1\mathbf{U}_1)^{H}(\boldsymbol{\Phi}_1\mathbf{U}_1) \circ \cdots \circ (\boldsymbol{\Phi}_R\mathbf{U}_R)^{H}(\boldsymbol{\Phi}_R\mathbf{U}_R)\right]^{\dagger}\left[(\boldsymbol{\Phi}_1\mathbf{U}_1)^{H} \odot \cdots \odot (\boldsymbol{\Phi}_R\mathbf{U}_R)^{H}\right]\mathbf{y}, \tag{6.9}$$

and then reconstruct $\hat{\mathbf{x}}$ using (5.2) or (5.3), respectively. Comparing (6.8) and (6.9) to (6.3) and (6.4) we can see that leveraging the Kronecker structure of the proposed sampling scheme allows to greatly reduce the computational complexity of the least-squares problem, as the pseudoinverses in (6.8) and (6.9) are taken on matrices of a much smaller dimensionality than in (6.3) and (6.4). An illustration of the comparison between unstructured sparse sensing and Kronecker-structured sparse sensing is shown in Fig. 6.1 for $R = 2$.

Suppose the measurements collected in $\mathbf{y}$ are perturbed by zero-mean white Gaussian noise with unit variance, then the least-squares solution has the inverse error covariance or the Fisher information matrix $\mathbf{T}(\mathcal{L}) = \mathbb{E}\{(\mathbf{g} - \hat{\mathbf{g}})(\mathbf{g} - \hat{\mathbf{g}})^H\} = \boldsymbol{\Psi}^H(\mathcal{L})\boldsymbol{\Psi}(\mathcal{L})$ that determines the quality of the estimators $\hat{\mathbf{g}}$. Therefore, we can use scalar functions of $\mathbf{T}(\mathcal{L})$ as a figure of merit to propose the sparse tensor sampling problem

$$\underset{\mathcal{L}_1,\ldots,\mathcal{L}_R}{\text{optimize}} \quad f\left\{\mathbf{T}(\mathcal{L})\right\} \text{ subject to } \sum_{i=1}^{R} |\mathcal{L}_i| = L,\ \mathcal{L} = \bigcup_{i=1}^{R} \mathcal{L}_i, \tag{6.10}$$

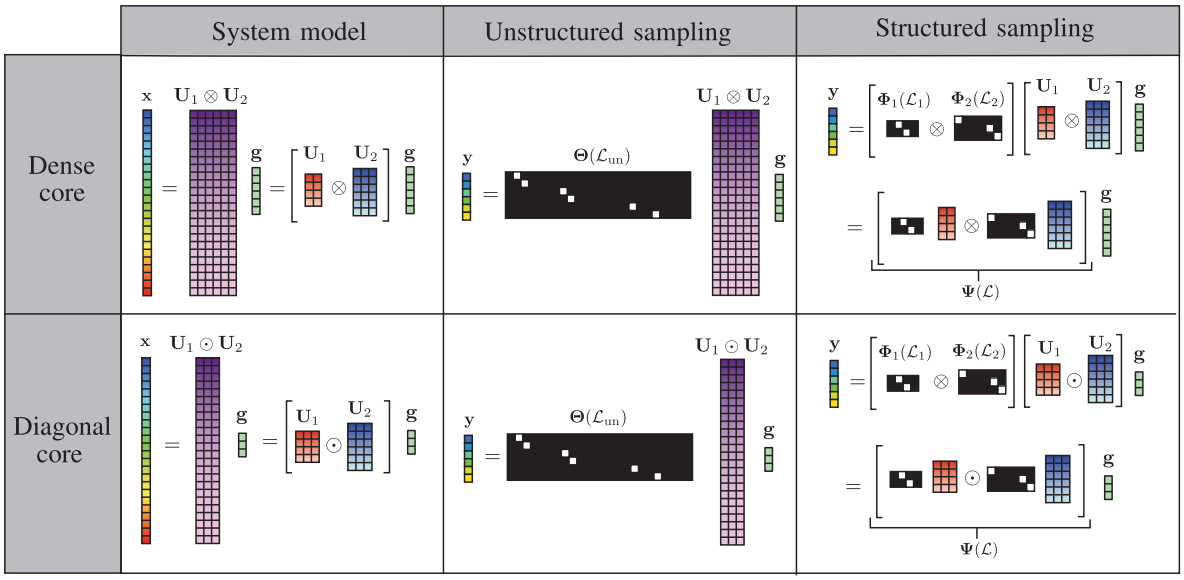



Figure 6.1: Comparison between unstructured sampling and structured sampling ($R = 2$). Black (white) cells represent zero (one) entries, and colored cells represent arbitrary numbers.

where with "optimize" we mean either "maximize" or "minimize" depending on the choice of the scalar function $f\{\cdot\}$. Solving (6.10) is not trivial due to the cardinality constraints. Therefore, in the following, we will propose tight surrogates for typically used scalar performance metrics $f\{\cdot\}$ in design of experiments with which the above discrete optimization problem can be solved efficiently and near optimally.

Note that the cardinality constraint in (6.10) restricts the total number of selected sensors to $L$, without imposing any constraint on the total number of gathered samples $\tilde{L}$. Although the maximum number of samples can be constrained using the constraint $\sum_{i=1}^{R} \log |\mathcal{L}_i| \leq \tilde{L}$, the resulting near-optimal solvers are computationally intense with a complexity of about $\mathcal{O}(N^5)$ [56, 57]. Such heuristics are not suitable for the large-scale scenarios of interest.

## 6.1 Prior art

Choosing the best subset of (spatial or temporal) measurements from a large set of candidate sensing locations has received a lot of attention, particularly for $R = 1$, usually under the name of sensor selection/placement, which also is more generally referred to as sparse sampling [4].

Typically sparse sampling design is posed as a discrete optimization problem that that tries to find the best subset of rows of a system matrix that optimizes an adequate statistical criterion on $\mathbf{T}(\mathcal{L})$. Some of the popular choices for the performance measure $f\{\cdot\}$ are:

- *A-optimality or mean squared error (MSE):* $f\left\{\mathbf{T}(\mathcal{L})\right\} := \operatorname{tr}\left\{\mathbf{T}^{-1}\right\}$ (minimize)
- *E-optimality:* $f\left\{\mathbf{T}(\mathcal{L})\right\} := \lambda_{\min}\{\mathbf{T}\}$ (maximize)

- *D-optimality:* $f\left\{\mathbf{T}(\mathcal{L})\right\} := \log\det\left\{\mathbf{T}\right\}$ (maximize)
- *Frame potential:* $f\left\{\mathbf{T}(\mathcal{L})\right\} := \mathrm{tr}\left\{\mathbf{T}^H\mathbf{T}\right\}$ (minimize).

In this work, we will focus on the D-optimality and frame potential criteria as we will show later that these performance metrics lead to very efficient sampler designs.

Depending on the strategy used to solve the optimization problem (6.10) we can classify the prior art in two categories: solvers based on convex optimization, and greedy methods that leverage submodularity.

In the former category, Joshi et al. [58] and Chepuri et al. [59] propose several convex relaxations of the sparse sampling problem for different optimality criteria for inverse problems with linear and non-linear models, respectively. In particular, due to the Boolean nature of the sensor selection problem (i.e., a sensor is either selected or not), its related optimization problem is not convex. Hence, the Boolean constraints are relaxed with box constraints. This way, once the relaxed convex problem is solved a thresholding heuristic (determinstc or randomized) is used to recover a Boolean solution. Despite its good performance, the complexity of convex optimization solvers is rather high (cubic with the dimensionality of the signal). Therefore, the use of convex optimization approaches to solve the sparse sampling problem in large-scale scenarios, such as the sparse tensor sampling problem of interest, gets even more computationally intense.

For high-dimensional scenarios, greedy methods (algorithms that select one sensor at a time) are more useful. A greedy algorithm scales linearly with the number of sensors, and if one can prove submodularity of the objective function, its solutions has a multiplicative near-optimal guarantee [50]. Several authors have followed this strategy and have proved submodularity of different optimality criteria such as $D$-optimality [51], mutual information [52], or frame potential [53]. All of them for the case $R = 1$.

Besides parameter estimation, sparse sampling has also been studied for other common signal processing tasks, like detection [60, 61] or filtering [62, 63]. Nevertheless, to our knowledge, the sparse sampling framework has never been extended for multidomain signals, and this is the focus of this work.

In a different context, the extension of Compressed Sensing (CS) to multidomain signals has been extensively studied [64–67]. CS is many times seen as a complementary sampling framework to sparse sampling [4], where in CS the focus is on recovering a sparse signal rather than designing a sparse measurement space. Furthermore, most of the work in CS deals with the design of reconstruction algorithms rather than on the sampling scheme (sampling is mostly dense and random in CS).

## 6.2 Our contributions

In this thesis, we extend the sparse sampling framework to *multilinear inverse problems*. We refer to it as "sparse tensor sampling". We also propose efficient algorithms to solve the sparse tensor sampling with theoretical guarantees. We focus on two particular cases, depending on the structure of the core tensor $\boldsymbol{\mathcal{G}}$:

- *Dense core*: Whenever the core tensor is non-diagonal, sampling can be performed based on (6.6). We will see that to ensure identifiability of the system, we need to select more entries in each domain than the rank of its corresponding system matrix, i.e., as a necessary condition we require $L \geq \sum_{i=1}^{R} K_i = K$ sensors. We provide two greedy algorithms based on the $D$-optimality and frame potential criteria to compute a near-optimal subsampling set.

- *Diagonal core*: Whenever the core tensor is diagonal, sampling can be performed based on (6.7). The use of the Khatri-Rao product allows for higher compression. In particular, under mild conditions on the entries of the factor matrices we can guarantee identifiability of the sampled equations using $L \geq K_{\mathrm{c}} + R - 1$ sensors. We propose a greedy method, again based on the frame potential, to compute a near-optimal sampling set.

# 7 Dense core sampling

In this chapter, we focus on the most general situation when $\boldsymbol{\mathcal{G}}$ is an unstructured dense tensor. Our objective is to design the sampling sets $\{\mathcal{L}_i\}_{i=1}^R$ by solving the discrete optimization problem (6.10).

To begin with, we will focus on the $D$-optimality criterion for the sparse tensor sampling problem. We will show that the Kronecker structure of the sampling matrix can be exploited to alleviate the curse of dimensionality, as it decomposes the objective function into a sum of small single domain terms. Furthermore, we will propose an efficient and near-optimal greedy algorithm to solve this problem.

After this, we work on the formulation of the sparse tensor sampling problem using the frame potential as a performance measure. Following the same rationale as in [53], but for multidomain signals, we will argue that the frame potential is a tight surrogate of the MSE. By doing so, we will see that when we impose a Kronecker structure on the sampling scheme, as in (6.5), the frame potential of $\boldsymbol{\Psi}$ can be factorized in terms of the frame potential of the different domain factors. This allows us to propose a low complexity algorithm for sampling tensor data.

Both the methods have their benefits and disadvantages, and depending on the situation, one might be more useful than the other. For this reason, we will finish each subsection by mentioning some practical considerations that need to be addressed when deciding on which method to use for a particular situation.

## 7.1 D-optimal greedy method

In $D$-optimal experiment design one is concerned with minimizing the volume of the confidence ellipsoid of the parameters to be estimated, which can be shown to be equivalent to maximizing $\log\det\{\mathbf{T}\}$. In the Kronecker-structured case, this function can be decomposed as a sum of contributions in each domain.

Recall that $\mathcal{N}_i$ denotes the set of row indices of $\mathbf{U}_i$ and $\mathcal{L}_i \subseteq \mathcal{N}_i$ denotes the set of indices of the selected rows of $\mathbf{U}_i$ for $i = 1, \dots, R$. Furthermore, recall the definitions of $\mathcal{N} = \bigcup_{i=1}^R \mathcal{N}_i$ and $\mathcal{L} = \bigcup_{i=1}^R \mathcal{L}_i$ with $|\mathcal{N}| = N$, $|\mathcal{L}| = L$, $|\mathcal{N}_i| = N_i$, and $|\mathcal{L}_i| = L_i$ for $i = 1, \dots, R$. Using these, let us define

$$\boldsymbol{\Psi}_i(\mathcal{L}_i) \coloneqq \boldsymbol{\Phi}_i(\mathcal{L}_i)\mathbf{U}_i, \tag{7.1a}$$

$$\boldsymbol{\Psi}(\mathcal{L}) \coloneqq \boldsymbol{\Psi}_1(\mathcal{L}_1) \otimes \cdots \otimes \boldsymbol{\Psi}_R(\mathcal{L}_R), \tag{7.1b}$$

and the respective Grammian matrices

$$\mathbf{T}_i(\mathcal{L}_i) \coloneqq \boldsymbol{\Psi}_i^H(\mathcal{L}_i)\boldsymbol{\Psi}_i(\mathcal{L}_i), \tag{7.2a}$$

$$\mathbf{T}(\mathcal{L}) \coloneqq \boldsymbol{\Psi}^H(\mathcal{L})\boldsymbol{\Psi}(\mathcal{L}) = \mathbf{T}_i(\mathcal{L}_i) \otimes \cdots \otimes \mathbf{T}_R(\mathcal{L}_R). \tag{7.2b}$$

Leveraging (7.2b), the $D$-optimal criterion becomes

$$\log\det\{\mathbf{T}(\mathcal{L})\} = \log\det\{\mathbf{T}_1 \otimes \cdots \otimes \mathbf{T}_R\}.$$

Furthermore, since for $\mathbf{A} \in \mathbb{C}^{K_\mathrm{A} \times K_\mathrm{A}}$, $\mathbf{B} \in \mathbb{C}^{K_\mathrm{B} \times K_\mathrm{B}}$ and $\mathbf{C} \in \mathbb{C}^{K_\mathrm{C} \times K_\mathrm{C}}$

$$\det\{\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}\} = \det\{\mathbf{A}\}^{K_\mathrm{B}K_\mathrm{C}} \det\{\mathbf{B}\}^{K_\mathrm{A}K_\mathrm{C}} \det\{\mathbf{C}\}^{K_\mathrm{A}K_\mathrm{B}},$$

the determinant of $\mathbf{T}$ can be expanded as

$$\det\{\mathbf{T}_1 \otimes \cdots \otimes \mathbf{T}_R\} = \prod_{i=1}^{R} \det\{\mathbf{T}_i\}^{\tilde{K}_{-i}},$$

where $\tilde{K}_{-i} = \prod_{j \neq i} K_j$ is used to simplify the notation.

Taking the logarithm of this expression we arrive at

$$\log\det\{\mathbf{T}_1 \otimes \cdots \otimes \mathbf{T}_R\} = \sum_{i=1}^{R} \tilde{K}_{-i} \log\det\{\mathbf{T}_i\}, \tag{7.3}$$

where we see that using a Kronecker-structured sampler allows to express the volume of the multidomain confidence ellipsoid as a weighted sum of the volumes of the single domain ellipsoids.

### 7.1.1 Submodularity of $\log\det\{\mathbf{T}\}$

Shamaiah et al. [51] showed that in the single domain case $\log\det\{\mathbf{T}(\mathcal{L}) + \epsilon\mathbf{I}\}$, with small $\epsilon$, is a normalized, monotone, submodular function. Here, we extend this result to the tensor case. To do so, we modify (7.3) to

$$LD(\mathcal{L}) = \sum_{i=1}^{R} \tilde{K}_{-i} \log\det\{\mathbf{T}_i(\mathcal{L}_i) + \epsilon\mathbf{I}\} - R\tilde{K}\log\epsilon. \tag{7.4}$$

For very small values of $\epsilon > 0$ (7.3) and (7.4) are almost equivalent. Thus maximizing (7.4) is approximately the same as maximizing (7.3). However, $LD(\mathcal{L})$ satisfies the conditions of the near-optimality theorems.

**Theorem 7.1.** *The set function $LD(\mathcal{L})$ with $\epsilon > 0$ defined in (7.4) is a normalized, monotone non-decreasing, submodular function for all subsets of $\mathcal{N} = \bigcup_{i=1}^{R} \mathcal{N}_i$.*

*Proof.* See Appendix A.1. ■

In virtue of Theorem 7.1, we define the following submodular optimization problem to find the $D$-optimal sensor placement

$$\begin{aligned} \underset{\mathcal{L} \subseteq \mathcal{N}}{\text{maximize}} \quad & LD(\mathcal{L}) \\ \text{subject to} \quad & |\mathcal{L}| = L, \quad \mathcal{L} = \bigcup_{i=1}^{R} \mathcal{L}_i. \end{aligned} \tag{7.5}$$

Problem (7.5) is a particularization of (6.10), and is in the form of (5.4). Hence, it satisfies the conditions of Theorem 5.1, and we can guarantee the near-optimality of the greedy solution.

**Corollary 7.1.** *The solution $\mathcal{L}_{greedy}$ to (7.5) obtained using the greedy Algorithm 1 is $(1-1/e)$-near-optimal, i.e.*

$$LD(\mathcal{L}_{greedy}) \geq \left(1 - \frac{1}{e}\right) LD(\mathcal{L}^*).$$

### 7.1.2 Computational complexity

A naïve application of Algorithm 1 to solve (7.5) yields a high computational complexity due to the unnecesary function evaluations that can be avoided by leveraging the structure in (7.3). In particular, one can improve the complexity of evaluating $LD(\mathcal{L} \cup \{s\})$ by keeping in memory the value of every summand in (7.4) in the last iteration and evaluating only the term where $s$ is added. Thus, in very iteration, Algorithm 1 performs $R$ single-domain maximization steps. Hence, the complexity of the iterations is governed by the domain with the highest dimensionality. Further, the time complexity of the single-domain maximization can be optimized using rank-one updates with complexity $\mathcal{O}(K_i^2 N_i L)$ as shown in [51]. Hence, using Algorithm 1 to solve (7.5) has a computational complexity $\mathcal{O}(K_{\max}^2 N_{\max} L)$, with $K_{\max} = \max_i K_i$ and $N_{\max} = \max_i N_i$.

The particularization of Algorithm 1 to solve (7.5) is given in Algorithm 3.

**Algorithm 3** Greedy maximization of $\log\det\{\mathbf{T}(\mathcal{L})\}$

**Require:** $\{\mathbf{U}_i\}_{i=1}^R$, $L$, $\epsilon > 0$, and $\mathcal{L} = \varnothing$.
1: **for** $k \leftarrow 1$ **to** $L$
2: **for** $i \leftarrow 1$ **to** $R$
3: $s_i^* \leftarrow \arg\max_{s_i \notin \mathcal{L}_i} \log\det\{\mathbf{T}_i(\mathcal{L}_i \cup \{s_i\}) + \epsilon\mathbf{I}\}$
4: **end**
5: $i^* \leftarrow \arg\max_{1 \leq i \leq R} LD(\mathcal{L} \cup \{s_i^*\})$
6: $\mathcal{L} \leftarrow \mathcal{L} \cup \{s_{i^*}^*\}$
7: **end**
8: **return** $\mathcal{L}$

### 7.1.3 Practical considerations

The greedy method has some peculiarities that need to be considered when used in practice. In particular, as the greedy algorithm adds elements from the ground set, the first iterations of the algorithm do not ensure identifiability of the solution. In fact, until the solution reaches the point where $L_i \geq K_i$ for $i = 1, \ldots, R$, all previous sensor selections render a singular system of equations. Indeed, the rank of $\mathbf{\Psi}$ follows $\operatorname{rank}(\mathbf{\Psi}) = \prod_{i=1}^R \operatorname{rank}(\mathbf{\Psi}_i)$. Therefore, to ensure $\operatorname{rank}(\mathbf{\Psi}) = \tilde{K}$, we require $\operatorname{rank}(\mathbf{\Psi}_i) = K_i$ for $i = 1, \ldots, R$.

Nevertheless, the necessary identifiability condition $L_i \geq K_i$ for $i = 1, \ldots, R$ is not a proper matroid constraint. Any matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$ requires that $\varnothing \in \mathcal{I}$, and the independent set $\mathcal{I} = \{\mathcal{L} \subseteq \mathcal{N} : |\mathcal{L}_i| \geq K_i\}$ does not contain $\varnothing$. This means that these constraints cannot be placed in the optimization if one is interested in having near-optimality guarantees.

The lack of identifiability constraints means that for some models the solution obtained by running the greedy method cannot be used. This situation, for instance, occurs for tensors with an unbalanced dimensionality, i.e. wide spread in the elements of $\{N_i\}_{i=1}^R$ and $\{K_i\}_{i=1}^R$. For those signals, the greedy method tends to produce singular solutions. This is due to the tendency of this algorithm to select elements from each domain in a round-robin fashion, i.e. domains are chosen in a circular manner. Every iteration an element is added to the domain with the smallest contribution (as this maximizes the sum of logarithms), and that gravitates the algorithm towards rotating domains in every step. In this sense, when the elements of $\{K_i\}_{i=1}^R$ are very spread it can happen that the algorithm reaches the maximum number of iterations without guaranteeing identifiability in the domains with the largest $K_i$. Besides, this tendency to alternate domains in the selection of elements yields in general solutions that require a higher number of measurements $\tilde{L}$ than other algorithms, as we will also demonstrate in the simulations.

In practice, however, whenever we are dealing with tensors with a well-ballanced dimensionality, greddy maximization of $LD(\mathcal{L})$ usually gives the best performance for a given compression. Not only regarding $D$-optimality, but also regarding MSE.

## 7.2 Frame potential

An alternative to the $D$-optimality criterion is the frame potential performance measure. Recall that the frame potential [68] of the matrix $\mathbf{\Psi}$ is defined as the trace of the Grammian matrix

$$\mathrm{FP}\left(\mathbf{\Psi}\right) \coloneqq \mathrm{tr}\left\{\mathbf{T}^H\mathbf{T}\right\},$$

with $\mathbf{T} = \mathbf{\Psi}^H\mathbf{\Psi}$.

In [53], they showed that the frame potential can be related to the theoretical mean squared error (MSE) of the least-squares estimator in presence of additive white Gaussian noise [68], expressed as

$$\mathrm{MSE}(\mathbf{\Psi}(\mathcal{L})) = \mathrm{tr}\left\{\mathbf{T}^{-1}(\mathcal{L})\right\}, \tag{7.6}$$

using the bound

$$c_1\frac{\mathrm{FP}\left(\mathbf{\Psi}(\mathcal{L})\right)}{\lambda^2_{\max}\{\mathbf{T}(\mathcal{L})\}} \leq \mathrm{MSE}(\mathbf{\Psi}(\mathcal{L})) \leq c_2\frac{\mathrm{FP}\left(\mathbf{\Psi}(\mathcal{L})\right)}{\lambda^2_{\min}\{\mathbf{T}(\mathcal{L})\}}, \tag{7.7}$$

where $c_1$, and $c_2$ are constants that depend on the characteristics of the system equations.

From the bound above, it is clear that by minimizing the frame potential of $\mathbf{\Psi}$ one can minimize the MSE, which is otherwise difficult to minimize as it is neither convex, nor submodular.

The frame potential of the matrix in (7.1b) can be expressed as the frame potential of its factors using (7.2b)

$$\begin{aligned}\mathrm{FP}\left(\mathbf{\Psi}(\mathcal{L})\right) &= \mathrm{tr}\left\{\mathbf{T}^H(\mathcal{L})\mathbf{T}(\mathcal{L})\right\} = \mathrm{tr}\left\{(\mathbf{T}_i \otimes \cdots \otimes \mathbf{T}_R)^H(\mathbf{T}_i \otimes \cdots \otimes \mathbf{T}_R)\right\} \\ &= \mathrm{tr}\left\{\mathbf{T}_1^H\mathbf{T}_1 \otimes \cdots \otimes \mathbf{T}_R^H\mathbf{T}_R\right\}. \end{aligned} \tag{7.8}$$

Now, using the fact that for any two matrices $\mathbf{A} \in \mathbb{C}^{K_A \times K_A}$ and $\mathbf{B} \in \mathbb{C}^{K_B \times K_B}$ we have $\operatorname{tr}\{\mathbf{A} \otimes \mathbf{B}\} = \operatorname{tr}\{\mathbf{A}\} \operatorname{tr}\{\mathbf{B}\}$, we can expand (7.8) as

$$\operatorname{FP}(\boldsymbol{\Psi}(\mathcal{L})) = \prod_{i=1}^{R} \operatorname{tr}\left\{\mathbf{T}_i^H \mathbf{T}_i\right\} = \prod_{i=1}^{R} \operatorname{FP}(\boldsymbol{\Psi}_i(\mathcal{L}_i)).$$

For brevity, we will write the above expression alternatively as an explicit function of the selection sets $\mathcal{L}_i$:

$$F(\mathcal{L}) \coloneqq \operatorname{FP}(\boldsymbol{\Psi}(\mathcal{L})) = \prod_{i=1}^{R} F_i(\mathcal{L}_i) \coloneqq \prod_{i=1}^{R} \operatorname{FP}(\boldsymbol{\Psi}_i(\mathcal{L}_i)) \tag{7.9}$$

Expression (7.9) shows again the advantage of working with a Kronecker-structured sampler: instead of computing every cross-product between the columns of $\boldsymbol{\Psi}$ to compute the frame potential, we can arrive to the same value using the frame potential of $\{\boldsymbol{\Psi}_i\}_{i=1}^{R}$.

### 7.2.1 Submodularity of $F(\mathcal{L})$

Function $F(\mathcal{L})$ as defined in (7.9) does not directly meet the conditions required for near-optimality of the greedy methods [cf. Theorem 5.1 and Theorem 5.2], but it can be modified slightly to satisfy them. In this sense, we define the function $G : 2^{\mathcal{N}} \to \mathbb{R}$ on the subsets of $\mathcal{N}$ as

$$G(\mathcal{S}) \coloneqq F(\mathcal{N}) - F(\mathcal{N} \setminus \mathcal{S}) = \prod_{i=1}^{R} F_i(\mathcal{N}_i) - \prod_{i=1}^{R} F_i(\mathcal{N}_i \setminus \mathcal{S}_i), \tag{7.10}$$

where

$$\mathcal{S} = \bigcup_{i=1}^{R} \mathcal{S}_i, \quad \mathcal{S}_i \cap \mathcal{S}_j = \varnothing \text{ for } i \neq j$$

and therefore $\{\mathcal{S}_i\}_{i=1}^{R}$ form a partition of $\mathcal{S}$.

It is clear that if we make the change of variables from $\mathcal{L}$ to $\mathcal{S}$ maximizing $G$ over $\mathcal{S}$ is the same as minimizing the frame potential over $\mathcal{L}$. However, working with the complement set results in a set function that is submodular and monotone non-decreasing, as shown in the next theorem. Consequently, $G$ satisfies the conditions of the near-optimality theorems.

**Theorem 7.2.** *The set function $G(\mathcal{S})$ defined in (7.10) is a normalized, monotone non-decreasing, submodular function for all subsets of $\mathcal{N} = \bigcup_{i=1}^{R} \mathcal{N}_i$.*

*Proof.* See Appendix A.2. ■

With this result we can now claim near-optimality of the greedy algorithm that solves the cardinality constrained maximization of $G(\mathcal{S})$. However, as we said, minimizing the frame potential only makes sense as long as (7.7) is tight. In particular,

whenever $\mathbf{T}(\mathcal{L})$ is singular we know that the MSE is infinity, and hence (7.7) is meaningless. For this reason, next to the cardinality constraint in (6.10) that limits the total number of sensors, we need to ensure that $\mathbf{\Psi}(\mathcal{L})$ has full column rank, i.e., $L_i \geq K_i$ for $i = 1, \ldots, R$. In terms of $\mathcal{S}$, this is equivalent to

$$|\mathcal{S}_i| = |\mathcal{N}_i \setminus \mathcal{L}_i| \leq N_i - K_i \quad i = 1, \ldots, R, \tag{7.11}$$

where this set of constraints forms a partition matroid $\mathcal{M}_p = (\mathcal{N}, \mathcal{I}_p)$ (cf. Example 5.2 from Definition 5.4). Hence, we can introduce the following submodular optimization problem as surrogate for the minimization of the frame potential

$$\begin{aligned} \underset{\mathcal{S}\subseteq\mathcal{N}}{\text{maximize}} \quad & G(\mathcal{S}) \\ \text{subject to} \quad & \mathcal{S} \in \mathcal{I} \qquad \mathcal{I} = \mathcal{I}_{\text{u}} \cap \mathcal{I}_p \\ & \mathcal{I}_{\text{u}} = \{\mathcal{A} \subseteq \mathcal{N} : |\mathcal{A}| \leq N - L\} \\ & \mathcal{I}_p = \{\mathcal{A} \subseteq \mathcal{N} : |\mathcal{A} \cap \mathcal{N}_i| \leq N_i - K_i \;\; i = 1, \ldots, R\}. \end{aligned} \tag{7.12}$$

Theorem 5.2 gives, therefore, all the ingredients to assess the near-optimality of Algorithm 2 applied on (7.12), for which the results are particularized as the following corollary.

**Corollary 7.2.** *The solution $\mathcal{S}_{greedy}$ to* (7.12) *obtained using the greedy Algorithm 2) is* 1/2*-near-optimal. That is,*

$$G(\mathcal{S}_{greedy}) \geq \frac{1}{2} G(\mathcal{S}^*).$$

*Proof.* From Theorem 5.2, we know that greedy maximization of a normalized, monotone non-decreasing, submodular function subject to $T$ matroid constraints has a $1/(T+1)$-near-optimality guarantee. In this case, (7.12) has only one matroid constraint (in the form of a truncated matroid), i.e., $T = 1$. ■

So far, we have seen that we can define a submodular surrogate for the minimization of the frame potential, and that this has a 1/2-guarantee. However, we still need to compute an explicit bound with respect to the frame potential of $\mathbf{\Psi}$ which is the objective function we really want to minimize. This bound is given in the following theorem.

**Theorem 7.3.** *The solution $\mathcal{L}_{greedy}$ to* (7.12) *obtained using the greedy Algorithm 2 is near-optimal with respect to the frame potential as*

$$F(\mathcal{L}_{greedy}) \leq \gamma F(\mathcal{L}^*)$$

*with* $\gamma = \frac{1}{2}\left(\frac{K}{L_{min}^2}\prod_{i=1}^{R} F_i(\mathcal{N}_i) + 1\right)$*, and* $L_{min} = \min_{i\in\mathcal{L}} \|\mathbf{u}_i\|_2^2$*, being* $\mathbf{u}_i$ *the ith row of* $(\mathbf{U}_1 \otimes \cdots \otimes \mathbf{U}_R)$.

*Proof.* See Appendix A.3. ■

As happens with the near-optimal guarantee for the single-domain greedy algorithm [53], $\gamma$ is heavily influenced by the frame potential of the unsampled system matrix: The lower $F(\mathcal{N})$ is or the smaller the core is, the tighter the approximation.

### 7.2.2 Computational complexity

The running time of Algorithm 2 applied to solve (7.12) can greatly be reduced if one precomputes the inner products between the rows of every $\mathbf{U}_i$ before starting the iterations. This has a complexity of $\mathcal{O}(N_i^2 K_i)$ for each domain. Once these inner products are computed, in each iteration we just need to find $R$ times the maximum over $\mathcal{O}(N_i)$ elements. Because we run $N - L$ iterations, the complexity of all iterations is $\mathcal{O}(N_{\max}^2)$, with $N_{\max} = \max_i N_i$. Therefore the total computational complexity of the greedy method is $\mathcal{O}(N_{\max}^2 K_{\max})$, being $K_{\max} = \max_i K_i$.

The particularization of Algorithm 2 to solve (7.12) is given in Algorithm 4.

**Algorithm 4** Greedy maximization of $G$

**Require:** $\{\mathbf{U}_i\}_{i=1}^R$, $L$, $\mathcal{S} = \varnothing$ and $\mathcal{D} = \{1, \ldots, R\}$.
**for** $k \leftarrow 1$ **to** $N - L$
  **for** $i \in \mathcal{D}$
    $s_i^* \leftarrow \arg\min_{s_i \notin \mathcal{S}_i} F_i(\mathcal{N}_i \setminus \mathcal{S}_i \cup \{s_i\})$
  **end**
  $i^* \leftarrow \arg\min_{i \in \mathcal{D}} \left\{ F_i(\mathcal{N}_i \setminus \mathcal{S}_i \cup \{s_i^*\}) \prod_{j \neq i} F_j(\mathcal{N}_j \setminus \mathcal{S}_j) \right\}$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_{i^*}^*\}$
  **if** $|\mathcal{S} \cap \mathcal{N}_{i^*}| = N_{i^*} - K_{i^*}$ **then**
    $\mathcal{D} \leftarrow \mathcal{D} \setminus i^*$
  **end**
**end**
**return** $\mathcal{L} = \mathcal{N} \setminus \mathcal{S}$

### 7.2.3 Practical considerations

Despite the near-optimality guarantees of the greedy method, there is one point in which this algorithm needs improvement. Namely, due to the characteristics of the greedy iterations, the algorithm tends to give solutions with a very unbalanced cardinality (as opposed to the method based on $D$-optimality that tends to balance the number of sensors selected in each domain). In particular, for most situations, the algorithm chooses one of the domains in the first few iterations and empties that set till it hits the identifiability constraint of that domain. Then, it proceeds to another domain and empties it as well. These steps are followed until the algorithm reaches the specified number of iterations.

The reason for this behavior is due to the definition of the objective function as a multiplication of smaller objectives. Indeed, if we are asked to minimize a multiplication of two elements by substracting a value from them, it is generally better to substract from the smallest element. Hence, if this minimization is performed multiple times we will tend to remove always from the same element.

The consequences of this behavior are twofold. On the one hand, this greedy method tends to give a sensor placement that yields a very small number of samples $\tilde{L}$, as we will also see in the simulations. Therefore, when comparing this method to other sensor selection schemes that produce solutions with a larger $\tilde{L}$ it generally ranks worse in

MSE for a given $L$. On the other hand, the solution of this scheme tends to be tight on the identifiability constraints for most of the domains, thus hampering the performance on those domains. This implication, however, has a simple solution. By introducing a small slack variable $\alpha_i > 0$ to the constraints, we can obtain a sensor selection which is not tight on the constraints. This amounts to solving the problem

$$\begin{aligned} \underset{\mathcal{S}\subseteq\mathcal{N}}{\text{maximize}} \quad & G(\mathcal{S}) \\ \text{subject to} \quad & |\mathcal{S}| = N - L \\ & |\mathcal{S} \cap \mathcal{N}_i| \leq N_i - K_i - \alpha_i \qquad i = 1, \ldots, R. \end{aligned} \tag{7.13}$$

Tuning $\{\alpha_i\}_{i=1}^R$ allows to regularize the tradeoff between compression and accuracy of the greedy solution.

## 7.3 Convex alternative

As mentioned before, the unidimensional sampling problem has also been addressed from a convex optimization point of view. In the unidimensional case, the Gram matrix can also be expressed as

$$\mathbf{T}(\mathcal{L}) = (\mathbf{\Phi}(\mathcal{L})\mathbf{U})^H(\mathbf{\Phi}(\mathcal{L})\mathbf{U}) = \mathbf{U}^H\mathbf{\Phi}(\mathcal{L})^H\mathbf{\Phi}(\mathcal{L})\mathbf{U} = \mathbf{U}^H\text{diag}\{\mathbf{w}\}\mathbf{U}, \tag{7.14}$$

where $\mathbf{w} \in \{0,1\}^N$ is a selection vector such that

$$w_i = \begin{cases} 1 & i \in \mathcal{L} \\ 0 & i \notin \mathcal{L} \end{cases} \qquad \text{for } i = 1, \ldots, N. \tag{7.15}$$

Using this alternative notation, we can define a suboptimal sampling problem as

$$\begin{aligned} \underset{\mathbf{w}}{\text{maximize}} \quad & \log\det\left\{\mathbf{U}^H\text{diag}(\mathbf{w})\mathbf{U}\right\} \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{w} = L \\ & w_i \in \{0,1\}. \end{aligned} \tag{7.16}$$

This problem is non-convex due to the last Boolean constraint. However, one can relax this problem to obtain the following convex proxy

$$\begin{aligned} \underset{\mathbf{w}}{\text{maximize}} \quad & \log\det\{\mathbf{U}^H\text{diag}(\mathbf{w})\mathbf{U}\} \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{w} = L \\ & \mathbf{0} \preceq \mathbf{w} \preceq \mathbf{1}. \end{aligned} \tag{7.17}$$

To generalize (7.16) to the multidomain case, we can use the fact that the $\mathbf{T}$ can be factorized as a Kronecker product and define a suboptimal tensor sampling problem in

terms of a convex optimization problem

$$\begin{aligned} \underset{\{\mathbf{w}_i\}_{i=1}^R}{\text{maximize}} \quad & \sum_{i=1}^{R} \tilde{K}_{-i} \log\det\{\mathbf{U}_i^H \operatorname{diag}\{\mathbf{w}_i\}\mathbf{U}_i\} \\ \text{subject to} \quad & \sum_{i=1}^{R} \mathbf{1}^T \mathbf{w}_i = L \\ & \mathbf{0} \preceq \mathbf{w}_i \preceq \mathbf{1} \quad i = 1, \ldots, R. \end{aligned} \tag{7.18}$$

As proposed in [58] and [59], we can always recover a discrete solution by thresholding the result of (7.18).

Being a natural extension of the method in [58] it is to be expected that using (7.18) would yield well performing samplers. However, as happened with its unidimensional problem, the complexity of the convex optimization is too high to be used in large-scale problems.

# 8 Diagonal core sampling

So far, we have focused on the case when $\boldsymbol{\mathcal{G}}$ has no particular structure, and is, in principle, a dense tensor. In that case, we have seen that we require at least $\sum_{i=1}^{R} K_i$ sensors to recover our signal with a finite MSE. Nevertheless, in many cases of interest, we can assume that $\boldsymbol{\mathcal{G}}$ has a structure. In particular, in this chapter, we investigate the case when $\boldsymbol{\mathcal{G}}$ is a diagonal tensor. We will show that when this happens, and assuming some mild conditions on the values of $\{\mathbf{U}\}_{i=1}^{R}$, we can leverage the structure of $\boldsymbol{\mathcal{G}}$ to further increase the compression. We will also propose an efficient and near-optimal greedy algorithm based on minimizing the frame potential to design the sampling set.

**Remark.** *Note that for the diagonal core we do not propose a greedy method to solve the* D*-optimal sparse tensor sampling problem. The reason for this is that the Kathri-Rao product in* (6.7) *yields a Grammian matrix whose* log det *cannot be transformed into a submodular function as done in Section 7.1. Indeed, we numerically tested Definition 5.1 on such function, and found out that in many ocassions the submodular inequality is violated*

## 8.1 Identifiability conditions

In contrast to the dense core case, the number of unknowns in a multilinear system of equations with a diagonal core does not increase with the tensor order, whereas for a dense core it grows exponentially. This means that when sampling signals with a diagonal core decomposition, one can allow for a stronger compression. In this sense, Bro and Sidiropoulos [69] showed that the rank of a Khatri-Rao product is always larger than the rank of its factors. More formally, they proved:

**Theorem 8.1** (Rank of Khatri-Rao product [69])**.** *Let* $\mathbf{A} \in \mathbb{C}^{N\times K}$ *and* $\mathbf{B} \in \mathbb{C}^{M\times K}$ *be two matrices with no all-zero column. Then,*

$$\operatorname{rank}(\mathbf{A} \odot \mathbf{B}) \geq \max\{\operatorname{rank}(\mathbf{A}), \operatorname{rank}(\mathbf{B})\}.$$

In virtue of this theorem, we can give the following sufficient conditions for identifiability of the solution to (6.10) when $\boldsymbol{\mathcal{G}}$ is diagonal.

**Corollary 8.1.** *Let* $\{z_i\}_{i=1}^{R}$ *denote the maximum number of zero entries in any column of* $\{\mathbf{U}_i\}_{i=1}^{R}$*. If for every* $\boldsymbol{\Psi}_i(\mathcal{L}_i)$ *we have* $|\mathcal{L}_i| > z_i$*, and there is at least one* $\boldsymbol{\Psi}_j$ *with* $\operatorname{rank}(\boldsymbol{\Psi}_j) = K_c$ *then* $\boldsymbol{\Psi}(\mathcal{L})$ *has full column rank.*

*Proof.* Selecting $L_i > z_i$ rows from each $\mathbf{U}_i$ ensures that no $\boldsymbol{\Psi}_i$ will have an all-zero column. Then, if for at least one $\boldsymbol{\Psi}_j$ we have $\operatorname{rank}(\boldsymbol{\Psi}_j) = K_{\mathrm{c}}$, then due to Theorem 8.1 we have

$$\operatorname{rank}(\boldsymbol{\Psi}(\mathcal{L})) \geq \max_{i=1,\dots,R}\{\operatorname{rank}(\boldsymbol{\Psi}_i)\} = \max\left\{\operatorname{rank}(\boldsymbol{\Psi}_j), \max_{i\neq j}\{\operatorname{rank}(\boldsymbol{\Psi}_j)\}\right\} = K_{\mathrm{c}}. \qquad \blacksquare$$

This way, we know that in order to guarantee identifiability we can select $L_j \geq \max\{K_c, z_j + 1\}$ rows from any factor matrix $j$, and $L_i \geq \max\{1, z_i + 1\}$ from the other factors with $i \neq j$. In many real scenarios we will have $\{z_i = 0\}_{i=1}^R$ since no entry in $\{\mathbf{U}_i\}_{i=1}^R$ will exactly be zero. In those situations we will require to select at least $L = \sum_{i=1}^R L_i \geq K_c + R - 1$ elements.

## 8.2 Greedy method

We here propose an efficient greedy algorithm to select the rows of $\{\mathbf{U}_i\}_{i=1}^R$ with near-optimality guarantees with respect to the frame potential. As we did for the case with a dense core, we start by finding an expression for the frame potential of a Khatri-Rao product in terms of its factors.

The Grammian matrix $\mathbf{T}(\mathcal{L})$ of a diagonal core tensor decomposition has the form

$$\begin{aligned}\mathbf{T} = \mathbf{\Psi}^H\mathbf{\Psi} &= (\mathbf{\Psi}_1 \odot \cdots \odot \mathbf{\Psi}_R)^H (\mathbf{\Psi}_1 \odot \cdots \odot \mathbf{\Psi}_R) \\ &= \mathbf{\Psi}_1^H\mathbf{\Psi}_1 \circ \cdots \circ \mathbf{\Psi}_R^H\mathbf{\Psi}_R = \mathbf{T}_1 \circ \cdots \circ \mathbf{T}_R.\end{aligned}$$

Using this expression, the frame potential of a Khatri-Rao product becomes

$$\mathrm{FP}(\mathbf{\Psi}) = \mathrm{tr}\left\{\mathbf{T}^H\mathbf{T}\right\} = \|\mathbf{T}\|_F^2 = \|\mathbf{T}_1 \circ \cdots \circ \mathbf{T}_R\|_F^2 . \tag{8.1}$$

For brevity, we will denote the frame potential as an explicit function of the selected set as

$$P(\mathcal{L}) \coloneqq \mathrm{FP}(\mathbf{\Psi}(\mathcal{L})) = \|\mathbf{T}_1(\mathcal{L}_1) \circ \cdots \circ \mathbf{T}_R(\mathcal{L}_R)\|_F^2 . \tag{8.2}$$

Unlike in the dense core case, the frame potential of a Khatri-Rao product cannot be separated in terms of the frame potential of its factors. Instead, (8.1) decomposes the frame potential using the Hadamard product of the Grammian of the factors.

### 8.2.1 Submodularity of $P(\mathcal{L})$

Since $P(\mathcal{L})$ does not directly satisfy the submodular near-optimality conditions, we propose using the following set function $Q : 2^{\mathcal{N}} \to \mathbb{R}$ as a submodular surrogate for the frame potential

$$\begin{aligned}Q(\mathcal{S}) &\coloneqq P(\mathcal{N}) - P(\mathcal{N} \setminus \mathcal{S}) \\ &= \|\mathbf{T}_1(\mathcal{N}_1) \circ \cdots \circ \mathbf{T}_r(\mathcal{N}_r)\|_F^2 - \|\mathbf{T}_1(\mathcal{N}_1 \setminus \mathcal{S}_1) \circ \cdots \circ \mathbf{T}_R(\mathcal{N}_R \setminus \mathcal{S}_R)\|_F^2 .\end{aligned} \tag{8.3}$$

We can show that this function satisfies the conditions required for near-optimality of the greedy methods.

**Theorem 8.2.** *The set function $Q(\mathcal{S})$ defined in (8.3) is a normalized, monotone non-decreasing, submodular function for all subsets of $\mathcal{N} = \bigcup_{i=1}^R \mathcal{N}_i$.*

*Proof.* See Appendix A.4. ■

Using $Q$ and imposing the identifiability constraints defined in Section 8.1 we can write the related optimization problem for the minimization of the frame potential as

$$\begin{aligned} \underset{\mathcal{S}\subseteq\mathcal{N}}{\text{maximize}} \quad & Q(\mathcal{S}) \\ \text{subject to} \quad & \mathcal{S} \in \mathcal{I} \quad \mathcal{I} = \mathcal{I}_u \cap \mathcal{I}_p \\ & \mathcal{I}_u = \{\mathcal{A} \subseteq \mathcal{N} : |\mathcal{S}| \leq N - L\} \\ & \mathcal{I}_p = \{\mathcal{A} \subseteq \mathcal{N} : |\mathcal{A} \cap \mathcal{N}_i| \leq \beta_i \quad i = 1, \ldots, R\} \\ & \beta_j = N_j - \max\{K_{\text{c}}, z_j\} \\ & \beta_i = N_i - \max\{1, z_i + 1\} \text{ for } i \neq j, \end{aligned} \tag{8.4}$$

where the choice of $j$ is arbitrary, and can be set depending on the application. For example, with some space-time signals it is more costly to sample space than time. Hence, in those cases, $j$ would generally be the temporal domain.

This is a submodular maximization problem with a truncated partition matroid constraint [cf. Example 5.2 from Definition 5.4]. Thus, from Theorem 5.2, we know that greedy maximization of (8.4) using Algorithm 2 has a multiplicative near-optimal guarantee.

**Corollary 8.2.** *The solution $\mathcal{S}_{greedy}$ to (8.4) $\mathcal{S}_{greedy}$ obtained using the greedy Algorithm 2 is 1/2-near-optimal. That is*

$$Q(\mathcal{S}_{greedy}) \geq \frac{1}{2} Q(\mathcal{S}^*).$$

As for the dense core case, we can also provide a bound on the near-optimality of of the greedy solution with respect to the frame potential.

**Theorem 8.3.** *The solution set $\mathcal{L}_{greedy}$ to (8.4) obtained using the greedy Algorithm 2 is near-optimal with regards to the frame potential as*

$$P(\mathcal{L}_{greedy}) \leq \gamma P(\mathcal{L}^*), \tag{8.5}$$

*with* $\gamma = \frac{1}{2}\left( \|\mathbf{T}_1(\mathcal{N}_1) \circ \cdots \circ \mathbf{T}_r(\mathcal{N}_r)\|_F^2 \, \frac{K}{L_{min}^2} + 1 \right)$.

*Proof.* The proof is analogous to the one of Theorem 7.3 but uses (8.1) instead of (7.9) in the derivation. ■

### 8.2.2 Computational complexity

The computational complexity of the greedy method is now governed by the complexity of computing the Grammian matrices $\mathbf{T}_i$. This can greatly be improved if before starting the iterations, one precomputes all the outer products in $\{\mathbf{T}_i\}_{i=1}^R$. Doing this has a computational complexity of $\mathcal{O}(N_{\max} K_{\text{c}}^2)$. Then, in every iteration, the evaluation of $P(\mathcal{L})$ would only cost $\mathcal{O}(RK_{\text{c}}^2)$ operations. Further, because in every iteration we need to query $\mathcal{O}(N_i)$ elements on each domain, and we run the algorithm for $N - L$ iterations, the total time complexity of the iterations is $\mathcal{O}(RN_{\max}^2 K_{\text{c}}^2)$. This term dominates over the complexity of the precomputations, and thus can be treated as the worst case complexity of the greedy method.

The particularization of Algorithm 2 to solve (8.4) is given in Algorithm 5.

**Algorithm 5** Greedy maximization of $Q$

**Require:** $\{\mathbf{U}_i\}_{i=1}^R$, $\{\beta_i\}_{i=1}^R$, $L$, $\mathcal{S} = \varnothing$ and $\mathcal{D} = \{1, \dots, R\}$.

**for** $k \leftarrow 1$ **to** $N - L$
  **for** $i \in \mathcal{D}$
    $s_i^* \leftarrow \arg\min_{s_i \notin \mathcal{S}_i} ||\mathbf{T}_i(\mathcal{N}_i \setminus \mathcal{S}_i \cup \{s_i\}) \circ \prod_{j \neq i} \circ \mathbf{T}_j(\mathcal{N}_j \setminus \mathcal{S}_j)||_F^2$
  **end**
  $i^* \leftarrow \arg\min_{i \in \mathcal{D}} ||\mathbf{T}_i(\mathcal{N}_i \setminus \mathcal{S}_i \cup \{s_i^*\}) \circ \prod_{j \neq i} \circ \mathbf{T}_j(\mathcal{N}_j \setminus \mathcal{S}_j)||_F^2$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_{i^*}^*\}$
  **if** $|\mathcal{S} \cap \mathcal{N}_{i^*}| = N_{i^*} - \beta_{i^*}$ **then**
    $\mathcal{D} \leftarrow \mathcal{D} \setminus i^*$
  **end**
**end**
**return** $\mathcal{L} = \mathcal{N} \setminus \mathcal{S}$

### 8.2.3 Practical considerations

The proposed scheme suffers from the same issues as the greedy minimization of the frame potential in the dense core case. Namely, it tends to empty the domains sequentially, thus producing solutions which are tight on the identifiability constraints. Nevertheless, as we indicated for the dense core, the drop in performance associated with the proximity of the solutions to the constraints can be reduced by giving some slack to the constraints.

## 8.3 Convex alternative

Finding a convex alternative for the diagonal case is more involved than in the non-diagonal case, since the log det of a Hadamard product cannot be decomposed in terms of the log det of its factors. Indeed, in this case

$$\begin{aligned} \log\det\{\mathbf{T}\} &= \log\det\left\{\mathbf{U}^H \left(\mathbf{\Phi}_1^H \mathbf{\Phi} \otimes \dots \mathbf{\Phi}_R^H \mathbf{\Phi}_R\right) \mathbf{U}\right\} \\ &= \log\det\left\{\mathbf{U}^H \left(\mathrm{diag}(\mathbf{w}_1) \otimes \mathrm{diag}(\mathbf{w}_R)\right) \mathbf{U}\right\} \\ &= \log\det\left\{\mathbf{U}^H \mathrm{diag}(\mathbf{w}_1 \otimes \cdots \otimes \mathbf{w}_R) \mathbf{U}\right\}, \end{aligned} \tag{8.6}$$

and it is clear that the Khatri-Rao product that forms $\mathbf{U}$ does not allow for further simplification. This means that even if we relax the Boolean constraints on the selection vectors, the relaxed suboptimal sampling problem

$$\begin{aligned} \underset{\{\mathbf{w}_i\}_{i=1}^R}{\text{maximize}} \quad & \log\det\left\{\mathbf{U}^H \mathrm{diag}(\mathbf{w}) \mathbf{U}\right\} \\ \text{subject to} \quad & \sum_{i=1}^R \mathbf{1}^T \mathbf{w}_i \leq L \\ & \mathbf{0} \preceq \mathbf{w} \preceq \mathbf{1} \\ & \mathbf{w} = \mathbf{w}_1 \otimes \cdots \otimes \mathbf{w}_R \end{aligned} \tag{8.7}$$

is not convex due to the non-linearity of the last constraint.

A possible convex relaxation for this problem can be obtained by relaxing the Kronecker structure of the optimization variable and optimizing

$$\begin{aligned} \underset{\mathbf{w}}{\text{maximize}} \quad & \log\det\left\{\mathbf{U}^H \text{diag}(\mathbf{w})\mathbf{U}\right\} \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{w} \leq L' \\ & \mathbf{0} \preceq \mathbf{w} \preceq \mathbf{1}, \end{aligned} \tag{8.8}$$

where $L'$ is a heuristic regularization constant introduced to indirectly control the number of selected sensors.

The Kronecker structure can later be recovered by finding a good sparse approximation of $\mathbf{w}$ in terms of a Kronecker product

$$\begin{aligned} \underset{\{\mathbf{w}_i\}_{i=1}^R}{\text{minimize}} \quad & \|\mathbf{w} - \mathbf{w}_1 \otimes \cdots \otimes \mathbf{w}_R\|_2^2 + \lambda \sum_{i=1}^R \mathfrak{R}_i(\mathbf{w}_i) \\ \text{subject to} \quad & \mathbf{0} \preceq \mathbf{w}_i \preceq \mathbf{1} \quad i = 1, \ldots, R, \end{aligned} \tag{8.9}$$

where the $\mathfrak{R}_i(\mathbf{w}_i)$ are sparsity promoting regularizer terms for the vectors $\{\mathbf{w}_i\}$.

Alternatively, one can restructure $\mathbf{w}$ into a tensor $\boldsymbol{\mathcal{W}}$, such that $\text{vec}(\boldsymbol{\mathcal{W}}) = \mathbf{w}$ giving

$$\begin{aligned} \underset{\{\mathbf{w}_i\}_{i=1}^R}{\text{minimize}} \quad & \|\boldsymbol{\mathcal{W}} - 1 \bullet_1 \mathbf{w}_1 \bullet_2 \cdots \bullet_R \mathbf{w}_R\|_F^2 + \lambda \sum_{i=1}^R \mathfrak{R}_i(\mathbf{w}_i) \\ \text{subject to} \quad & \mathbf{0} \preceq \mathbf{w}_i \preceq \mathbf{1} \quad i = 1, \ldots, R. \end{aligned} \tag{8.10}$$

This alternative formulation suggests that solving (8.9) is equivalent to finding a rank-one decomposition of $\boldsymbol{\mathcal{W}}$ given by sparse factors. For the general case with $R > 2$ there is not much literature addressing this problem. Nevertheless, for the $R = 2$ case one can rewrite (8.10) as

$$\begin{aligned} \underset{\mathbf{w}_1,\mathbf{w}_2}{\text{minimize}} \quad & \left\|\mathbf{W} - \mathbf{w}_1\mathbf{w}_2^T\right\|_F^2 + \lambda\mathfrak{R}_1(\mathbf{w}_1) + \lambda\mathfrak{R}_2(\mathbf{w}_2) \\ \text{subject to} \quad & \mathbf{0} \preceq \mathbf{w}_i \preceq \mathbf{1} \quad i = 1, 2. \end{aligned} \tag{8.11}$$

This last problem is very similar to the sparse Singular Value Decomposition introduced by Lee et al. in [70]. In fact, their formulation of this problem only differs from (8.11) in the last constraint, which defines a subset of the original constraint

$$\begin{aligned} \underset{\mathbf{w}_1,\mathbf{w}_2}{\text{minimize}} \quad & \left\|\mathbf{W} - \mathbf{w}_1\mathbf{w}_2^T\right\|_F^2 + \lambda\mathfrak{R}_1(\mathbf{w}_1) + \lambda\mathfrak{R}_2(\mathbf{w}_2) \\ \text{subject to} \quad & \|\mathbf{w}_i\|_2 = 1 \quad i = 1, 2. \end{aligned} \tag{8.12}$$

In their paper [70], Lee et al. propose an efficient solution to the sparse SVD problem in terms of an alternative minimization of (8.12) for fixed $\mathbf{w}_1$, first, and $\mathbf{w}_2$, second. Thus, as a heuristic method, one could try to approximate (8.7) by first solving (8.8), and then solving (8.12) using the sparse SVD algorithm to obtain a rank-1 solution. Finally, in order to recover a discrete solution one would need to threshold the obtained results.

It is important to highlight though, that taking this approach does not guarantee the solution to be optimal, nor feasible. It could very well be the case that the solution to (8.12) does not satisfy the $\mathbf{1}^T\mathbf{w} \leq L'$ constraint of the original problem. Furthermore, this method has many extra levels of complexity compared to the proposed greedy optimization using Algorithm 5, and thus it is not suitable for large scale problems. Moreover, for $R > 2$, there is not even a convex optimization alternative to the proposed algorithm.

# 9 Numerical results

In this chapter we will illustrate the applicability of the proposed framework through a series of examples. First, we will show some results obtained on a series of synthetic dataset to compare the performance of the different near-optimal algorithm. Then, we will focus on a few real dataset of higher dimensionality to show the benefits of the developed framework[1].

## 9.1 Synthetic data

### 9.1.1 Dense core

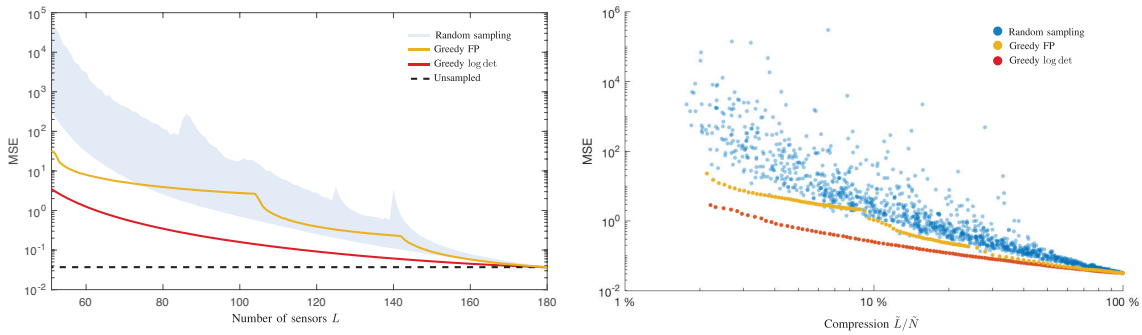



Figure 9.1: Performance comparison for the dense core case. Results obtained for $R = 3$ with $N_1 = 50, N_2 = 60, N_3 = 70$, and with $K_1 = 10, K_2 = 20, K_3 = 15$.

We compare the performance in terms of the theoretical MSE in (7.6) of both greedy algorithms developed for the dense core case (Algorithm 3 and Algorithm 4) to a random sampling scheme based on randomly selecting rows of $\mathbf{U}_i$ such that the resulting subset of samples is Kronecker-structured. Only those selections that satisfy the identifiability constraints in (7.11) are treated as valid. We note that the time complexity of evaluating $M$ times the MSE for a Kronecker-structured sampler is $\mathcal{O}(MN_{\max}^2 K_{\max})$. For this reason, using random sampling with a large $M$ to guess a good sparse sampler is computationally intense.

To perform this comparison, we draw 100 realization of three random Gaussian matrices $\{\mathbf{U}_i \in \mathbb{R}^{N_i \times K_i}\}_{i=1}^{R=3}$ with different dimensionalities. For each of these models we obtain the solution of (7.5) and (7.12) for different number of sensors $L$ using Algorithm 3 and Algorithm 4, respectively. We also compute $M = 100$ realizations of random sampling for each $L$. Fig. 9.1 shows the results of these experiments. The plot

[1]The code to reproduce the results presented in this chapter can be found at https://gitlab.com/gortizji/sparse_tensor_sensing.

on the left shows the average performance over the different models against number of sensors and the blue shaded area represents the 10-90 percentile average interval of the random sampling scheme. We underline that the absolute values in these plots do not mean anything. Only the relative differences in orders of magnitude among the different sampling methods should be taken into account. Furthermore, because the estimation performance is heavily influenced by its related number of samples $\tilde{L}$, and noting the fact that a value of $L$ may lead to different $\tilde{L}$, we also present the performance comparison for one model realization against the relative number of samples $\tilde{L}/\tilde{N}$ so that differences in the informative quality of the selections are highlighted.

In light of these results, it is clear that both proposed methods outperform a naïve random selection of rows. As expected, the differences in performance get reduced when we increase the number of sensors. Moreover, we see that the greedy algorithm based on $D$-optimality shows the best performance with regards to MSE. Whereas, optimizing the frame potential, seems to perform slightly worse, especially for the lower compression rates. The algorithm based on frame potential ranks consistently on par with the best realizations of random sampling as shown in the right plot.

The plots in Fig. 9.1 also illustrate some important features of the proposed sparse sampling methods. As seen from the plot on the right, greedy maximization of $\log\det \mathbf{T}(\mathcal{L})$ produces solutions with less compression, i.e. the red points in the right side of the plot are more dense. On the other hand, greedy minimization of the frame potential achieves lower rates of compression with the same number of sensors. Besides, the $D$-optimal algorithm has a continuous and smooth improvement in performance for increasing number of sensors, the algorithm based on frame potential shows three bumps (note $R = 3$) in the performance curve. This is a consequence of the behaviour described in Chapter 7, where we mentioned the tendency of Algorithm 4 to meet the identifiability constraints with equality. As we increase $L$ the solutions of Algorithm 4 increase in cardinality by adding more elements to a single domain until this is full, and then continue to the next domain. The bumps in Fig. 9.1 correspond precisely to those moments, when the cardinality of one domain becomes maximum and the cardinality of another domain starts to increase.

### 9.1.2 Diagonal core

We perform the same experiment for the diagonal core case, but now only for the algorithm based on frame potential and random sampling. The results are shown in Fig. 9.2. Again we see that the proposed algorithm outperforms a random selection of rows, especially when collecting just a few samples. Furthermore, as happened in the dense core case, the performance curve of Algorithm 5 follows a stairway shape.

## 9.2 Sampling a dynamical point cloud

We move now to study the applicability of the sparse tensor sampling framework to some real dataset with very high dimensionality. First, we test our sensor selection algorithms for a dense core decomposition on a three-dimensional dynamic point cloud composed of $N_1 = 1502$ points and $N_2 = 573$ frames, i.e. a 3rd-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{1502\times 573\times 3}$ with

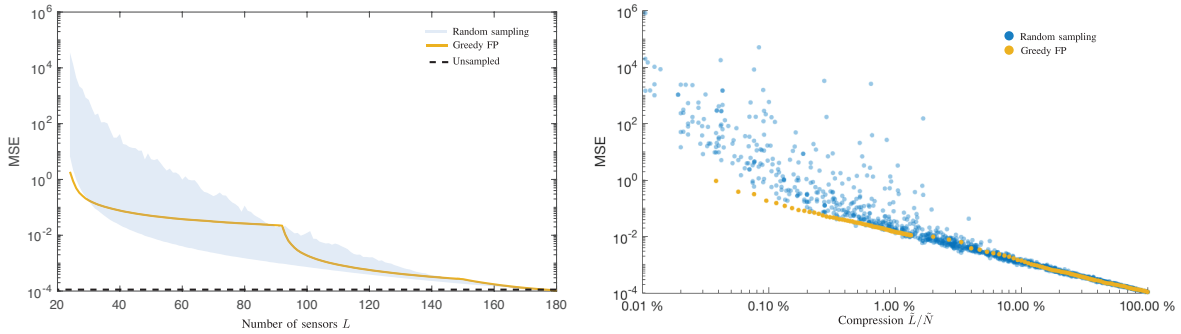



Figure 9.2: Performance comparison for the diagonal core case. Results obtained for $R = 3$ with $N_1 = 50, N_2 = 60, N_3 = 70$, and with $K_c = 20$.

more than $\tilde{N} = 2.5$ million entries, that represents the movement of a dancer. At this point, we highlight the need for our framework, since it is obvious that designing an unstructered sampling set with 2.5 million candidate locations is unfeasible with our current computing resources.

A model of $\boldsymbol{\mathcal{X}}$ in the form of (5.1) can be obtained by regarding $\boldsymbol{\mathcal{X}}$ as a graph signal. To represent the data as a graph signal we build a spatial 5-nearest-neighbor graph with the time-averaged position of the $N_2$ markers as suggested in [12]; and consider time as a cycle graph with $N_1$ vertices. The resulting product graph consists of more than $850,000$ nodes that serve as support for a vector-valued graph signal that represents the movement of the dancer.

In this context, $\boldsymbol{\mathcal{X}}$ can be decomposed as

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}_{\mathrm{f}} \bullet_1 \mathbf{U}_{\mathcal{G}} \bullet_2 \mathbf{U}_{\mathcal{T}}.$$

Here, $\mathbf{U}_{\mathcal{G}} \in \mathbf{C}^{N_1 \times N_1}$ and $\mathbf{U}_{\mathcal{T}} \in \mathbf{C}^{N_2 \times N_2}$ are the eigenbasis of the Laplacians of the spatial and temporal graphs, respectively [cf. Section 2.2.3]; and $\boldsymbol{\mathcal{X}}_{\mathrm{f}} \in \mathbf{C}^{N_1 \times N_2 \times N_3}$ is the graph spectrum of $\boldsymbol{\mathcal{X}}$, whose first and second modes correspond to the spatial and temporal frequencies, respectively.

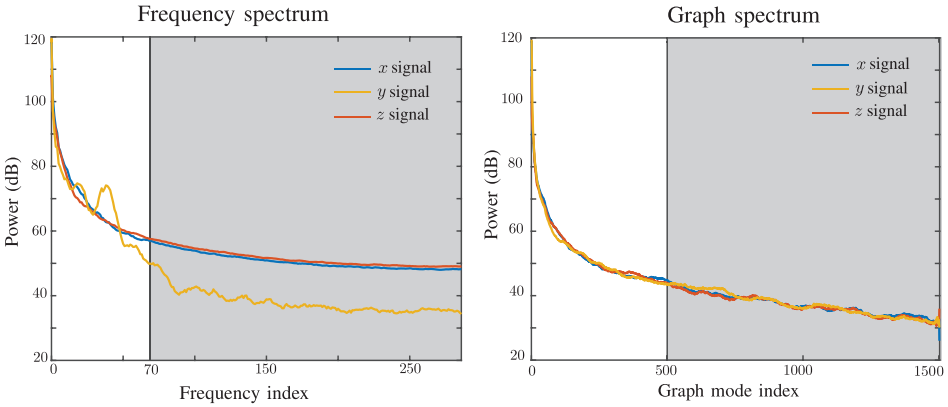



Figure 9.3: Maximum power of $\boldsymbol{\mathcal{X}}_{\mathrm{f}}$ in its temporal and graph spectral modes. The shaded area represents the modes that are removed to arrive at (9.1).

| Method | Number of sensors | $\tilde{L}/\tilde{N}$ | Error |
|---|---|---|---|
| Frame potential | 600 | 4.57% | 3.47% |
| $D$-optimality | 600[2] | 4.57% | 3.61% |

Table 9.1: Performance of proposed sparse tensor sampling algorithms on dancer point cloud.

Fig. 9.3 shows the maximum energy of $\boldsymbol{\mathcal{X}}_{\text{f}}$ along the spatial and temporal frequencies. A visual inspection of the spectral decomposition of these signals shows that most of the energy is confined in the first few eigenmodes of the temporal and spatial graphs. In particular, in our simulations we limit the support of the signals to the first $K_1 = 500$ and $K_2 = 70$ spatial and temporal frequencies, respectively. Considering the dramatic drop in signal energy for $K_1 = 500$ and $K_2 = 70$ (more than 60 dB), a good approximation of $\boldsymbol{\mathcal{X}}$ can be obtained using

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{G}} \bullet_1 \mathbf{U}_1 \bullet_2 \mathbf{U}_2 \bullet_3 \mathbf{U}_3 \tag{9.1}$$

with $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{X}}_{\text{f}}(1:K_1, 1:K_2, :)$, $\mathbf{U}_1 = \mathbf{U}_{\mathcal{G}}(:, 1:K_1)$, $\mathbf{U}_2 = \mathbf{U}_{\mathcal{T}}(:, 1:K_2)$,and $\mathbf{U}_3 = \mathbf{I}_3$.

Using this decomposition, we obtain different sampling sets running our both near-optimal algorithms with $L = 600$ and keeping the third domain unsampled. We also run 1000 realizations of random sampling with the same $L$. To compare the performance of all methdos, we compute the error in the estimation of $\boldsymbol{\mathcal{X}}$ as

$$\text{Error} = \frac{\left\| \boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}} \right\|_F}{\|\boldsymbol{\mathcal{X}}\|_F}.$$

With these parameters, however, only the method based on the frame potential is able to obtain a non-singular solution. In the case of the $D$-optimal method, due to the significant differences in the dimensionalities of $\mathbf{U}_1$ and $\mathbf{U}_2$ Algorithm 3 does not give a sampling set that produces a $\boldsymbol{\Psi}_1(\mathcal{L}_1)$ with full-column rank. For random sampling also, the selection of rows is so poor that none of its subset candidates results in an identifiable system after subsampling.

An illustration of the quality of the results of Algorithm 4 for two different frames of the point cloud video is shown in Fig. 9.4. Even with this amount of compression on such a highly dimensional signal the reconstruction is very accurate. The point cloud's shape is hardly distorted and the point deviations are very small. Furthermore, the magnitude of the relative error, as shown in Table 9.1 is also negligible. For the sake of completeness, and to have something to compare the results of Algorithm 4 with, we run Algorithm 3 for $\mathbf{U}_1$ and $\mathbf{U}_2$ separately, specifying the resulting $L_1$ and $L_2$ from the solution to the frame potential minimization as dimensionalities beforehand. This amounts to computing twice a single-domain subsampling set using the method proposed in [51]. Despite the good performance of this approach, we highlight that in many situations one might not know the desired $L_1$ and $L_2$ and would like the algorithm

[2]This is set by fixing the dimensions of the single domain sparse samplers using the results from the frame potential optimization ($L_1 = 75$ and $L_2 = 525$). This is therefore the solution to a different problem than (6.10).

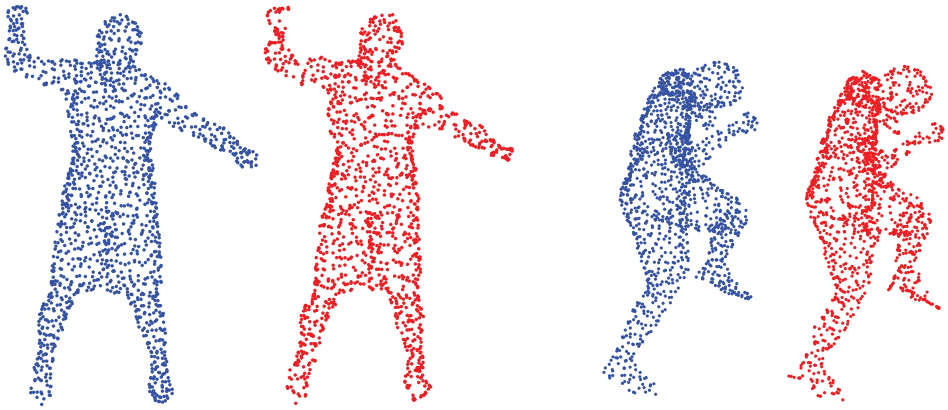


Figure 9.4: Two frames of the dancer dynamic point cloud. The blue dots correspond to the original data and the red dots to the subsampled version.

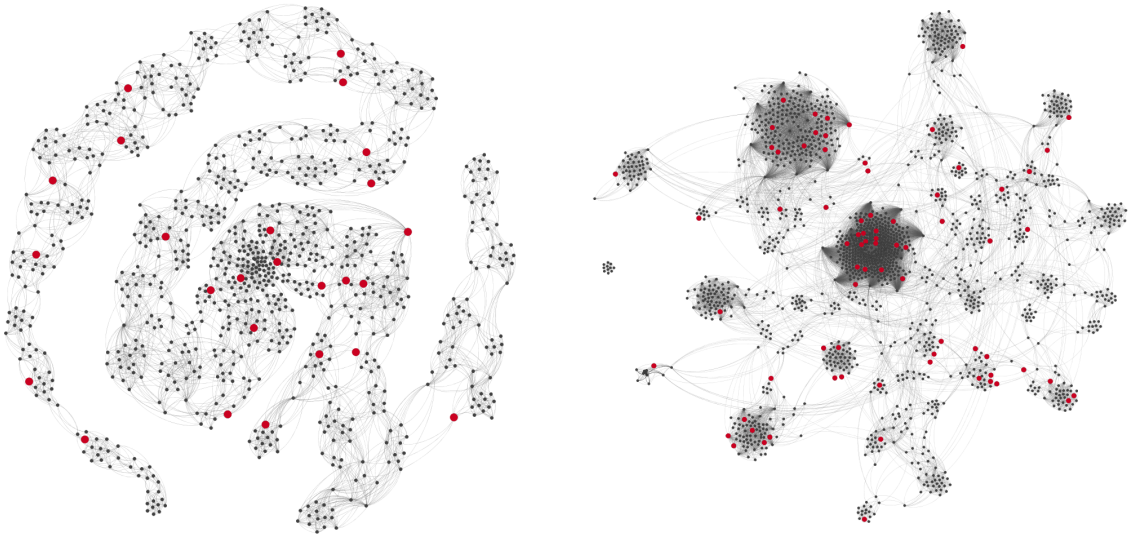
to compute these internally. In particular, if one is interested in solving (6.10) swaping the objective function and the cardinality constraints, i.e. minimizing number of sensors subject to a specified performance, separately solving the single-domain optimizations would yield a suboptimal sensor distribution.

## 9.3 Active learning for recommender systems

Current recommendation algorithms seek solving an estimation problem of the form: given the past recorded preferences of a set of users, what is the *rating* that these would give to a set of products? In this thesis, in contrast, we focus on the data acquisition phase of the recommender system, which is also referred to as active learning/sampling. In particular, we claim that by carefully designing which users to poll and on which items, we can obtain an estimation performance on par with the state-of-the-art methods, but using only a fraction of the data that current methods require, and using a simple least-squares estimator.

We showcase this idea on the MovieLens $100k$ dataset [71] that contains partial ratings of $N_1 = 943$ users over $N_2 = 1682$ movies which are stored in a second-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_1 \times N_2}$. A model of $\boldsymbol{\mathcal{X}}$ in the form of (5.1) can be obtained by viewing $\boldsymbol{\mathcal{X}}$ as a signal that lives on a graph. In particular, the first two modes of $\boldsymbol{\mathcal{X}}$ can be viewed as a signal defined on the Cartesian product of a user and movie graph, respectively. These two graphs, shown in Fig. 9.5, are provided in the dataset and are two 10-nearest-neighbors graphs created based on the user and movie features.

The bandlimitidness of $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_1 \times N_2}$ has already been exploited to impute its missing entries [72, 73]. In our experiments, we use $K_1 = K_2 = 20$. Using this representation, we run our greedy algorithm with $L = 100$, resulting in a selection of $L_1 = 25$ user and $L_2 = 75$ movie vertices, i.e., 1875 vertices in the product graph. Fig. 9.5, shows the sampled users and movies, i.e., users to be probed for movie ratings. The user graph [cf. Fig. 9.5a] is made out of small clusters connected in a chain-like structure, resulting in a uniformly spread distribution of observed vertices. On the other hand, the movies graph [cf. Fig. 9.5b] is made out of a few big and small clusters. Hence, the



(a) User graph (b) Movie graph

Figure 9.5: User and movie networks. The red (black) dots represent the observed (unobserved) vertices. Visualization obtained using Gephi [74].

proposed active querying scheme assigns more observations to the bigger clusters and fewer observations to the smaller ones.

To evaluate the performance of our algorithm, we compute the RMSE of the estimated data using the test mask provided by the dataset. Nevertheless, since our active query method requires access to ground truth data (i.e., we need access to the samples at locations suggested by the greedy algorithm) which is not provided in the dataset, we use GRALS [75] to complete the matrix, and use its estimates when required. A comparison of our algorithm to the performance of the state-of-the-art methods run on the same dataset is shown in Table 9.2. In light of these results, it is clear that a proper design of the sampling set allows to obtain top performance with significantly fewer ratings, i.e., about an order of magnitude, and using a much simpler non-iterative estimator.

| **Method** | **Number of samples** | **RMSE** |
|---|---|---|
| GMC [73] | 80,000 | 0.996 |
| GRALS [75] | 80,000 | 0.945 |
| sRGCNN [23] | 80,000 | 0.929 |
| GC-MC [28] | 80,000 | **0.905** |
| Our method | **1,875** | 0.9347 |

Table 9.2: Performance on MovieLens 100$k$. Baseline scores are taken from [28].

## 9.4 Multiuser source separation

In multiple-input multiple-output (MIMO) communications [76], the use of rectangular arrays [77] allows to separate signals coming from different azimuth and elevation angles, and it is common that users transmit data using different spreading codes to reduce the interference from other sources. Reducing hardware complexity by minimizing the number of antennas and samples to be processed is an important concern in the design of MIMO receivers. This design can be seen as a particular instance of sparse tensor sampling.

We consider a scenario with $K_{\mathrm{c}}$ users located at different angles of azimuth ($\phi$) and elevation ($\theta$) transmitting using unique spreading sequences of length $N_3$. The receiver consists of a uniform rectangular array (URA) with antennas located on a $N_1 \times N_2$ grid. Each time instant, every antenna receives [77]

$$x(r,l,m,n) = \sum_{k=1}^{K_{\mathrm{c}}} s_k(r) c_k(l) e^{j2\pi n \Delta_x \sin\theta_k} e^{j2\pi m \Delta_y \sin\phi_k} + w(r,l,m,n),$$

where $s_k(r)$ the symbol transmitted by user $k$ in the $r$th symbol period; $c_k(l)$ the $l$th sample of the spreading sequence of the $k$th user; $\Delta_x$ and $\Delta_y$ the antenna separations in wavelengths of the URA in the $x$ and $y$ dimensions, respectively; and $\phi_k$ and $\theta_k$ the azimuth and elevation coordinates of user $k$, respectively; and where $w(r,l,m,n)$ represents an additive white Gaussian noise term with zero mean and variance $\sigma^2$. For the $r$-th symbol period, all these signals can be collected in a 3rd-order tensor $\boldsymbol{\mathcal{X}}(r) \in \mathbb{C}^{N_1 \times N_2 \times N_3}$ that can be decomposed as

$$\boldsymbol{\mathcal{X}}(r) = \boldsymbol{\mathcal{S}}(r) \bullet_1 \mathbf{U}_1 \bullet_2 \mathbf{U}_2 \bullet_3 \mathbf{U}_3 + \boldsymbol{\mathcal{W}}(r)$$

where $\mathbf{U}_1 \in \mathbb{C}^{N_1 \times K_{\mathrm{c}}}$ and $\mathbf{U}_2 \in \mathbb{C}^{N_2 \times K_{\mathrm{c}}}$ are the array responses for the $x$ and $y$ directions, respectively; $\mathbf{U}_3 \in \mathbb{C}^{N_3 \times K_{\mathrm{c}}}$ contains the spreading sequences of all users in its columns; and $\boldsymbol{\mathcal{S}}(r) \in \mathbb{C}^{K_{\mathrm{c}} \times K_{\mathrm{c}} \times K_{\mathrm{c}}}$ is a diagonal tensor that stores the symbols of all users for the $r$th symbol period on its diagonal.

We simulate this setup using $K_{\mathrm{c}} = 10$ users that transmit BPSK symbols with different random powers and that are equispaced in azimuth and elevation. We use

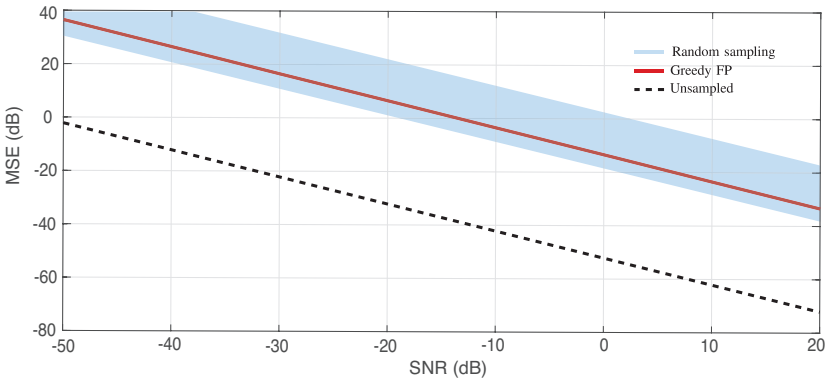



Figure 9.6: MSE of symbol reconstruction. $N_1 = 50$, $N_2 = 60$, $N_3 = 100$, and $L = 15$.

a rectangular array with $N_1 = 50$ and $N_2 = 60$ for the ground set locations of the antennas, and binary random spreading sequences of length $N_3 = 100$. With these parameters, each $\boldsymbol{\mathcal{X}}(r)$ has $300,000$ entries. We generate many realizations of these signals for different levels of signal-to-noise ratio (SNR) and sample the resulting tensors using the greedy algorithm for the diagonal core case with $L = 15$, resulting in a relative number of samples of 0.048%. The results are depicted in Fig. 9.6, where the blue shaded area represents the MSE obtained with the best and worst random samplers. As expected, the MSE of the reconstruction decreases exponentially with the SNR. For a given MSE, achieving maximum compression requires transmitting with a higher SNR of about 30dB than the one needed for no compression. Besides, we see that our proposed greedy algorithm consistently performs as well as the best random sampling scheme.

# 10 Conclusions

Multidomain graph data is becoming more common every day, and with the advent of this type of signals, classical signal processing techniques need to be reinvented. In this thesis, we have made two contributions to the processing and understanding of multidomain graph signals, which have lead to the proposition of two new frameworks that generalize tools from machine learning and signal processing to multidomain data with an irregular support.

In Part I, we have proposed an extension of the geometric deep learning framework to handle multidomain graph datasets. We refer to it as *multidomain geometric deep learning*. We have defined a new type of graph convolutional layer that can exploit correlations among multiple graph domains, and particularize it to time-varying graph signals. This layer has a fixed number of parameters, it is localized in space and time, can be written in terms of a universal graph filter and runs efficiently on a GPU. Furthermore, the experiments performed on a synthetic dataset have shown that 2D-GCNNs generalize better and are more robust than 1D-GCNNs.

In Part II, we have focused on the design of sparse samplers for multidomain graph signals, also leading to a new theory of sparse sampling for linear inverse problems with tensors. We refer to this extension as *sparse tensor sampling*. We have seen that by using samplers with a Kronecker structure we can overcome the curse of dimensionality, and design efficient subsampling schemes that guarantee a good performance for the reconstruction of tensor signals. In this sense, we have worked under two different assumptions: that signals can be decomposed using a multilinear model with a dense core, or with a diagonal core. For both cases, we have provided several near-optimal greedy algorithms to design the sampling sets, and provided bounds on their resulting performance. We have extensively evaluated these algorithms, as well as shown their applicability to real life problems.

## Future directions

We summarize a list of ideas that could lead to future extensions of the work presented in this thesis:

- **Validation of TV-GCNNs on real data**

  In Chapter 4 of Part I, we compared the performance of the TV-GCNNs to the *de facto* standard 1D-GCNNs on a controllable synthetic dataset. Nevertheless, the assessment of the performance of the TV-GCNN with real data needs still to be performed. In this sense, we propose targeting three different applications in which exploiting the space-time interactions of the graph data might lead to an improvement of the current state-of-the-art. They are:

- *EEG decoding*: A hot topic in neuroscience is the classification of brain signals for the development of brain computer interfaces (BCI) [78]. In this context, EEG signals can be viewed as time-varying graph processes where the spatial domain is represented by a graph of electrodes that are placed on the human skull.
- *Traffic prediction*: As mentioned in the introduction, monitoring traffic in a city is a major concern for local governments. TV-GCNNs could be a potential tool for traffic prediction.
- *Characterization of epidemics*: One of the main issues in epidemiology is the statistical characterization of the dynamics of a disease outbreak given the recorded time variations of the disease state around the world [79]. To tackle this problem, epidemiologists rely on complicated systems of coupled non-linear differential equations that are governed by a few characteristic parameters, i.e., probability of contagion, illness duration, or death rate. In this context, TV-GCNNs could be used as powerful regressors to infer these parameters from data.

- **2D-GCNNs with other filter types**

  In this thesis, we have only focused on the extension of FIR graph convolutional layers to multidomain graph signals. However, many other types of filters have also been proposed in the literature, e.g. ARMA graph filters [18] and node and edge-varying filters [80, 81]. A possible extension of this work would be the definition of new types of multidomain graph CNNs using these alternative filter paradigms.

- **Sparse tensor sampling with smoothness prior**

  In Part II, we always assumed that the support of the tensor signals was perfectly known. However, in many real scenarios, this prior knowledge is not available. An alternative assumption is the smoothness prior [82], i.e. $\|D(\mathbf{x})\| \leq \rho$ for some invertible operator $D$ such as the differential operator or the total-variation operator. The formulation of the sparse tensor sampling problem under this assumption is still an open problem.

- **Robust sparse tensor sampling**

  Working with single domain signals, Joshi and Boyd [58] showed that it is possible to design sampling schemes that are robust against bounded model misspecifications, i.e. with an unknown system matrix $\mathbf{\Psi}$, but knowing that this belongs to a set of possible system matrices $\mathcal{S} = \{\mathbf{\Psi} = \mathbf{\Psi}_0 + \mathbf{\Delta} : \|\mathbf{\Delta}\|_2 \leq \epsilon\}$. Extending this result to the multidomain case, where $\mathbf{\Psi}$ has a Kronecker, or Khatri-Rao structure, could be another possible extension of this thesis.

- **Sparse tensor sampling for detection or filtering**

  In this thesis, we have only focused on the design of sparse samplers for the estimation of tensor signals. Nevertheless, the sparse tensor sampling framework can also be extended to address different inference tasks such as filtering or detection, and hence generalize the results of the single domain theory.

# Proofs of Part II

# A

## A.1 Proof of Theorem 7.1

Let us start by proving normalization. Indeed

$$LD(\varnothing) = \sum_{i=1}^{R} \tilde{K}_{-i} \log\det\{\epsilon\mathbf{I}\} - R\tilde{K}\log\epsilon = \sum_{i=1}^{R} \tilde{K}_{-i} \log\epsilon^{K_i} - R\tilde{K}\log\epsilon$$
$$= \sum_{i=1}^{R} \tilde{K}\log\epsilon - R\tilde{K}\log\epsilon = 0.$$

Let $LD_i : 2^{\mathcal{N}_i} \to \mathbb{R}$ be of the form of the single-domain function which is optimized in [51] to obtain a near-optimal sensor selection with respect to $D$-optimality, i.e. $LD_i(\mathcal{L}_i) = \log\det\{\mathbf{T}_i(\mathcal{L}_i) + \epsilon\mathbf{I}\}$. In terms of these functions, $LD$ can be expressed as

$$LD(\mathcal{L}) = \sum_{i=1}^{R} \tilde{K}_{-i} LD(\mathcal{L} \cap \mathcal{N}_i) - R\tilde{K}\log\epsilon,$$

where we see that $LD(\mathcal{L})$ is a non-negative linear combination of $LD_i(\mathcal{L} \cap \mathcal{N}_i)$. Therefore, since the class of monotone non-decreasing submodular functions is closed under non-negative linear combinations, to prove submodularity and monotonicity of $LD$ it is sufficient to prove that all $LD_i(\mathcal{L} \cap \mathcal{N}_i)$ are monotone non-decreasing and submodular.

We know that $LD_i$ are submodular, and monotone non-decreasing functions for the subsets of $\mathcal{N}_i$ with $i = 1, \ldots, R$ [51]. We need to prove that their extension to the subsets of $\mathcal{N} = \bigcup_{i=1}^{R} \mathcal{N}_i$ defined as $LD_i^{\mathrm{e}}(\mathcal{L}) = LD_i(\mathcal{L} \cap \mathcal{N}_i)$ for $\mathcal{L} \subseteq \mathcal{N}$ is monotone non-decreasing and submodular.

Monotonicity of $LD_i^{\mathrm{e}}$ requires that for any $\mathcal{X} \subseteq \mathcal{Y} \subseteq \mathcal{N}$

$$LD_i^{\mathrm{e}}(\mathcal{X}) \leq LD_i^{\mathrm{e}}(\mathcal{Y}).$$

Substituting the definition of $LD_i^{\mathrm{e}}$, this becomes

$$LD_i(\mathcal{X} \cap \mathcal{N}_i) \leq LD_i(\mathcal{Y} \cap \mathcal{N}_i). \tag{A.1}$$

Since $\mathcal{X} \cap \mathcal{N}_i \subseteq \mathcal{Y} \cap \mathcal{N}_i \subseteq \mathcal{N}_i$, we know that (A.1) is always satisfied. Hence, $LD_i^{\mathrm{e}}$ is monotone non-decreasing.

For $LD_i^{\mathrm{e}}$ to be submodular we require that for any $x, y \in \mathcal{N} \setminus \mathcal{L}$ the following inequality is satisfied

$$LD_i^{\mathrm{e}}(\mathcal{L} \cup \{x\}) - LD_i^{\mathrm{e}}(\mathcal{L}) \geq LD_i^{\mathrm{e}}(\mathcal{L} \cup \{x, y\}) - LD_i^{\mathrm{e}}(\mathcal{L} \cup \{y\}). \tag{A.2}$$

Nevertheless, depending on wether $x, y$ belong to $\mathcal{N}_i$, their addition to $\mathcal{L}$ may not have any effect in the value of $LD_i^e$. This way, when both elements belong to $\mathcal{N}_i$, $LD_i^e$ is virtually the same function as $LD_i$. Hence, it satisfies (A.2). On the other hand, when at least one of the elements does not belong to $\mathcal{N}_i$, both sides of (A.2) take the same value. Therefore, proving submodularity for the remaining cases.

For these reasons, we know that $LD$ is a normalized, monotone non-decreasing submodular function. ■

## A.2 Proof of Theorem 7.2

In order to simplify the derivations, let us introduce the notation

$$\bar{F}_i(\mathcal{S}_i) = F_i(\mathcal{N}_i \setminus \mathcal{S}_i)$$

so that $G(\mathcal{S})$ can also be written

$$G(\mathcal{S}) \coloneqq \prod_{i=1}^{R} F_i(\mathcal{N}_i) - \prod_{i=1}^{R} \bar{F}_i(\mathcal{S}_i), \tag{A.3}$$

From (A.3) it is clear that $G(\varnothing) = 0$. Thus, proving that $G$ is normalized. To prove monotonicity, recall that the single domain frame potential terms $F_i(\mathcal{L}_i)$ are non-negative, monotone non-decreasing functions for all $\mathcal{L}_i \subseteq \mathcal{N}_i$ [53]. Therefore $\bar{F}_i(\mathcal{S}_i) = F_i(\mathcal{N} \setminus \mathcal{S}_i)$ will be non-negative, but monotone non-increasing. Then, let $\mathcal{S} \subseteq \mathcal{N}$ and $x \in \mathcal{N} \setminus \mathcal{S}$. Without loss of generality, let us assume $x \in \mathcal{N}_i$. We have

$$G(\mathcal{S} \cup \{x\}) = \prod_{i=1} F_i(\mathcal{N}_i) - \bar{F}_i(\mathcal{S}_i \cup \{x\}) \prod_{j \neq i} \bar{F}_j(\mathcal{S}_j),$$
$$G(\mathcal{S}) = \prod_{i=1} F_i(\mathcal{N}_i) - \bar{F}_i(\mathcal{S}_i) \prod_{j \neq i} \bar{F}_j(\mathcal{S}_j).$$

Now, since $\bar{F}_i(\mathcal{S}_i) \geq \bar{F}_i(\mathcal{S}_i \cup \{x\})$, we know that

$$G(\mathcal{S} \cup \{x\}) \geq G(\mathcal{S}).$$

Hence, $G(\mathcal{S})$ is monotone non-decreasing.

To prove submodularity recall that every $F_i(\mathcal{L}_i)$ is supermodular [53]. As taking the complement preserves (super)submodularity, $\bar{F}_i(\mathcal{L}_i) = F_i(\mathcal{N}_i \setminus \mathcal{L}_i)$ is also supermodular.

Let $\mathcal{S} = \bigcup_{i=1}^{R} \mathcal{A}_i$, with $\mathcal{A}_i \subseteq \mathcal{N}_i$ for $i = 1, \ldots, R$, such that $\{\mathcal{A}_i\}_{i=1}^{R}$, forms a partition of $\mathcal{S}$. Now, recall from Definition 5.1 that for $G$ to be submodular we require that $\forall x, y \in \mathcal{N} \setminus \mathcal{S}$

$$G(\mathcal{S} \cup \{x\}) - G(\mathcal{S}) \geq G(\mathcal{S} \cup \{x, y\}) - G(\mathcal{S} \cup \{y\}). \tag{A.4}$$

As the ground set is now partitioned into the union of several ground sets, there are two possible ways the elements $x$ and $y$ can be selected: Either they both belong to the same domain, or they belong to different domains. We prove that (A.4) is satisfied for both cases.

1. If $x, y \in \mathcal{N}_i$, then (A.4) can be developed as

$$\begin{aligned}\bar{F}_i(\mathcal{A}_i)\prod_{j\neq i}\bar{F}_j(\mathcal{A}_j) - \bar{F}_i(\mathcal{A}_i\cup\{x\})\prod_{j\neq i}\bar{F}_j(\mathcal{A}_j)\\ \geq \bar{F}_i(\mathcal{A}_i\cup\{y\})\prod_{j\neq i}\bar{F}_j(\mathcal{A}_j) - \bar{F}_i(\mathcal{A}_i\cup\{i,j\})\prod_{j\neq i}\bar{F}_j(\mathcal{A}_j),\end{aligned}$$

and simplifying

$$\bar{F}_i(\mathcal{A}_i) - \bar{F}_i(\mathcal{A}_i\cup\{x\}) \geq \bar{F}_i(\mathcal{A}_i\cup\{y\}) - \bar{F}_i(\mathcal{A}_i\cup\{x,y\}).$$

Multiplying both sides of the inequality by $-1$ we get

$$\bar{F}_i(\mathcal{A}_i\cup\{x\}) - \bar{F}_i(\mathcal{A}_i) \leq \bar{F}_i(\mathcal{A}_i\cup\{x,y\}) - \bar{F}_i(\mathcal{A}_i\cup\{y\})$$

which is always satisfied since $\bar{F}_i$ is supermodular.

2. If $x \in \mathcal{N}_i$ and $y \in \mathcal{N}_j$ with $i \neq j$, then (A.4) can be developed as

$$\begin{aligned}\prod_{k\neq i,j}\bar{F}_k(\mathcal{A}_k)\left[\bar{F}_i(\mathcal{A}_i)\bar{F}_j(\mathcal{A}_j) - \bar{F}_i(\mathcal{A}_i\cup\{x\})\bar{F}_j(\mathcal{A}_j)\right]\\ \geq \prod_{k\neq i,j}\bar{F}_k(\mathcal{A}_k)\left[\bar{F}_i(\mathcal{A}_i)\bar{F}_j(\mathcal{A}_j\cup\{y\}) - \bar{F}_i(\mathcal{A}_i\cup\{x\})\bar{F}_j(\mathcal{A}_j\cup\{y\})\right].\end{aligned}$$

Extracting the common factors

$$\left[\bar{F}_i(\mathcal{A}_i) - \bar{F}_i(\mathcal{A}_i\cup\{x\})\right]\left[\bar{F}_j(\mathcal{A}_j) - \bar{F}_j(\mathcal{A}_j\cup\{y\})\right] \geq 0. \tag{A.5}$$

Since $\bar{F}_i$ and $\bar{F}_j$ are non-increasing

$$\begin{aligned}\bar{F}_i(\mathcal{A}_i) - \bar{F}_i(\mathcal{A}_i\cup\{x\}) \geq 0\\ \bar{F}_j(\mathcal{A}_j) - \bar{F}_j(\mathcal{A}\cup\{y\}) \geq 0.\end{aligned}$$

Thus, (A.5) is always satisfied, thus proving that (A.4) is satisfied for any $\mathcal{S} \subseteq \mathcal{N}$ and $x, y \in \mathcal{N}\setminus\mathcal{S}$ and therefore $G$ is submodular. ■

## A.3 Proof of Theorem 7.3

From Corollary 7.2 we know that $G(\mathcal{S}_{\text{greedy}}) \geq \frac{1}{2}G(\mathcal{S}^*)$. Therefore,

$$F(\mathcal{N}\setminus\mathcal{S}_{\text{greedy}}) \leq \frac{1}{2}\left[F(\mathcal{N}) + F(\mathcal{N}\setminus\mathcal{S}^*)\right].$$

Introducing the change of variable $\mathcal{L}_{\text{greedy}} = \mathcal{N}\setminus\mathcal{S}_{\text{greedy}}$ and $\mathcal{L}^* = \mathcal{N}\setminus\mathcal{S}^*$, this inequality becomes

$$F(\mathcal{L}_{\text{greedy}})) \leq \frac{1}{2}\left[F(\mathcal{N}) + F(\mathcal{L}^*)\right] = \frac{1}{2}\left(\frac{F(\mathcal{N})}{F(\mathcal{L}^*)} + 1\right)F(\mathcal{L}^*).$$

Now, if we recall that the frame potential of a tight frame [53] is equal to $L_{\text{opt}}^2/K$ with $L_{\text{opt}} = \sum_{i\in\mathcal{L}^*} \|\mathbf{u}_i\|_2^2$, and that this is minimum with respect to the frame potential of any other frame of the same size, we can bound the previous equation as

$$F(\mathcal{L}_{\text{greedy}}) \leq \frac{1}{2}\left(\frac{K}{L_{\text{opt}}^2}F(\mathcal{N}) + 1\right) F(\mathcal{L}^*).$$

Finally, knowing that $L_{\text{opt}} \geq L_{\text{min}} = \min_{\mathcal{L}} \sum_{i\in\mathcal{L}} \|\mathbf{u}_i\|_2^2$ we get

$$F(\mathcal{L}_{\text{greedy}}) \leq \frac{1}{2}\left(\frac{K}{L_{\text{min}}^2}\prod_{i=1}^{R} F_i(\mathcal{N}_i) + 1\right) F(\mathcal{L}^*). \qquad \blacksquare$$

## A.4 Proof of Theorem 8.2

We will divide the proof in two parts. First, we will introduce some new notation, and derive some properties of the involved operations that are useful to simplify the proof. Then, we will use this to derive the proof.

### Preliminaries

First, note that the single-domain Grammian matrices satisfy the following lemma.

**Lemma A.1** (Grammian of disjoint sunion)**.** *Let $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{N}_i$ with $\mathcal{X} \cap \mathcal{Y} = \varnothing$. Then the Grammian of $\mathcal{X} \cup \mathcal{Y}$ satisfies*

$$\mathbf{T}_i(\mathcal{X} \cup \mathcal{Y}) = \mathbf{T}_i(\mathcal{X}) + \mathbf{T}_i(\mathcal{Y}).$$

*Proof.* Let $\mathbf{u}_{i,j}$ denote the $j$th row of $\mathbf{T}_i$. Then

$$\mathbf{T}_i(\mathcal{X} \cup \mathcal{Y}) = \sum_{j\in\mathcal{L}_1\cup\mathcal{L}_2} \|\mathbf{u}_{i,j}\|_2^2 = \sum_{j\in\mathcal{L}_1} \|\mathbf{u}_{i,j}\|_2^2 + \sum_{j\in\mathcal{L}_2} \|\mathbf{u}_{i,j}\|_2^2 . \qquad \blacksquare$$

Let us introduce the complement Grammian matrix

$$\bar{\mathbf{T}}_i(\mathcal{S}_i) \coloneqq \mathbf{T}_i(\mathcal{N}_i \setminus \mathcal{S}_i) = \mathbf{T}_i(\mathcal{N}_i) - \mathbf{T}_i(\mathcal{S}_i), \tag{A.6}$$

which satisfies the following lemma.

**Lemma A.2** (Complement Grammian of disjoint union)**.** *Let $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{N}_i$ with $\mathcal{X} \cap \mathcal{Y} = \varnothing$. Then*

$$\bar{\mathbf{T}}_i(\mathcal{X} \cup \mathcal{Y}) = \bar{\mathbf{T}}_i(\mathcal{X}) - \mathbf{T}_i(\mathcal{Y}).$$

*Proof.* From (A.6) and Lemma A.1 we have

$$\bar{\mathbf{T}}_i(\mathcal{X} \cup \mathcal{Y}) = \mathbf{T}_i(\mathcal{N}_i) - [\mathbf{T}_i(\mathcal{X}) + \mathbf{T}_i(\mathcal{Y})] = \bar{\mathbf{T}}_i(\mathcal{X}) - \mathbf{T}_i(\mathcal{Y}). \qquad \blacksquare$$

Now, let us introduce an operator to compress the writing of the multidomain Hadamard product

$$\mathbb{T}(\mathcal{L}) \coloneqq \mathbf{T}_1(\mathcal{L}_1) \circ \cdots \circ \mathbf{T}_R(\mathcal{L}_R),$$

or alternatively for the complement Grammian

$$\bar{\mathbb{T}}(\mathcal{S}) \coloneqq \bar{\mathbf{T}}_1(\mathcal{S}_1) \circ \cdots \circ \bar{\mathbf{T}}_R(\mathcal{S}_R).$$

Furthermore, we will write the Hadamard multiplication of all $\mathbf{T}_i$ with $i = 1, \ldots, R$, but $j$ as

$$\mathbb{T}_{-j}(\mathcal{L}) \coloneqq \mathbb{T}(\mathcal{L}) \circ \mathbf{T}_j(\mathcal{L}_j)^{\circ -1},$$

where $\mathbf{A}^{\circ n}$ denotes the element-wise $n$th power of $\mathbf{A}$. Analogously for the complement Grammians, we will use $\bar{\mathbb{T}}_{-i}(\mathcal{S})$.

The following theorem guarantees that these matrices are all positive semidefinite.

**Theorem A.1** (Schur product theorem)**.** *The Hadamard product of two positive semidefinite matrices $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{N \times N}$ is always positive semidefinite.*

We will also make use of the following property of the Hadamard product.

**Property A.1.** *Let $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{N \times N}$. Then*

$$\|\mathbf{A} \circ \mathbf{B}\|_F^2 = \operatorname{tr}\left\{\mathbf{A}^{\circ 2}\left(\mathbf{B}^{\circ 2}\right)^T\right\} = \left\langle \mathbf{A}^{\circ 2}, \mathbf{B}^{\circ 2} \right\rangle.$$

Since we will work a lot with the element-wise 2nd powers of the Grammians, let us introduce the notation

$$\mathbf{H}_i(\mathcal{S}) \coloneqq \mathbf{T}_i^{\circ 2}(\mathcal{S}) \qquad \text{and} \qquad \bar{\mathbf{H}}_i(\mathcal{S}) \coloneqq \bar{\mathbf{T}}_i^{\circ 2}(\mathcal{S}), \tag{A.7}$$

which satisfies the following lemma.

**Lemma A.3** ($\mathbf{H}_i$ of disjoint union)**.** *Let $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{N}_i$ with $\mathcal{X} \cap \mathcal{Y} = \varnothing$. Then*

$$\mathbf{H}_i(\mathcal{X} \cup \mathcal{Y}) = \mathbf{T}_i^{\circ 2}(\mathcal{X} \cup \mathcal{Y}) = \left(\mathbf{T}_i(\mathcal{X}) + \mathbf{T}_i(\mathcal{Y})\right)^{\circ 2} = \mathbf{H}_i(\mathcal{X}) + \mathbf{H}_i(\mathcal{Y}) + 2\mathbf{T}_i(\mathcal{X}) \circ \mathbf{T}_i(\mathcal{Y}).$$

*and*

$$\bar{\mathbf{H}}_i(\mathcal{X} \cup \mathcal{Y}) = \bar{\mathbf{T}}_i^{\circ 2}(\mathcal{X} \cup \mathcal{Y}) = \left(\bar{\mathbf{T}}_i(\mathcal{X}) - \mathbf{T}_i(\mathcal{Y})\right)^{\circ 2} = \bar{\mathbf{H}}_i(\mathcal{X}) + \mathbf{H}_i(\mathcal{Y}) - 2\bar{\mathbf{T}}_i(\mathcal{X}) \circ \mathbf{T}_i(\mathcal{Y}).$$

Moreover, as we did with the Grammian matrices we introduce the notation

$$\mathbb{H}(\mathcal{L}) \coloneqq \mathbf{H}_1(\mathcal{L}_1) \circ \cdots \circ \mathbf{H}_R(\mathcal{L}_R),$$

and

$$\mathbb{H}_{-j}(\mathcal{L}) \coloneqq \mathbb{H}(\mathcal{L}) \circ \mathbf{H}_j(\mathcal{L}_j)^{\circ -1},$$

with its analagous $\bar{\mathbb{H}}$, and $\bar{\mathbb{H}}_{-j}$. In virtue of the Schur product theorem, all these are also positive semidefinite.

Finally, note that with the new notation we can simplify the definition of $Q$ to

$$Q(\mathcal{S}) \coloneqq \|\mathbb{T}(\mathcal{N})\|_F^2 - \left\|\bar{\mathbb{T}}(\mathcal{S})\right\|_F^2. \tag{A.8}$$

**Derivation**

Normalization is trivially derived from the fact that $\bar{\mathbf{T}}_i(\varnothing) = \mathbf{T}_i(\mathcal{N})$. To prove monotonicity, let $\mathcal{S} \subseteq \mathcal{N}$ and $x \in \mathcal{N} \setminus \mathcal{S}$. Without loss of generality, assume $x \in \mathcal{N}_i$. We have

$$Q(\mathcal{S} \cup \{x\}) = \|\mathbb{T}(\mathcal{N})\|_F^2 - \left\|\bar{\mathbf{T}}_i(\mathcal{S}_i \cup \{x\}) \circ \bar{\mathbb{T}}_{-i}(\mathcal{S})\right\|_F^2,$$
$$Q(\mathcal{S}) = \|\mathbb{T}(\mathcal{N})\|_F^2 - \left\|\bar{\mathbf{T}}_i(\mathcal{S}_i) \circ \bar{\mathbb{T}}_{-i}(\mathcal{S})\right\|_F^2.$$

Monotonicity requires

$$Q(\mathcal{S}) \leq Q(\mathcal{S} \cup \{x\}),$$
$$-\left\|\bar{\mathbf{T}}_i(\mathcal{S}_i) \circ \bar{\mathbb{T}}_{-i}(\mathcal{S})\right\|_F^2 \leq -\left\|\bar{\mathbf{T}}_i(\mathcal{S}_i \cup \{x\}) \circ \bar{\mathbb{T}}_{-i}(\mathcal{S})\right\|_F^2.$$

Using Property A.1, we get

$$\left\langle \bar{\mathbf{T}}_i(\mathcal{S}_i), \bar{\mathbb{T}}_{-i}(\mathcal{S}) \right\rangle \geq \left\langle \bar{\mathbf{T}}_i(\mathcal{S}_i \cup \{x\}), \bar{\mathbb{T}}_{-i}(\mathcal{S}) \right\rangle.$$

Expanding the unions using Lemma A.2 and due to the linearity of the inner product this becomes

$$0 \leq \left\langle \mathbf{T}_i(\mathcal{S}_i \cup \{x\}), \bar{\mathbb{T}}_{-i}(\mathcal{S}) \right\rangle,$$

which is always satisfied because the inner product between two positive semidefinite matrices is always greater or equal than zero.

To prove submodularity, let again $\mathcal{S} = \bigcup_{i=1}^{R} \mathcal{A}_i$, with $\mathcal{A}_i \subseteq \mathcal{N}_i$ for $i = 1, \ldots, R$ such that $\{\mathcal{A}_i\}_{i=1}^{R}$, forms a partition of $\mathcal{S}$. For $Q$ to be submodular we require that $\forall x, y \in \mathcal{N} \setminus \mathcal{S}$

$$Q(\mathcal{S} \cup \{x\}) - Q(\mathcal{S}) \geq Q(\mathcal{S} \cup \{x, y\}) - Q(\mathcal{S} \cup \{y\}). \tag{A.9}$$

Again, we have two different cases:

1. If $x, y \in \mathcal{N}_i$, then (A.9) can be developed as

$$\left\|\bar{\mathbb{T}}(\mathcal{A})\right\|_F^2 - \left\|\bar{\mathbf{T}}_i(\mathcal{A}_i \cup \{x\}) \circ \bar{\mathbb{T}}_{-i}(\mathcal{A})\right\|_F^2$$
$$\geq \left\|\bar{\mathbf{T}}_i(\mathcal{A}_i \cup \{y\}) \circ \bar{\mathbb{T}}_{-i}(\mathcal{A})\right\|_F^2 - \left\|\bar{\mathbf{T}}_i(\mathcal{A}_i \cup \{x, y\}) \circ \bar{\mathbb{T}}_{-i}(\mathcal{A})\right\|_F^2.$$

Rewriting this expression using Property A.1 we can express the left hand side as

$$\left\langle \bar{\mathbf{H}}_i(\mathcal{A}_i), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle - \left\langle \bar{\mathbf{H}}_i(\mathcal{A}_i \cup \{x\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle,$$

and the right hand side as

$$\left\langle \bar{\mathbf{H}}_i(\mathcal{A}_i \cup \{y\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle - \left\langle \bar{\mathbf{H}}_i(\mathcal{A}_i \cup \{x, y\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle$$

Leveraging the linearity of the inner product we arrive at

$$\langle \bar{\mathbf{H}}_i(\mathcal{A}_i) - \bar{\mathbf{H}}_i(\mathcal{A}_i \cup \{x\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \rangle \geq \left\langle \bar{\mathbf{H}}_i(\mathcal{A}_i \cup \{y\}) - \bar{\mathbf{H}}_i(\mathcal{A}_i \cup \{x, y\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle. \tag{A.10}$$

Developing the matrices using Lemma A.3, we can operate on both sides of this expression giving, for the left hand side

$$\left\langle -\mathbf{H}_i(\{x\}) + 2\bar{\mathbf{T}}_i(\mathcal{A}_i) \circ \mathbf{T}_i(\{x\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle,$$

and for the right hand side

$$\left\langle -\mathbf{H}_i(\{x\}) + 2\bar{\mathbf{T}}_i(\mathcal{A}_i \cup \{y\}) \circ \mathbf{T}_i(\{x\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle.$$

Substituting back in (A.10), we get

$$\left\langle \bar{\mathbf{T}}_i(\mathcal{A}_i) \circ \mathbf{T}_i(\{x\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle \geq \left\langle \bar{\mathbf{T}}_i(\mathcal{A}_i \cup \{y\}) \circ \mathbf{T}_i(\{x\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle$$

Operating using Lemma A.2 we finally arrive at

$$\left\langle \mathbf{T}_i(\{y\}) \circ \mathbf{T}_i(\{x\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle \geq 0, \tag{A.11}$$

which is always satisfied because the inner product of positive semidefinite matrices is always non-negative.

2. If $x \in \mathcal{N}_i$ and $y \in \mathcal{N}_j$ with $i \neq j$, then (A.9) can be rewritten as

$$\left\langle \bar{\mathbf{H}}_i(\mathcal{A}_i) - \bar{\mathbf{H}}_i(\mathcal{A}_i \cup \{x\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle \geq \left\langle \bar{\mathbf{H}}_i(\mathcal{A}_i) - \bar{\mathbf{H}}_i(\mathcal{A}_i \cup \{x\}), \bar{\mathbf{H}}_j(\mathcal{A}_j \cup \{y\}) \circ \bar{\mathbb{H}}_{-(i,j)}(\mathcal{A}) \right\rangle.$$

Using once again Lemma A.3 we can further develop this expression into

$$\begin{aligned} &\left\langle -\mathbf{H}_i(\{x\}) + 2\bar{\mathbf{T}}_i(\mathcal{A}_i) \circ \mathbf{T}_i(\{x\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) \right\rangle \\ &\geq \left\langle -\mathbf{H}_i(\{x\}) + 2\bar{\mathbf{T}}_i(\mathcal{A}_i) \circ \mathbf{T}_i(\{x\}), \bar{\mathbf{H}}_j(\mathcal{A}_j \cup \{y\}) \circ \bar{\mathbb{H}}_{-(i,j)}(\mathcal{A}) \right\rangle. \end{aligned}$$

Leveraging the linearity of the inner product this can be simplified as

$$\left\langle -\mathbf{H}_i(\{x\}) + 2\bar{\mathbf{T}}_i(\mathcal{A}_i) \circ \mathbf{T}_i(\{x\}), \bar{\mathbb{H}}_{-i}(\mathcal{A}) - \bar{\mathbf{H}}_j(\mathcal{A}_j \cup \{y\}) \circ \bar{\mathbb{H}}_{-(i,j)}(\mathcal{A}) \right\rangle \geq 0 \tag{A.12}$$

Here, we can factorize the left entry of the inner product as

$$\begin{aligned} -\mathbf{H}_i(\{x\}) + 2\bar{\mathbf{T}}_i(\mathcal{A}_i) \circ \mathbf{T}_i(\{x\}) &= \mathbf{T}_i(\{x\}) \circ \left[2\bar{\mathbf{T}}_i(\mathcal{A}_i) - \mathbf{T}_i(\{x\})\right] \\ &= \mathbf{T}_i(\{x\}) \circ \left[\bar{\mathbf{T}}_i(\mathcal{A}_i) + \bar{\mathbf{T}}_i(\mathcal{A}_i \cup \{x\})\right] \end{aligned} \tag{A.13}$$

which is positive semidefinite thanks to Schur product theorem and the fact that the set of positive semidefinite matrices is closed under matrix addition.

Similarly, the right entry of the inner product in (A.12) can be factorized as

$$\begin{aligned} \bar{\mathbb{H}}_{-i}(\mathcal{A}) - \left(\bar{\mathbf{H}}_j(\mathcal{A}_j) + \mathbf{H}_j(\{y\}) - 2\bar{\mathbf{T}}_j(\mathcal{A}_j) \circ \mathbf{T}_j(\{y\})\right) \circ \bar{\mathbb{H}}_{-(i,j)}(\mathcal{A}) \\ = \left(-\mathbf{H}_j(\{y\}) + 2\bar{\mathbf{T}}_j(\mathcal{A}_j) \circ \mathbf{T}_j(\{y\})\right) \circ \bar{\mathbb{H}}_{-(i,j)}(\mathcal{A}). \end{aligned}$$

The expression inside the parenthesis is analagous to that in (A.13). Hence, the resulting matrix is then positive semidefinite, and thus (A.12) is always satisfied, proving submodularity of $Q$ for all cases. ■

# Bibliography


[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[2] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph Signal Processing: Overview, Challenges, and Applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[3] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning going beyond euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, 2017.

[4] S. P. Chepuri and G. Leus, "Sparse sensing for statistical inference," *Foundations and Trends in Signal Processing*, vol. 9, no. 3-4, pp. 233–368, 2016.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[7] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *Proc. IAPR Asian Conf. Pattern Recognition*, Nov 2015, pp. 730–734.

[8] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[9] S. Mallat, "Understanding deep convolutional networks," *Philos. Trans. A. Math. Phys. Eng. Sci.*, vol. 374, no. 2065, p. 20150203, 2016.

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

[11] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[12] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud, "A Time-Vertex Signal Processing Framework: Scalable Processing and Meaningful Representations for Time-Series on Graphs," *IEEE Trans. Signal Process.*, vol. 66, no. 33, pp. 817–829, Feb 2018.

[13] G. P. Wagner and P. F. Stadler, "Quasi-Independence, Homology and the Unity of Type: A Topological Theory of Characters," *J. Theor. Biol.*, vol. 220, no. 4, pp. 505 – 527, Feb 2003.

[14] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proc. ACM Int. Conf. Web Search and Data Mining*, 2011, pp. 287–296.

[15] W. Imrich, S. Klavzar, and D. Rall, *Topics in graph theory: Graphs and their Cartesian product*. A K Peters Ltd, 2008.

[16] A. Sandryhaila and J. M. F. Moura, "Big Data Analysis with Signal Processing on Graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80–90, 2014.

[17] E. Isufi, G. Leus, and P. Banelli, "2-Dimensional finite impulse response graph-temporal filters," in *Proc. IEEE Global Conf. Signal Inf. Process.*, pp. 405–409.

[18] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive Moving Average Graph Filtering," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 274–288, Jan 2017.

[19] ——, "Filtering random graph processes over random time-varying graphs," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4406–4421, Aug 2017.

[20] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," Banff, Canada, April 2014.

[21] J. Bruna and X. Li, "Community detection with graph neural networks," *ArXiv e-prints*, vol. 1705, p. arXiv:1705.08415, 2017.

[22] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Advances Neural Inf. Process. Systems*, Montreal, Canada, Dec 2016.

[23] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proc. Advances Neural Inf. Process. Systems*, Montreal, Canada, Dec 2017, pp. 3700–3710.

[24] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," *ArXiv e-prints*, vol. 1612, p. arXiv:1612.07659, 2016.

[25] S. I. Ktena, S. Parisot, E. Ferrante, M. Rajchl, M. Lee, B. Glocker, and D. Rueckert, "Distance metric learning using graph convolutional networks: Application to functional brain networks," *ArXiv e-prints*, vol. 1703, p. arXiv:1703.02161, 2017.

[26] F. Gama, G. Leus, A. Garcia Marques, and A. Ribeiro, "Convolutional neural networks via node-varying graph filters," *ArXiv e-prints*, vol. 1710, p. arXiv:1710.10355, 2017.

[27] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *ArXiv e-prints*, vol. 1609, p. arXiv:1609.02907, 2016.

[28] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv:1706.02263*, 2017.

[29] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, "Geodesic Convolutional Neural Networks on Riemannian Manifolds," in *Proc. IEEE Int Conf. Computer Vision Workshop*, Dec 2015, pp. 832–840.

[30] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Proc. Advances in Neural Inf. Process. Systems*, Montreal, Dec 2016, pp. 3189–3197.

[31] F. Monti, D. Boscaini, J. Masci, E. RodolÃ , J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," *ArXiv e-prints*, 2016.

[32] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep learning on spatio-temporal graphs," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2016, pp. 5308–5317.

[33] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," *arXiv preprint arXiv:1612.07659*, 2016.

[34] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. Int. Conf. Learning Representations*, Vancouver,Canada, Apr. 2018.

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/

[36] K. Naskovska, M. Haardt, and A. L. F. de Almeida, "Generalized tensor contraction with application to khatri-rao coded mimo ofdm systems," in *IEEE Int. Workshop Computational Advances Multi-Sensor Adaptive Process.*, Dec 2017, pp. 1–5.

[37] M. Defferrard, L. Martin, R. Pena, and N. Perraudin, "PyGSP: Graph Signal Processing in Python," Oct. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1003158

[38] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted Graph Cuts without Eigenvectors A Multilevel Approach,," *IEEE Trans. Pattern Anal. Mach. Intell.*

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learning Representations*, Banff, Canada, Apr 2014.

[40] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Trans. Signal Process.*, vol. 64, no. 18, pp. 4845–4860, Sep.

[41] X. Zhu and M. Rabbat, "Approximating signals supported on graphs," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Kyoto, Japan, March 2012, pp. 3921–3924.

[42] A. Anis, A. Gadde, and A. Ortega, "Towards a sampling theorem for signals on arbitrary graphs," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Florence, Italy, May 2014, pp. 3864–3868.

[43] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete Signal Processing on Graphs: Sampling Theory," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6510–6523, Dec 2015.

[44] S. Chepuri, Y. Eldar, and G. Leus, "Graph Sampling With and Without Input Priors," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Calgari, Canada, April 2018, pp. 4564–4568.

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

[46] S. Liu and G. Trenkler, "Hadamard, Khatri-Rao, Kronecker and other matrix products," *Int. J. Inf. Syst. Sci*, vol. 4, no. 1, pp. 160–177, 2008.

[47] A. Cichocki, D. Mandic, L. D. Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor Decompositions for Signal Processing Applications: From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, Mar 2015.

[48] D. Romero, D. D. Ariananda, Z. Tian, and G. Leus, "Compressive covariance sensing: Structure-based compressive sensing beyond sparsity," *IEEE Signal Process. Mag.*, vol. 33, no. 1, pp. 78–93, Jan 2016.

[49] S. P. Chepuri and G. Leus, "Graph Sampling for Covariance Estimation," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 451–466, Sept 2017.

[50] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Math. Program.*, vol. 14, no. 1, pp. 265–294, Dec 1978.

[51] M. Shamaiah, S. Banerjee, and H. Vikalo, "Greedy sensor selection: Leveraging submodularity," in *Proc. 49th IEEE Conf. Decis. Control*, Atlanta, GA, USA, Dec 2010, pp. 2572–2577.

[52] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies," *J. Mach. Learn. Res.*, vol. 9, pp. 235–284, Feb 2008.

[53] J. Rainieri, A. Chebira, and M. Vetterli, "Near-Optimal Sensor Placement for Linear Inverse Problems," *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1135–1146, March 2014.

[54] R. G. Parker and R. L. Rardin, "Polynomial Algorithms–Matroids," in *Discrete Optimization*, ser. Computer Science and Scientific Computing, R. G. Parker and R. L. Rardin, Eds. San Diego: Academic Press, 1988, pp. 57–106.

[55] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions—II," in *Polyhedral combinatorics*. Springer, Dec 1978, pp. 73–87.

[56] R. K. Iyer and J. A. Bilmes, "Submodular optimization with submodular cover and submodular knapsack constraints," in *Proc. Advances in Neural Inf. Process. Systems*, Montreal, Canada, Dec 2013, pp. 2436–2444.

[57] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Oper. Res. Lett.*, vol. 32, no. 1, pp. 41 – 43, Jan 2004.

[58] S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Trans. Signal Process.*, vol. 57, no. 2, pp. 451–462, Feb 2009.

[59] S. P. Chepuri and G. Leus, "Sparsity-promoting sensor selection for non-linear measurement models," *IEEE Trans. Signal Process.*, vol. 63, no. 3, pp. 684–698, Feb 2015.

[60] ——, "Sparse sensing for distributed detection," *IEEE Trans. Signal Process.*, vol. 64, no. 6, pp. 1446–1460, Mar 2016.

[61] C.-T. Yu and P. K. Varshney, "Sampling design for Gaussian detection problems," *IEEE Trans. Signal Process.*, vol. 45, no. 9, pp. 2328–2337, Sep 1997.

[62] S. P. Chepuri and G. Leus, "Sparsity-promoting adaptive sensor selection for non-linear filtering," in *Proc. of IEEE Int. Conf. Acoust., Speech, Signal Process.*, Florence, Italy, May 2014, pp. 5080–5084.

[63] E. Masazade, M. Fardad, and P. K. Varshney, "Sparsity-promoting extended Kalman filtering for target tracking in wireless sensor networks," *IEEE Signal Process. Lett.*, vol. 19, no. 12, pp. 845–848, Dec 2012.

[64] M. F. Duarte and R. G. Baraniuk, "Kronecker Compressive Sensing," *IEEE Trans. Image Process.*, vol. 21, no. 2, pp. 494–504, Feb 2012.

[65] C. F. Caiafa and A. Cichocki, "Computing Sparse Representations of Multidimensional Signals Using Kronecker Bases," *Neural Computation*, vol. 25, no. 1, pp. 186–220, Jan 2013.

[66] C. Caiafa and A. Cichocki, "Multidimensional compressed sensing and their applications," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 6, pp. 355–380, Oct 2013.

[67] Y. Yu, J. Jin, F. Liu, and S. Crozier, "Multidimensional Compressed Sensing MRI Using Tensor Decomposition-Based Sparsifying Transform," *PLoS ONE*, vol. 9, no. 6, p. e98441, Jun 2014.

[68] M. Fickus, D. G. Mixon, and M. J. Poteet, "Frame completions for optimally robust reconstruction," in *Wavelets and Sparsity XIV*, vol. 8138. Int. Society for Optics and Photonics, Sep, p. 81380Q.

[69] N. D. Sidiropoulos and R. Bro, "On the uniqueness of multilinear decomposition of n-way arrays," *J. Chemom.*, vol. 14, no. 3, pp. 229–239, Jun 2000.

[70] M. Lee, H. Shen, J. Z. Huang, and J. S. Marron, "Biclustering via Sparse Singular Value Decomposition," *Biometrics*, vol. 66, no. 4, pp. 1087–1095, 2010.

[71] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, p. 19, Jan 2016.

[72] W. Huang, A. G. Marques, and A. Ribeiro, "Collaborative filtering via graph signal processing," in *Proc. Eur. Signal Process. Conf.*, Kos, Greece, Aug 2017, pp. 1094–1098.

[73] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Matrix completion on graphs," in *Proc. Neural Inf. Process. Systems, Workshop "Out of the Box: Robustness in High Dimension"*, Montreal, Canada, Dec 2014.

[74] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Int. AAAI Conf. Weblogs and Social Media*, San Jose, CA, USA, May 2009.

[75] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Proc. Neural Inf. Process. Systems,*, Montreal, Canada, Dec 2015, pp. 2107–2115.

[76] F. Rusek, D. Persson, B. K. Lau, E. G. Larsson, T. L. Marzetta, O. Edfors, and F. Tufvesson, "Scaling up MIMO: Opportunities and challenges with very large arrays," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 40–60, Jan 2013.

[77] P. Larsson, "Lattice array receiver and sender for spatially orthonormal MIMO communication," in *Proc. IEEE Veh. Technol. Conf.*, vol. 1, Stockholm, Sweden, May 2005, pp. 192–196.

[78] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball, "Deep learning with convolutional neural networks for eeg decoding and visualization," *Hum Brain Mapp*, vol. 38, no. 11, pp. 5391–5420, 2017.

[79] R. Anderson and R. May, *Infectious Diseases of Humans: Dynamics and Control*, ser. Dynamics and Control. OUP Oxford, 1992.

[80] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal Graph-Filter Design and Applications to Distributed Linear Network Operators," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 4117–4131, Aug 2017.

[81] M. Coutino, E. Isufi, and G. Leus, "Distributed edge-variant graph filters," in *2017 IEEE 7th Int. Work. Comput. Adv. Multi-Sensor Adapt. Process.* IEEE, Dec 2017, pp. 1–5.

[82] Y. C. Eldar, *Sampling Theory: Beyond Bandlimited Systems.* Cambridge University Press, 2015.