Graph Structured Data Viewed Through a Fourier Lens

Venkatesan Ekambaram

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-209

http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-209.html

December 16, 2013

Graph-Structured Data Viewed Through a Fourier Lens

by

Venkatesan Nallampatti Ekambaram

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Kannan Ramchandran, Chair
Professor Martin Wainwright
Professor Raja Sengupta

Fall 2013

Graph-Structured Data Viewed Through a Fourier Lens

Abstract

Graph-Structured Data Viewed Through a Fourier Lens

by

Venkatesan Nallampatti Ekambaram

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Kannan Ramchandran, Chair

Graph-structured data appears in many modern applications like social networks, sensor networks, transportation networks and computer graphics. These applications are defined by an underlying graph (e.g. a social graph) with associated nodal attributes (e.g. number of ad-clicks by an individual). A simple model for such data is that of a *graph signal*—a function mapping every node to a scalar real value. Our aim is to develop signal processing tools for analysis of such signals defined over irregular graph-structured domains, analogous to classical Fourier and Wavelet analysis defined for regular structures like discrete-time sequences and two-dimensional grids.

In this work, we start by reviewing the notion of a Graph Fourier Transform (GFT), which has been defined in the literature for graph signals. We examine the spatial and spectral features of circulant graphs, which accommodate linear shift-invariant operations. We describe fundamental operations such as shifting, sampling, graph-reconnection and linear filtering for signals on circulant graphs and derive associated sampling and graph-reconnection theorems. We also develop wavelet filter bank structures for multi resolution analysis of large-scale graphs.

We present a method to decompose an arbitrary graph into a linear combination of circulant graphs. This helps extend fundamental operations such as sampling, filtering and multi resolution filter banks to general graphs. We present an application in the area of graph semi-supervised learning where some of the existing algorithms can be viewed as suitably designed filters defined in the GFT domain. We propose a wavelet regularized learning algorithm and evaluate the performance on some real-world datasets.

To my beloved parents Kamala and Ekambaram, my dearest brothers Kumar and Sridhar, my sweetest sisters Jayashree and Padu, my compassionate sisters-in-law Uma and Lalitha, my dear brothers-in-law Narayanaswamy and Balaji, all my gorgeous nieces and nephews, I hereby dedicate the fruits of my humble toil.

Contents

Acknowledgments

आचार्यात् पादमादत्ते पादं शिष्यः स्वमेधया ।
सब्रह्मचारिभ्यः पादं पादं कालक्रमेण च ॥

AchAryAt pAdamAdatte, pAdam shiShyaH swamedhayA |
sa-brahmachAribhyaH pAdam, pAdam kAlakrameNa cha ||

A student learns a quarter from his teacher, a quarter from his own intelligence
a quarter from fellow students, and the rest in course of time

---Neeti Sastra

All of us are students no matter what stage of life we are in. It is only the *gurus* (teachers) who change— be it your parents, advisors, friends, colleagues or pure experience. There are very few occasions where you get to express your gratitude and thankfulness to all the *gurus* in your life who have equipped you with the necessary tools to help reach your goals. This, I consider as one of those few opportunities to express my sincere gratitude to everyone who has helped me reach this significant milestone in my life.

Parents are the first and foremost *gurus* in ones life. There are no words that can express my thankfulness to my wonderful mother Kamala, and father Ekambaram, whose unconditional love and guidance has helped me progress this far. I would like to profusely thank them for all their support. Equally important are ones siblings, especially elder ones, who take on the roles as friends or parents as the need arises. I have been fortunate to have four amazing elder siblings Kumar, Sridhar, Jayashree and Padmavathy, who have been an immense source of support all through my life and I would like to thank them. This thesis is dedicated to all their years of effort in nurturing me.

I have been very fortunate to have had a PhD advisor like Prof. Kannan Ramchandran. His dynamism and ability to connect multiple topics has always amazed me. He has a great taste for practical problems, yet deeply rooted in theory. He is very motivational and has kept my energy levels high throughout my PhD. I would like to thank him for letting me explore diverse research topics of my interest. Without him, I wouldn't have gotten the opportunity to work with researchers on a wide range of interesting topics.

Next, I would like to thank my collaborator and thesis committee member Prof. Raja Sengupta. It would be hard to judge whether I spent more time discussing with Raja or with Kannan. Raja, again is a highly energized person with diverse interests. It has been due to him that I got introduced to research areas like behavioral sciences whose importance and impact I have begun to appreciate much more than earlier. He has always been supportive of my weird ideas in his research group meetings given my limited knowledge in these areas, though I must apologize for not having made much progress with my ideas. I would like to thank him for his encouragement through out my stay here.

I would next like to thank Prof. Martin Wainwright who is my other thesis committee member and also Prof. Claire Tomlin who was on my Quals committee. The insights provided by Martin during the early stages of my research have been very helpful particularly in the area of graphical models. The discussions I have had with Claire and her students as part of the larger project which was my funding source, has been very useful in connecting applications in control to my work on location estimation that I had worked on in the early years of my PhD.

I would next like to express my sincere gratitude to Prof. P. Vijay Kumar from the Indian Institute of Science, Bangalore, who was my masters advisor. My decision to pursue doctoral studies was largely influenced by him. Without his guidance and encouragement, I would have never been able to pursue my PhD in a wonderful institute like U C Berkeley.

Outside my thesis committee, there have been many others without whose collaboration my progress would have been limited. My main collaborators for the work I have presented in this thesis are Dr. Babak Ayazifar and Giulia Fanti. I have known Babak both as a lecturer for whose course I had been a teaching assistant as well as a great collaborator. I don't think I need to say much about his teaching capabilities—any undergrad who has taken his course can vouch for his class as being the best they would have probably taken in electrical engineering. He has an amazing style of teaching and keeps up the energy levels in the class. As a collaborator, he is one of the few faculty who actually sit with you and derive the nitty gritty details of proofs. I remember that for every paper deadline he used to sit late at night editing the paper while I would probably be dozing off even though I was on the west coast time and he on the east coast! Thanks for all your help Babak.

Giulia has been a wonderful collaborator for both my research work on location estimation as well as the work presented here. There hasn't been a time where I have asked her for some help and she has refused irrespective of whether she is busy with exams or working on her own deadlines. I am very fortunate to have had her as a collaborator.

I would also like to thank my other collaborators who have exposed me to a very diverse range of research topics yet intrinsically connected in terms of the underlying tools being used. These include Dr. Gerald Friedland, Jaeyoung Choi, Dr. Christian Manasseh, Adam Goodliss, Andre Carrel, Dr Jerry Jariyasunant, Prof. Joan Walker and her students and Sameer Pawar. In particular, I would like to thank Sameer for the wonderful discussions we have had, both technical as well as non-technical ones, especially during our lunch breaks. These were probably much more important in shaping our career paths and widening our knowledge than any of the other research discussions we might have had!

As the slogan at the beginning of this section says, a quarter of what you learn is from your fellow colleagues. I think the Wireless Foundations lab environment is an excellent example of this. I have been fortunate enough to have had some of the best brains around the globe as my colleagues. I would like to extend my sincere gratitude to all of them for having made my stay a memorable one. These include Mark, Hao, Nihar, Rashmi, Nima, Nebojsa, Po Ling, Varun, Chang Ho, Sahand, KK, Gireeja, Naveen, Kate, Kris, Pulkit, Vasuki, Kangwook, Vijay, Stephan, Sudeep, Se Young, Salim, Sreeram and other new students and postdocs. I would also like to thank the various faculty of WiFo some of whose courses have laid the foundations for the tools I have been using in my research. Thanks are also due to wonderful staff like Kim Kail and Lea Barker who have always been there when we need them.

Last but not the least, I would like to thank my present and past roommates, friends from SPIC-MACAY and all other friends who have made my stay at Berkeley a very memorable one.

Part I

Background

Chapter 1

Introduction

Graph-structured data is present in numerous modern applications, such as social media services (e.g. Facebook and Twitter [1]), wireless sensors (e.g. temperature measurements [2]), power networks [3], computer graphics [4, 5], and finite-element meshes [6]. Fig 1.1 gives some examples of such graphs, which can have upwards of millions of nodes in practice. Most of these graphs have attributes associated with the nodes or edges. For example, sensor nodes have measurement values associated with each node, and social media graphics have attributes like the name, age, gender, or number of ad clicks associated with each node in the graph. Given such data, problems of interest include finding patterns, predicting unobserved data, or obtaining multiscale representations of the graph and the associated data for efficient processing.

The machine learning (ML) community has spent considerable effort developing techniques to understand and process large-scale, graph-structured data. Many efficient prediction and classification algorithms [7] have been developed for different applications of interest and these techniques have had considerable impact in practice. However, ML algorithms suffer from certain drawbacks. In particular, many of these algorithms are tailored for specific applications and there is no clear understanding of when one should choose a particular approach over another. Further, each algorithm has many tunable parameters and the choice of these parameters is more of an art than design which heavily depends on the dataset of interest. Thus, there is lack of a unified theory for processing graph-structured data encompassing various applications that provides optimal design constructs.

The signal processing (SP) community on the other hand, has been largely successful in developing a concrete theory for well-ordered structures such as time-lines and Cartesian grids for the analysis of time-signals, images and videos. These regular structures in fact have a graph-theoretic viewpoint and can be interpreted as functions defined over path graphs and grid graphs (see Chapter 2). Fourier and Wavelet analysis tools have had considerable impact in many applications dealing with time signals and images. However these existing signal processing tools do not easily carry over

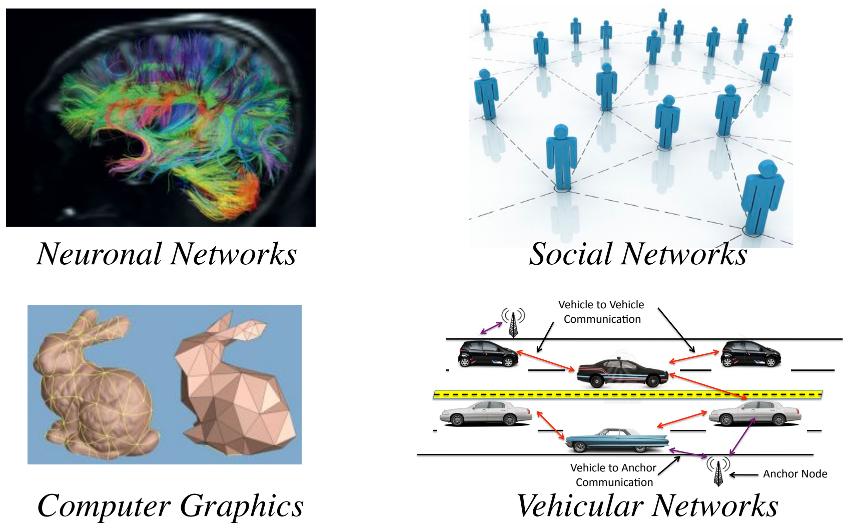


Figure 1.1: Example of some real world graph structured data. The values on the nodes could be scalar values (e.g. Mesh Networks), time varying (e.g. Neuronal Networks, Vehicular Networks) or abstract (e.g. Social Networks).

to arbitrary graphs and a substantially more nuanced paradigm is required to deal with general graph-structured data.

Consequently, there has been a recent surge of interest in the SP community to develop a unified theory for the analysis and processing of "graph signals," which model data defined over graphs. In the most simplest form, a graph signal (defined more formally in Chapter 2) is essentially a function that maps each node in the graph to a scalar real value. The values associated with the nodes are representative of the nodal attributes in different applications. Given the notion of a graph signal, it is thus natural to ask whether one can extend Fourier and Wavelet analytical tools defined in classical signal processing to the more abstract, irregular domains of graphs and manifolds. Such an exposition is of significant intellectual interest in extending classical DSP beyond time-lines and grids. Further, it can also foster a better understanding of various ML algorithms under this perspective and thereby lead to improved algorithms that can be provably optimal.

In order to build a theory for the analysis of graph signals, it is important to ask questions of the following form: What functions defined on graphs can serve as a Fourier-like bases? Which graph signals are considered to have high or low frequency content? What interesting properties, if any, does a graph Fourier decomposition possess with respect to signal analysis or filter design? Is there a fundamental tradeoff, such as an uncertainty principle, governing graph signals? Can we design wavelet-like basis functions that have localized support on the graph domain as well as the spectral domain?

There has been much recent interest in the SP community in addressing some of the above questions. A survey of this extensive body of literature is provided in Chapter 10. However, much work remains in the quest for a unified, let alone comprehensive, signal processing theory for graphs. Under this context, this thesis is a modest attempt in exploring some of the nuances of signal processing concepts when applied to graphs.

A major challenge in defining even fundamental operations like signal shifting is the irregular structure of general graphs; graphs typically look different from different nodes. Hence, it is unclear how to "shift" or "sample" a signal on the graph. Authors have either resorted to unconventional definitions for these operations [8] or defined them in domains different from that of the signal [9]. Each approach has its own philosophy and comes with its own advantages and disadvantages. Our approach is motivated by linear time-invariant (LTI) signal processing, which has had considerable impact in classical signal processing. We view general graphs as *linear shift-varying (LSV)* systems analogous to linear time-varying (LTV) systems in the classical theory. A corpus of research literature deals with LTV systems through the lens of LTI systems [10–14]. Accordingly, we first study *circulant* graphs—a class of graphs amenable to *linear shift-invariant (LSI)* operations and then extend the developed tools to general graphs through a circulant decomposition of these graphs.

Our main contributions can be summarized as follows:

- We explore fundamental signal processing operations on circulant graphs to a substantive depth. In particular, we analyze the properties of the Graph Fourier Transform (GFT) as defined in the literature for circulant graphs. We define basic operations like shifting, sampling and graph reconnection strategies for circulant graphs and analyze the corresponding properties in the spectral domain. Fundamental sampling theorems and uncertainty principles are derived. These form the basis for filter design and multi-resolution analysis.

- We design three classes of two-channel filter bank structures for signals on circulant graphs. These are shown to satisfy different desirable properties of multi resolution filter banks. Further they provide wavelet bases at different levels which in turn result in a multiscale representation of the given graphical data.

- For analyzing signals on general graphs, we provide a decomposition of an arbitrary graph into circulant graphs. In particular, we show that the adjacency matrix of a given graph can be written as a suitable linear combination of the adjacency matrices of individual circulant graphs. Fundamental operations such as sampling and filter design are extended to general graphs through this decomposition.

- Two-channel filter bank structures are designed for general graphs that help obtain a multiscale representation of any given data on a general graph. These filter bank structures are shown to satisfy desirable properties like critical-sampling and perfect-reconstruction.

- We discuss a specific application related to a class of semi-supervised ML algorithms, known as graph semi supervised learning (GSSL) algorithms. Some of the existing GSSL algorithms can be viewed as appropriate filter designs in the graph domain. A wavelet based semi-supervised algorithm is proposed and the performance is evaluated on different datasets in comparison to existing techniques.

This thesis is organized as follows. Chapter 2 formally introduces the concept of graph signals and the graph Fourier transform. Chapter 3 introduces circulant graphs and associated fundamental signal processing operations for signals defined on these graphs. In Chapter 4 we discuss the important operation of sampling for circulant graph signals and derive sampling and aliasing theorems. Further, we also discuss reconnection methodologies to define the downsampled graph after sampling. Chapter 5 discusses multi resolution filter bank design for signals on circulant graphs and their associated properties. In Chapter 6, we introduce general graphs and provide methods to decompose a general graphs into circulant graphs. This decomposition is exploited to extend operations like sampling from circulant graphs to general graphs in Chapter 7. Chapter 8 discusses design of multi resolution filter bank structures for signals on general graphs. In Chapter 9, we provide an example application in the area of semi-supervised learning. Chapter 10 provides a literature survey of different results in this domain. We conclude with a discussion of open research problems in Chapter 11. A table of notations is also provided at the end of the thesis.

Chapter 2

Discrete Signals on Graphs

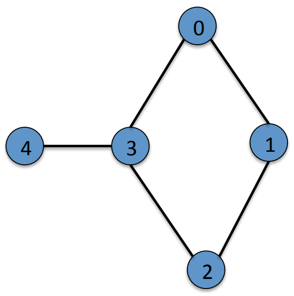
2.1 Graphs

Graphs are mathematical structures that represent a set of objects which are related to each other. The objects are represented by vertices or nodes and the relations are encoded by edges that interconnect the vertices. The objects and the interconnections vary depending on the application of interest. For example, in social network graphs, the objects correspond to different users and links are present between users if they share a social or a professional relationship.

Formally, a graph G of size N, is an ordered pair (V, E), where V is the set of N vertices of the graph indexed 0 to $N-1$ and E is the set of all edges in the graph. In general, the edges could be either directed or undirected. In many applications, it suffices to deal with undirected graphs. However, there are certain applications that inherently require a directed graph representation. For example, internet web graphs that have vertices representing webpages and edges representing hyperlinks from one webpage to another, are directed in nature. For the rest of this thesis, we will only consider *undirected simple* graphs that have no-self loops on any vertex or multiple edges between a pair of vertices.

Further, one could also have weights associated with each of the edges that is a measure of the strength of the relation between the corresponding nodes. For example, in sensor network graphs, the weights are usually a function of the distance between the sensor nodes which reflects the correlation between the sensor readings at those nodes. We restrict ourselves to unweighted graphs (i.e. all the weights are unity) while discussing different operations and theoretical results. However, many of the results apply to weighted graphs as well unless stated otherwise.

There are two matrices of interest that can be associated with a graph—the adjacency matrix $\mathbf{A}$ and the Laplacian matrix $\mathbf{L}$. The adjacency matrix of an unweighted graph is a $N \times N$ matrix



$$A=\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array} \qquad L=\begin{bmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Figure 2.1: Example of an unweighted graph and the associated adjacency and Laplacian matrices. Note that the matrices are defined for a fixed node ordering.

defined as follows,

$$A(i,j) = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

For a general weighted graph, $A(i,j)$ can be any non-negative real number whenever there is an edge between the two nodes. Let $\mathbf{D}$ be a diagonal matrix, where the diagonal entry $D(i,i)$, is the degree of node i for an unweighted graph, and all other entries are zero. For a weighted graph, it is the sum of the weights of the edges incident to node i. The non-normalized Laplacian matrix of a graph is defined as follows,

$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{2.2}$$

The Laplacian matrix characterizes many properties of a graph, and it plays a significant role in our framework.

One can also defined normalized versions of the adjacency and Laplacian matrices. There are two normalized versions of the adjacency (Laplacian) matrix that are typically used in the literature—the symmetric normalized adjacency (Laplacian) matrix, $\mathbf{A}^{\mathsf{S}}(\mathbf{L}^{\mathsf{S}})$ and the random walk normalized adjacency (Laplacian) matrix $\mathbf{A}^{\mathsf{RW}}(\mathbf{L}^{\mathsf{RW}})$.These are defined as follows,

$$\mathbf{A}^{\mathsf{S}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \tag{2.3}$$
$$\mathbf{A}^{\mathsf{RW}} = \mathbf{D}^{-1}\mathbf{A}, \tag{2.4}$$
$$\mathbf{L}^{\mathsf{S}} = \mathbf{I}_N - \mathbf{A}^{\mathsf{S}}, \tag{2.5}$$
$$\mathbf{L}^{\mathsf{RW}} = \mathbf{I}_N - \mathbf{A}^{\mathsf{RW}}, \tag{2.6}$$

where $\mathbf{I}_N$ is the identity matrix of size N. For the rest of the thesis, we will only work with the non-normalized adjacency and Laplacian matrices unless stated otherwise.

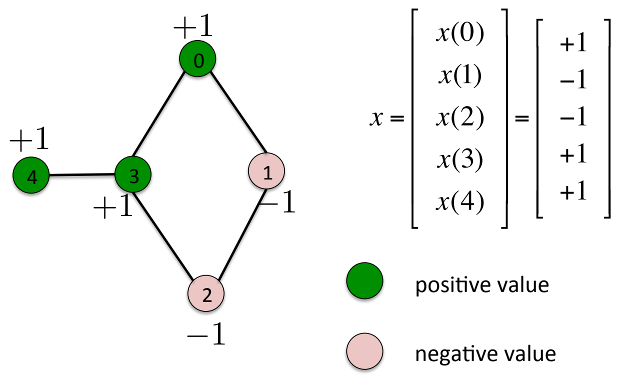

+1
0
+1
4
3
+1
1
−1
2
−1
$x = \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \end{bmatrix} = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$
positive value
negative value

Figure 2.2: Example of a graph signal on a graph of five nodes. The graph signal is a function that maps every vertex in the graph to a real-value.

Fig 2.1 shows an example of an unweighted graph and its associated adjacency and Laplacian matrices. The matrices are defined for a fixed node ordering and would correspondingly change if the nodes are renumbered.

The Laplacian matrix has many properties of interest. We state a few of them below without proof and refer to reader to standard references like [15] for an in-depth treatment.

Proposition 1. *The Laplacian matrix* $\mathbf{L}$ *of an undirected graph is a symmetric positive semi-definite matrix.*

Proposition 2. $\mathbf{L}$ *has the eigenvalue zero corresponding to the all-ones eigenvector* $\mathbf{1}$.

Proposition 3. *The multiplicity of the eigenvalue zero of the Laplacian matrix* $\mathbf{L}$*, is the number of connected components in the graph* G.

These properties will be useful in our theoretical analysis as we shall see in the later chapters.

2.2 Graph Signals

Many applications that present graph-structured data, have attributes associated with the nodes or edges in addition to the underlying graph itself. A simple mathematical model for such nodal attributes is that of a *graph signal*.

Definition 1. *A graph signal, $\boldsymbol{x} = [x(0)x(1)\cdots x(N-1)]^T$, is a scalar real-valued function defined on the vertices of the graph, i.e.,*

$$x \; : \; V \to \mathbb{R}, \tag{2.7}$$

$$v \to x(v). \tag{2.8}$$

Fig 2.2 shows an example of a signal defined on a graph with five nodes. The signal is binary in this example, with each node associated with either positive unity or negative unity.

Note that one could extend the definition of a graph signal to vector valued functions on nodes or time-series on each node. However, we shall restrict ourselves to the simplest case of a real-valued scalar function. Further, we will only consider graphs with a finite number of nodes.

The notion of a graph signal encompasses the signal definitions that we encounter in classical discrete signal processing. In particular, aperiodic time-signals can be viewed as graph-signals defined on a path graph (see Fig 2.3 (a)) and periodic time-signals correspond to graph signals on a ring graph (see Fig 2.3 (b)). Further, images can be viewed as graph-signals on a two-dimensional grid graph (Fig 2.3 (c)), and video signals can be viewed as graph signals on a multi-dimensional grid graph (Fig 2.3 (d)).

Given the definition of a graph signal, a standard question that often arises is, "What does an edge between two nodes signify?". To answer this question, it helps to start with classical signals such as time signals and images. Signal values on adjacent time instants or adjacent pixel values usually tend to be highly correlated either positively as in the case of low-pass signals or negatively as in the case of high-frequency signals. Of course, for completely random signals one cannot predict how adjacent signal values change. In such a case, it may not even matter as to how we order the signal values and therefore disregard the time-axis for time signals or the space-axis for image signals.

A similar notion could be extended for general graphs, wherein the graph is provided by the application. Consider for example social network graphs, where the signal on each node is defined as the number of ad-clicks by that user. If the number of ad-clicks of a particular user is high, it is likely that the friends of that user also have high ad-clicks since they share common interests. Hence the graph signal value on a particular node in this example, is highly correlated with its neighboring nodes than compared to other nodes in the graph. Thus the notion of adjacency on a graph reflects certain correlation properties of typical signals defined on the graph.

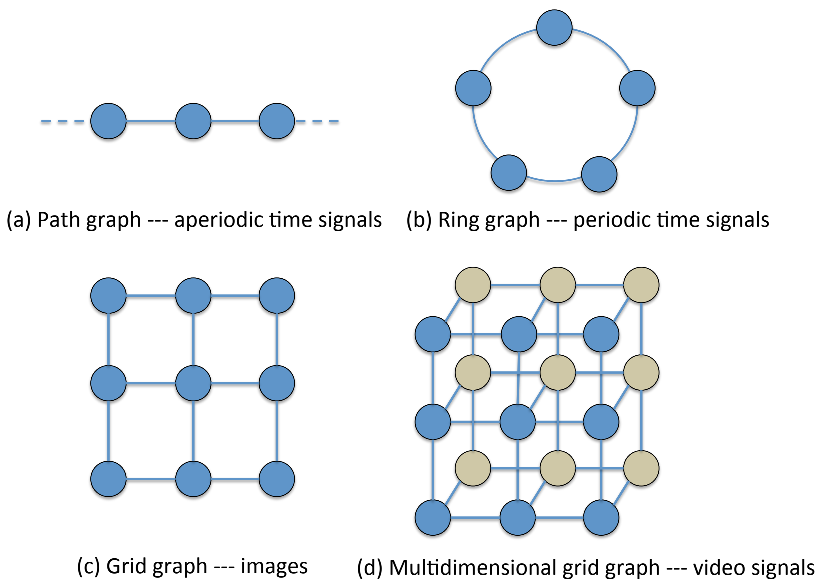

(a) Path graph --- aperiodic time signals
(b) Ring graph --- periodic time signals
(c) Grid graph --- images
(d) Multidimensional grid graph --- video signals

Figure 2.3: Examples of signals in classical signal processing that can be viewed as graph signals. (a) The path graph corresponds to aperiodic time signals. (b) The cycle graph corresponds to periodic time-signals. (c) The grid graph corresponds to images with each pixel represented by a node and edges are between adjacent pixels in the image. The signal value is the intensity of the corresponding pixel. (d) The multidimensional grid graph corresponds to video-signals.

2.3 The Graph Fourier Transform

In the previous section, we introduced the notion of graph signals. Given different applications which present data that can be modeled as graph signals, we are now interested in developing tools to analyze such signals. One of the most important tools for analysis in classical signal processing has been the Discrete Fourier Transform (DFT). The DFT is essentially an alternative basis representation of signals in the time-domain. The set of basis vectors into which we decompose the given signal is the Fourier basis, which has many interesting properties that can be exploited for signal analysis. Much of existing signal processing techniques for both time and image signals relies on the DFT representation of these signals. This motivates one to derive a set of basis vectors similar to the Fourier basis for graph signals.

Clearly any set of vectors that span $\mathbb{R}^N$ would be a valid basis for graph signals. However we are not interested in any arbitrary basis. We want something that is analogous to the Fourier basis. In particular, it would be desirable to have basis vectors that capture notions of "high" and "low"-frequencies on graphs similar to that of sinusoids in time domain. A low-frequency graph signal would be one that varies very slowly with respect to its neighbors and a high-frequency signal would be one that varies significantly with respect to the neighboring nodes (see Fig 2.4 for some examples). Further, the basis vectors should be invariant to the node-ordering.

There has been quite some work in the literature pursuing an ideal Fourier-like basis for signals defined on graphs, focusing particularly on the properties of Laplacian matrix eigenvectors. Interestingly, these eigenvectors are analogous to sinusoids in the time domain in that they have a natural signal-frequency interpretation. The Laplacian matrix $\mathbf{L}$ of an undirected graph is a symmetric positive semi-definite matrix. The spectral decomposition theorem guarantees the existence of an orthonormal matrix $\mathbf{U}$ that diagonalizes $\mathbf{L}$, i.e.,

$$\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^{\mathsf{H}}, \tag{2.9}$$

where Λ is a diagonal matrix of non-negative real eigenvalues. The columns of $\mathbf{U}$ which are the eigenvectors $\{\boldsymbol{u}_0, \boldsymbol{u}_2, ..., \boldsymbol{u}_{N-1}\}$ corresponding to the ordered eigenvalues $\{0 \leq \lambda_0 \leq \lambda_1 ... \leq \lambda_{N-1}\}$, constitute an orthonormal basis for $\mathbb{R}^N$.

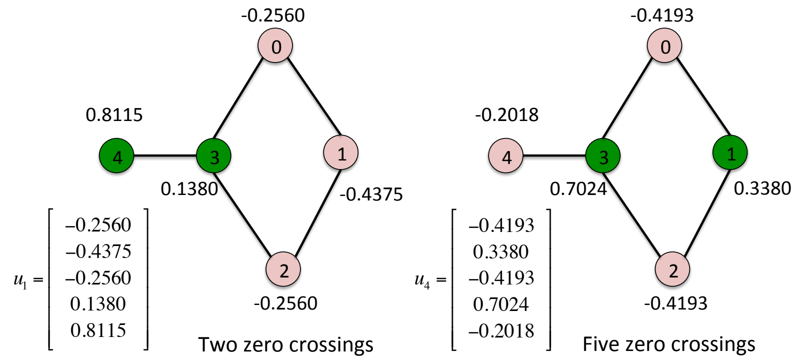
Given a node ordering, each element of an eigenvector can be associated with a corresponding node in the graph (see Fig 2.4). If the node ordering changes, then the ordering of the elements in the eigenvector would also correspondingly change. Hence the mapping between the values of the eigenvector to the nodes is permutation invariant. The number of pairs of adjacent nodes with a sign change represents the frequency of that eigenvector on the graph. Interestingly, eigenvectors with larger eigenvalues have more sign changes than those with smaller eigenvalues. For example, the eigenvector corresponding to eigenvalue zero is constant; it does not change its value across

Eigenvalues

$$\Lambda = \begin{bmatrix} 0.0000 \\ 0.8299 \\ 2.0000 \\ 2.6889 \\ 4.4812 \end{bmatrix}$$

Eigenvectors

$$U = \begin{bmatrix} -0.4472 & -0.2560 & 0.7071 & 0.2422 & -0.4193 \\ -0.4472 & -0.4375 & 0.0000 & -0.7031 & 0.3380 \\ -0.4472 & -0.2560 & -0.7071 & 0.2422 & -0.4193 \\ -0.4472 & 0.1380 & 0.0000 & 0.5362 & 0.7024 \\ -0.4472 & 0.8115 & 0.0000 & -0.3175 & -0.2018 \end{bmatrix}$$



$$u_1 = \begin{bmatrix} -0.2560 \\ -0.4375 \\ -0.2560 \\ 0.1380 \\ 0.8115 \end{bmatrix} \qquad u_4 = \begin{bmatrix} -0.4193 \\ 0.3380 \\ -0.4193 \\ 0.7024 \\ -0.2018 \end{bmatrix}$$

Figure 2.4: Example illustrating the frequency interpretation of the eigenvectors of the Laplacian matrix. Given a node-ordering, every element of each eigenvector can be mapped to the corresponding node in the graph. When one considers the number of sign transitions on every edge in the graph for different eigenvectors, then one finds that the eigenvectors corresponding to a lower eigenvalue has lesser number of transitions as compared to that of a eigenvector corresponding to a higher eigenvalue. In this example, the eigenvector $\boldsymbol{u}_1$ has two-zero crossings as compared to $\boldsymbol{u}_4$ that has five zero-crossings. One can similarly verify this for the other eigenvectors.

nodes and hence is like a DC signal on a graph. Fig 2.4 shows an example of the eigenvectors of the Laplacian of a graph. The eigenvector $\boldsymbol{u}_1$ corresponding to λ_1 has fewer zero-crossings than $\boldsymbol{u}_4$, which corresponds to the higher eigenvalue λ_4.

These ideas have been formalized by nodal domain theorems in the literature [16], which justify the high-frequency and low-frequency interpretations of the eigenvectors on a graph. Further, the Laplacian matrix provides a stencil approximation for the double-differential operator in continuous time. The eigenfunctions of the continuous time double differential operator are the complex exponentials that form the Fourier basis. Finally note that the basis of choice, i.e. the Laplacian matrix eigenvectors, are only dependent on the graph and not the signal on the graph. These properties motivate the following definition of the Graph Fourier Transform (GFT).

Definition 2. *The GFT $\mathbf{X}^G$, of a graph signal $\boldsymbol{x}$, is the decomposition of the graph signal $\boldsymbol{x}$ with respect to the orthonormal eigenvector basis $\mathbf{U}$:*

$$X^G(k) = \langle \boldsymbol{u}_k, \boldsymbol{x} \rangle, \tag{2.10}$$

$$\mathbf{X}^G = \mathbf{U}^H \boldsymbol{x}, \tag{2.11}$$

where $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^H \boldsymbol{b}$ is the inner product between the two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

For the cycle graph, that corresponds to periodic time-signals, the Laplacian matrix is a circulant matrix. Hence the eigenvectors are the columns of the DFT matrix. Further, the eigenvalues are a monotonically increasing function of the Fourier frequency and hence the GFT corresponds to the DFT conforming to our intuition. Similarly for a grid graph, we obtain sinusoidal functions for the eigenvectors. Fig 2.5 shows examples of some of the eigenvectors corresponding to a path graph of 16 nodes. One can observe the similarities between the eigenvectors of the path graph to sinusoids in the time-domain.

Mathematically, the above analogies and results in the literature justify to a certain extent the treatment of the Laplacian eigenvectors as a Fourier-like basis for graph signals. However, one still wonders as to why the Laplacian matrix has such an interesting property? Following is a heuristic explanation of this behavior. One can think of the Laplacian matrix as a linear operator for signals defined on the corresponding graph. In particular, for a graph signal $\boldsymbol{x}$, one can analyze the operation $\mathbf{L}\boldsymbol{x}$. The output of this operation essentially retains weighted differences of signal values on each node with respect to their neighboring nodes. Thus, the operator $\mathbf{L}$ can be viewed as a "high"-pass operator for signals defined on the graph. Hence we would expect that the eigenvectors corresponding to higher eigenvalues would also reflect this property i.e. behave like

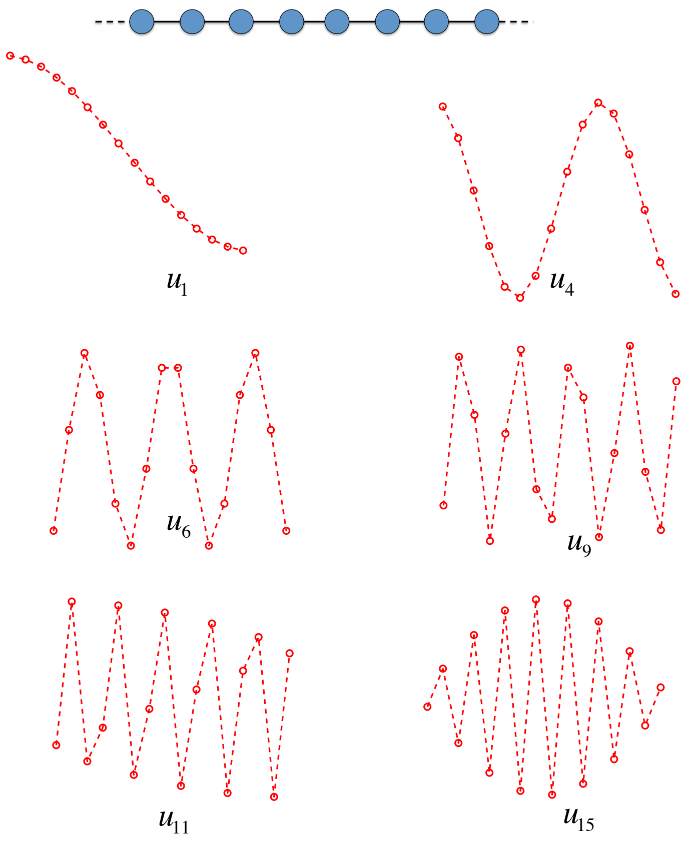


Figure 2.5: Some of the eigenvectors are shown for a path graph with 16 nodes. One can observe that the frequency of the eigenvectors increase as we move to higher eigenvalues. Further, they strongly resemble the classical sinusoids in the time domain.

"high"-frequency signals as opposed to the eigenvectors corresponding to the lower eigenvalues.

For the rest of this work, we will use the above definition for the GFT. However, there are certain caveats of the treatment of the eigenvectors of the Laplacian as a Fourier basis that one should keep in mind which we detail below.

1. The nodal domain theorems only assert that the number of zero-crossings do not decrease with increasing eigenvalues. It may be that the number of zero-crossings remain the same for many eigenvectors of a given graph. Further it is unclear if zero-crossings are to be the only metric for choice of a frequency representative basis.

2. For Laplacian matrices that have repeated eigenvalues, one needs to treat the eigenvector subspace of the repeated eigenvalue as corresponding to a certain frequency. In this case, the eigenvectors are also not unique and it is unclear as to which eigenvectors should one choose. As an example, for ring graphs corresponding to periodic time-signals, we know that the DFT is a good basis to choose in spite of repeated eigenvalues. Further we also know the right ordering of the basis. However, it is unclear for general graphs.

3. The eigenvalues help us in ordering the eigenvectors. However, it is unclear as to what the actual values represent in our context. Do closer eigenvalues represent relatively "closer" frequencies? For the cycle graph, we get eigenvalues that are non-uniformly spaced and they have a one-to-one mapping with the discrete Fourier frequencies. In our work, we will disregard the eigenvalues, other than using them for ordering the eigenvectors. However, we will point out certain interesting observations as we proceed with our theory and that in the literature.

4. The frequency interpretations of the eigenvectors are not shown to carry over to directed graphs.

Each of the above questions are important and interesting research problems in their own right. There is some research work in the literature actively trying to address a subset of these questions. In this thesis, we will restrict ourselves to the current definition of the GFT and proceed to analyze its properties and propose filter designs based on this definition.

2.4 Chapter highlights

Following is a summary of the main points of this chapter.

- Graph signals are a mathematical model for graph-structured data wherein we have an underlying graph with nodal attributes. Mathematically, a graph signal is a function that maps every node in a graph to a scalar real value.
- The goal is to develop tools analogous to Fourier analysis to process such data.
- The eigenvectors of the Laplacian matrix satisfy many properties similar to sinusoids in classical signal processing, thereby motivating the definition of a Graph Fourier Transform (GFT).
- The GFT is defined as projecting the graph signal onto the eigenvectors of the Laplacian matrix. The GFT reduces to the DFT for cycle graphs corresponding to time-signals.

Part II

Signal Processing on Circulant Graphs

Chapter 3

Fundamental signal processing operations on circulant graphs

3.1 Motivation

In the previous chapter, we defined the basic notion of a graph signal and the associated Graph Fourier Transform (GFT). Given this, the problem of interest is to develop tools like filter banks and wavelets for signal analysis. Gaining insights from the well developed classical signal processing theory, we shall extend some of the existing tools for analysis of graph signals. Accordingly, before we proceed to filter design, we first need to define fundamental operations like shifting, sampling, correlation and convolution for graph signals which are essential components of filter design.

The first and most important signal operation that needs to be defined is shifting. In fact, this is the first definition (after the definition of signals) that one encounters in any book on classical discrete signal processing (e.g. see Oppenheim's book [17] Definition 2.3). Once there is notion of a shift, it is intuitive to visualize linear filtering as the linear combination of a signal value on a node and its multi-hop neighbors, which is then repeated for every other node by "shifting" the filter coefficients. Thus linear filters are classically represented as polynomials in the shift-operator. Similarly, a downsampling operation can also be interpreted as retaining signal values on every alternative shift on a graph. The definition of a shift-operator therefore fundamentally distinguishes different theoretical approaches for graph signal processing.

Classical signal processing has significantly benefitted from the theory of Linear Time-Invariant (LTI) signal processing. Analogously, it would be valuable to develop a Linear Shift-Invariant (LSI) theory for graph signal processing in order to reap similar benefits as LTI theory in the classical domain. Intuitively, LSI filters would be defined by a fixed set of filter coefficients that remain the same for every node and its neighbors on the graph. A weighted average of the neighbors with weights defined by the filter coefficients is computed. The filter coefficients are then shifted to the subsequent node and the operation is repeated. For such an operation to be defined, it is necessary that the neighborhood structure of every node in the graph be similar or in other words the graph needs to exhibit certain symmetry properties. For a general graph, this is not possible since the number of neighbors of each node could vary.

This motivates us to analyze a special class of graphs known as *circulant graphs* which are amenable to LSI operations. In the sections to follow, we will define fundamental signal processing operations on these classes of graphs and provide filter bank designs. We treat general graphs as only allowing for linear shift-varying (LSV) operations similar to linear time-varying (LTV) signal processing. We decompose a general graph into a bank of circulant graphs and then extend the different signal processing operations.

3.2 Circulant graphs

3.2.1 Definition

In order to introduce notions of shift-invariant processing on graphs, we need graphs that "look" the same from any node. There exists such a class of so-called *symmetric* graphs [18], an important subset of which is known as *circulant* graphs.

Definition 3. *A graph G is circulant if there exists some ordering of nodes for which the adjacency matrix of the graph is circulant. An alternative definition of a circulant graph arises from the way it is constructed. A graph G is circulant with a set generating $S = \{s_1, s_2, ..., s_M\}, 0 < s_k \leq N-1$ whenever there is an edge between nodes $(i, (i+s_k)_N), \forall s_k \in S$, where $()_N$ represents the* $\mathrm{mod}\ N$ *operation.*

Fig. 3.1 shows examples of some circulant graphs. The generating set S determines the structure of the graph. In particular, when all the elements of S is odd, it is easy to see that the graph is bipartite. Note that the Laplacian matrix of a circulant graph is a circulant matrix as well.

Given a graph, it is not straightforward to determine whether it is circulant or not. A graph is circulant if its adjacency matrix is symmetric and circulant, but this structure depends heavily on the graph's node ordering. There exist polynomial time algorithms [19] to determine whether a given graph is circulant under certain constraints (e.g. prime number of nodes). However, in the following analysis, we assume that we are given an ordering of nodes for which the adjacency matrix is circulant.

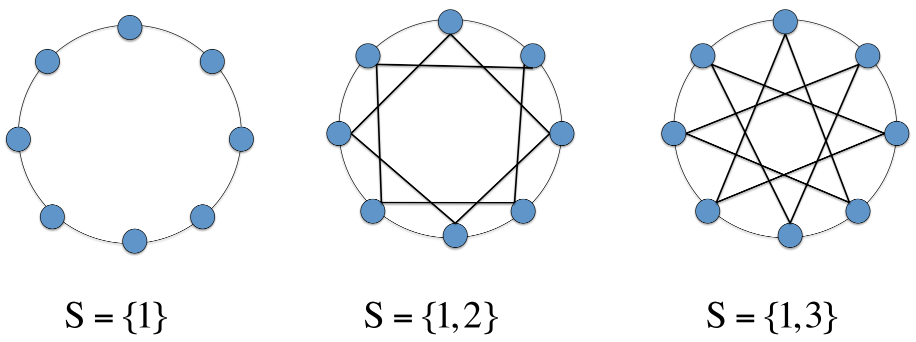


Figure 3.1: Examples of circulant graphs. The generating set S determines the structure of the graph. Note that when all the elements of S is odd, the graph is bipartite.

For simplicity, we consider connected graphs whose sizes are integer powers of 2; that is, $N = 2^n$ for a positive integer n. The following lemma allows for important simplifications.

Lemma 1. *Suppose the set S defines a connected circulant graph G with 2^n nodes. We can always take $s_1 = 1$ in the set S. In other words, there exists another set $T = \{t_1, t_2, ..., t_M\}$ such that $t_1 = 1$ and the circulant graph G' defined with set T is isomorphic to G.*

Proof: It is well known that a circulant graph is connected [20] iff $\gcd(N, s_1, .., s_M) = 1$. Further, it is also known [19] that if $\gcd(\ell, N) = 1$ then the graph defined by the set $T' = (\ell S)_N$, where the multiplication and modulation operations are applied to every element in the set S, is isomorphic to the circulant graph defined by S. However this holds only in one direction, i.e. there could exist isomorphic graphs without this property holding true (see Adam's conjecture [21]).

Thus, if we are given a circulant graph with 2^n nodes that is connected, then there exists $s^* \in S$ such that s^* is odd since $\gcd(s^*, 2^n) = 1$, for the graph to be connected. From Bezout's identity [22], we know that there exist integers a and b such that, $s^* a + 2^n b = 1$, which implies that a is odd. Taking modulus with respect to $N = 2^n$, we get that $1 = (s^* a)_N$. Hence we can construct an isomorphic graph by generating the new set $T = (aS)_N$, since $\gcd(a, 2^n) = 1$, where the element $s^* \in S$ maps to the element 1 in T.

$\blacksquare$

This does not hold true for general circulant graphs. Consider the circulant graph with $N = 6$ and $S = \{2, 3\}$. This graph is connected, and there exists a Hamiltonian cycle[1], but we cannot reorder nodes to get a circulant ordering.

[1]A Hamiltonian cycle [15] is a graph path that visits each vertex exactly once.

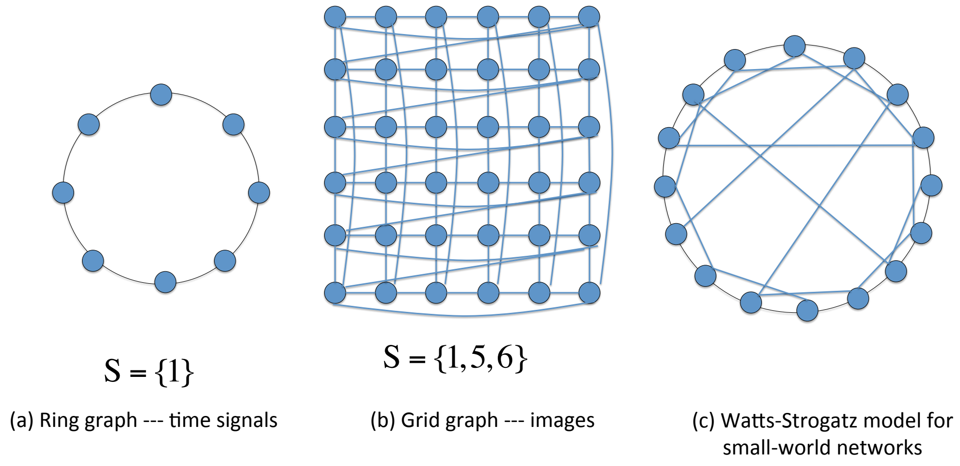

S = {1}
(a) Ring graph --- time signals
S = {1,5,6}
(b) Grid graph --- images
(c) Watts-Strogatz model for small-world networks

Figure 3.2: Examples of circulant graphs in real-world applications. Periodic time signals and images can be viewed as circulant graph signals as shown in (a) and (b). The Watts-Strogatz model for small-world networks starts with a circulant graph and randomly reconnects a fraction of the edges as shown in (c).

Circulant graphs are of interest in certain applications. Clearly periodic time signals are a subclass of circulant graph signals where the graph of interest is the ring graph (see Fig 3.2 (a)). Images can also be viewed as circulant graphs once we take care of the boundaries in the grid graph to make it symmetric (see Fig 3.2 (b)). There are models for small world networks that are built using circulant graphs. In particular, the Watts-Strogatz model [23] is constructed by starting with a circulant graph and randomly reconnecting the edges. In particular, to generate a graph that has certain small-world network properties with average degree $2d$, one constructs a circulant graph with the generating set $\{1, 2, \cdots, d\}$. Given a flip probability β that controls the connectivity properties, each edge is randomly reconnected with probability β. Thus the original circulant graph exhibits strong local connectivity leading to high clustering and the parameter β controls the connectivity across clusters leading to short average path lengths. Fig 3.2 (c) shows an example of a small-world network graph constructed using the Watts-Strogatz model.

3.2.2 A group theoretic viewpoint

Circulant graphs also have some connections to group theory. They belong to a class of graphs known as *Cayley graphs*. Cayley graphs encode the abstract structure of *groups*.

Definition 4. *A finite* group $(\mathcal{G}, *)$ *is a set of elements* $\{g_0, g_1, \cdots, g_{N-1}\}$ *together with an operation* $*$*, satisfying the following axioms,*

- **Closure :** $\forall (g_i, g_j) \in \mathcal{G}, g_i * g_j \in \mathcal{G}$.
- **Associativity :** $\forall (g_i, g_j, g_k) \in \mathcal{G}, (g_i * g_j) * g_k = g_i * (g_j * g_k)$.
- **Identity element:** *There exists an element* $e \in \mathcal{G}$ *such that* $\forall g_i \in \mathcal{G}, g_i * e = e * g_i = g_i$.
- **Inverse element:** $\forall g_i \in \mathcal{G}$ *there exists* $\tilde{g}_i \in \mathcal{G}$ *such that* $g_i * \tilde{g}_i = \tilde{g}_i * g_i = e$.

The set of all integers $\mathbb{Z}$ along with the usual addition operation $+$, i.e. $(\mathbb{Z}, +)$ is a group. The interested reader is referred to [24] for an in-depth treatment of groups.

A generating set S of a group is a subset of elements $\{s_1, s_2, \cdots, s_M\}$ such that every element of the group can be represented as a linear combination of finitely many elements in the subset and their inverses. Given a group $\mathcal{G}$ and a generating set S, a Cayley graph G on N vertices with vertex set V and edge set E is defined as follows,

- Each element $g_i \in \mathcal{G}$ is associated with a vertex $v_i \in V$.
- Every edge in E is of the form $(g, g * s)$ for some $g \in \mathcal{G}$ and $s \in S$.

It is easy to see that circulant graphs are Cayley graphs defined by the group $(\mathbb{Z}, +)$. An rigorous treatment of Cayley graphs and the associated group theory can be found in [18].

Fourier transforms over groups have been well studied [25]. Here, the transform is defined over the elements of the group. However, note that in our case, the group structure only defines the graph. We are interested in transforms of signals defined over the nodes of this graph and the signal could take any value on the reals. One could restrict the signal to take values restricted to a group and this group could be different from the underlying group that defines the graph. Even in this case, it is unclear on how to define notions of a Fourier transform. However, a group theoretic analysis might provide some insights while manipulating the underlying graph itself (e.g. downsampling etc). We will not adopt this viewpoint in our work, but will point out connections wherever appropriate.

3.3 GFT for signals on a circulant graph

The GFT for signals defined on a circulant graph is greatly simplified because the Laplacian matrix is circulant. It is well known that a valid set of eigenvectors for a circulant matrix are the columns of the DFT matrix. The corresponding eigenvalues are in fact the DFT coefficients of the first column of the circulant matrix. Hence it seems like the GFT and the DFT of a signal on a circulant graph are the same. However, there is a slight difference in the ordering of the Fourier basis. The

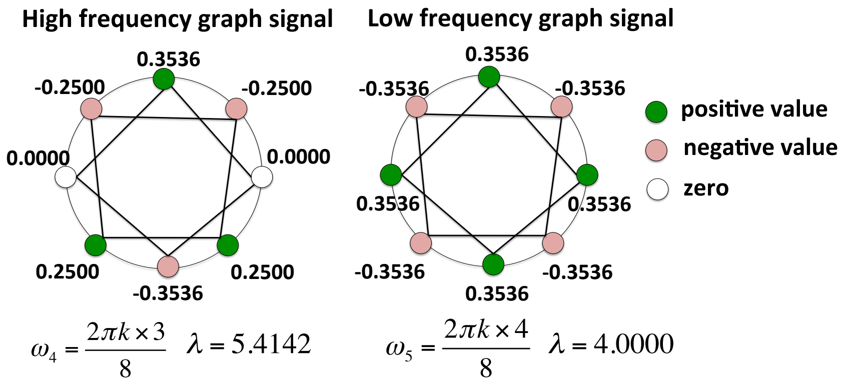

High frequency graph signal
Low frequency graph signal
0.3536
-0.2500
-0.2500
0.0000
0.0000
0.2500
0.2500
-0.3536
-0.3536
-0.3536
0.3536
0.3536
-0.3536
-0.3536
0.3536
positive value
negative value
zero
$\omega_4 = \frac{2\pi k \times 3}{8}$ $\lambda = 5.4142$
$\omega_5 = \frac{2\pi k \times 4}{8}$ $\lambda = 4.0000$

Figure 3.3: Example of Fourier sinusoids on the circulant graph. The sinusoid corresponding the Fourier frequency ω_4 is a higher frequency vector on the graph than that corresponding to ω_5 since the number of sign changes with respect to neighbors is more for ω_4.

k-th highest frequency vector in the DFT matrix corresponds to the frequency $\omega_k = 2\pi k/N$. However this is not necessarily equal to the k-th highest frequency in the GFT. In other words, assuming an ordered set of eigenvalues, the eigenvector corresponding to the k^{th} highest eigenvalue does not necessarily correspond to ω_k.

To illustrate this, Fig. 3.3 shows an example of a 8-node circulant graph. The Fourier vector corresponding to the frequency ω_4 is a higher frequency vector on the graph than that corresponding to ω_5. This is because the signal values of neighboring nodes change signs more frequently for ω_4 than for ω_5. Hence if $\mathbf{X}^{\mathsf{G}}$ is the GFT and $\mathbf{X}^{\mathsf{F}}$ is the regular Fourier transform of a graph signal $\boldsymbol{x}$, then there exists a permutation σ that maps the components of $\mathbf{X}^{\mathsf{F}}$ to $\mathbf{X}^{\mathsf{G}}$ i.e.,

$$X^{\mathsf{G}}(k) \quad = \quad X^{\mathsf{F}}(\sigma(k)).$$

The permutation σ is completely defined by taking the Fourier transform of the first row of the Laplacian matrix (which is the set of eigenvalues) and ordering them in ascending order. The permutation that maps from the original sequence to the ordered sequence is the permutation of interest. Note that the elements of the vector $\boldsymbol{x}$ are assumed to be ordered according to the node ordering that gives a circulant adjacency matrix.

Fig. 3.4 shows an example of such a permutation for the circulant graph in Fig. 3.3. Note that the multiplicity of each eigenvalue except zero and four is at least two; since the eigenvalues are defined by taking the DFT of the real-valued first column of the Laplacian matrix, which renders the transform to be symmetric. The permutation in this case is not unique, since eigenvalues repeat. One could also define the permutation mapping from the corresponding eigenspaces of the GFT and the DFT which would give a unique mapping. However, we just define a single permutation

$$diag(FLF^H) = \begin{bmatrix} 0.0000 \\ 2.5858 \\ 6.0000 \\ 5.4142 \\ 4.0000 \\ 5.4142 \\ 6.0000 \\ 2.5858 \end{bmatrix} \qquad \begin{aligned} \sigma(0) &= 0 \\ \sigma(1) &= 1 \\ \sigma(2) &= 6 \\ \sigma(3) &= 4 \\ \sigma(4) &= 3 \\ \sigma(5) &= 5 \\ \sigma(6) &= 7 \\ \sigma(7) &= 2 \end{aligned}$$

$$\sigma(2) = 6 \implies \text{3rd highest Fourier frequency } (\omega_2) \text{ vector}$$
$$\text{is the 6th highest GFT vector}$$

Figure 3.4: Eigenvalues and the permutation function mapping the Fourier frequencies and the GFT frequencies for the graph shown in Fig. 3.3. $\mathbf{F}$ is the DFT matrix.

mapping between the eigenvalues for notational simplicity since this does not affect the results we discuss here in general.

Note that due to the multiplicity of the eigenvalues, the eigenvectors are also not unique and the Fourier basis is only one valid set of eigenvectors. We can always choose a real set of eigenvectors for the Laplacian matrix. In this work, we restrict ourselves to the DFT basis due to its structure, which will help carry over properties from the time domain to the graph domain.

Despite working with the Fourier basis, there are a couple of challenges. First, aliasing expressions must take into account the permutation in the frequency domain. Second, the operations of "shifting", "convolution" and "sampling" need to be generalized from time domain to circulant graphs since each node has multiple neighbors and these definitions will induce different properties in the GFT domain. We will discuss each of these operations in detail in the chapters to follow.

3.4 The Uncertainty Principle

There is a fundamental tradeoff between sparsity of a signal's support in the graph domain and in the GFT domain. This tradeoff dates back to Heisenberg's uncertainty principle, which fundamentally limits the locality of a signal's support in both time and frequency domains. That is, a signal that is sufficiently localized in time cannot be localized in frequency. Similarly, one would expect that a signal that is highly localized on a graph cannot be localized in the GFT domain. This property has been investigated for general graphs [26], but the derived bounds are not tight. For circulant graphs, since the GFT is a permuted version of the DFT, one can obtain uncertainty principle bounds that are tight.

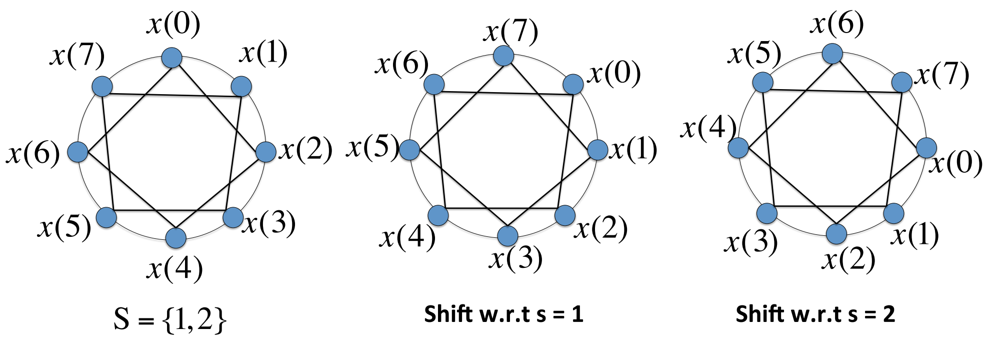


Figure 3.5: Shifting by one hop with respect to different neighbors i.e. elements of the generator set S.

Theorem 1. *(*UNCERTAINTY PRINCIPLE:*) If a signal on a circulant graph with N nodes has N_S non-zero components in the graph domain, and its corresponding GFT has N_G non-zero components, then we have,*

$$N_S N_G \geq N. \tag{3.1}$$

Proof: In classical signal processing, suppose that a N length discrete signal with N_S non-zero signal values has a DFT with N_F non-zero coefficients, then it is well known that [27], the following holds true, $N_S N_F \geq N$. Since this result does not depend on the location of the non-zero coefficients, the result carries over to the GFT for circulant graph signals, since the GFT coefficients are essentially a permutation of the DFT coefficients. Note that the lower bound is tight since it is achieved by the constant graph signal that only has a single non-zero GFT coefficient at the zeroth frequency. ■

3.5 Shifting signals on a circulant graph

3.5.1 Shift in the graph domain

In the time domain, a delay or a shift by one sample involves moving the sample at the current time instant to the next or the previous time instant. If we consider the cycle graph that underlies discrete-time, finite-length signals, a shift by one time step (i.e. one node) would involve cyclically shifting all the values on the nodes in the graph by one index. For a general circulant graph, it is intuitive to define a shift by one as moving the value on each node to the neighboring node. Since it is possible to have more than one neighbor per node as defined by the graph set S, we can define a separate shift operation for each element of that set.

Fig. 3.5 shows examples of shifting with respect to different elements of the set S. If we shift with respect to an odd element of the set, then a shift of N corresponds to the identity shift. However, if we shift with respect to an even set element, then a shift of less than N gives the identity shift. In particular, if $r = gcd(s, N)$, then a shift of N/r gives the identity shift when shifted with respect to the element $s \in S$.

The shift operation can be formalized in terms of the matrix $\mathbf{P}$ defined as follows,

$$\mathbf{P} = \begin{bmatrix} \mathbf{0}_{N-1}^{\mathsf{T}} & 1 \\ \mathbf{I}_{N-1} & \mathbf{0}_{N-1} \end{bmatrix}. \tag{3.2}$$

Note that, given a vector $\boldsymbol{x}$ the operation $\mathbf{P}\boldsymbol{x}$, cyclically shifts the elements of the vector by one step (equivalent to a delay of one in time-domain). Thus for a connected circulant graph, a shift of a graph signal $\boldsymbol{x}$ by one node defined with respect to the element $s = 1 \in S$ corresponds to the operation $\mathbf{P}\boldsymbol{x}$. Following is an example of a shift by one for the graph in Fig. 3.5.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix} = \begin{bmatrix} x(7) \\ x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \end{bmatrix}. \tag{3.3}$$

More generally, a shift by k corresponds to $\mathbf{P}^k\boldsymbol{x}$. Now consider shifting by one with respect to a general element $s \in S$. In this case, the value at node i gets shifted to that at node $(i + s)_N$. The signal is therefore multiplied by a matrix $\mathbf{P}_s$, which is easily shown to equal $\mathbf{P}^s$. Fig. 3.5 shows examples of shifting with respect to different $s \in S$. Thus a shift by k of a signal $\boldsymbol{x}$ on a circulant graph G defined with respect to the element $s \in S$ corresponds to the following operation,

$$\mathbf{P}_s^k\boldsymbol{x} = (\mathbf{P}^s)^k\boldsymbol{x}. \tag{3.4}$$

3.5.2 Shift in the GFT domain

A shift in the GFT domain can again be expressed as a function of the shift matrix $\mathbf{P}$ that we have defined in the previous section. Consider a shift by ℓ in the frequency domain. The new GFT, $\mathbf{Y}^{\mathsf{G}}$, is defined as follows,

$$\mathbf{Y}^{\mathsf{G}} = \mathbf{P}^{\ell}\mathbf{X}^{\mathsf{G}}. \tag{3.5}$$

Suppose that $\boldsymbol{y}$ is the corresponding graph-domain signal obtained after shifting the signal $\boldsymbol{x}$ in the frequency domain, we get the following relation.

$$\begin{aligned} \mathbf{U}^{\mathsf{H}}\boldsymbol{y} &= \mathbf{P}^{\ell}\mathbf{U}^{\mathsf{H}}\boldsymbol{x}, & (3.6)\\ \boldsymbol{y} &= \mathbf{U}\mathbf{P}^{\ell}\mathbf{U}^{\mathsf{H}}\boldsymbol{x}. & (3.7)\end{aligned}$$

Note that since $\mathbf{P}$ is a circulant matrix, it will be diagonalized by the Fourier matrix. Thus if $\mathbf{F}$ is the DFT matrix, then

$$\mathbf{F}\mathbf{P}^{\ell}\mathbf{F}^{\mathsf{H}} = \text{diag}\left(\left[e^{-2\pi k\ell/N}\right]_{k=0}^{N-1}\right), \tag{3.8}$$

since $\mathbf{P}^{\ell}$ is a delay operator for time signals as well. We know that $\mathbf{U}$ is a permuted version of the DFT matrix with the permutation defined by σ. Hence we have that,

$$\mathbf{U}\mathbf{P}^{\ell}\mathbf{U}^{\mathsf{H}} = \left[e^{-j2\pi\sigma(k)\ell/N}\right]_{k=0}^{N-1}. \tag{3.9}$$

Thus the following holds true,

$$\begin{aligned} \mathbf{Y}^{\mathsf{G}} &= \mathbf{P}^{\ell}\mathbf{X}^{\mathsf{G}}, & (3.10)\\ y(k) &= e^{-j2\pi\sigma(k)\ell/N}x(k) \;\; k = 0, \cdots, N-1. & (3.11)\end{aligned}$$

Shifting in the DFT domain is typically used for modulating a signal which is useful in communication applications. Similarly one could modulate multiple bandlimited signals defined over the same graph at different graph frequencies and store the resulting signal. However, it is immediately unclear whether such operations would be useful for graph signals. We have not explicitly used this operation in any of our filter designs or in the examples but have provided it here for sake of completeness.

3.6 Correlation

The correlation operation between two signals in the time domain involves shifting, multiplying, and adding the signals. Given our definition of shifts on a circulant graph, we can define correlation in a similar manner. Let $\boldsymbol{h}$ and $\boldsymbol{x}$ be two vectors defined on the circulant graph G. Define the matrix

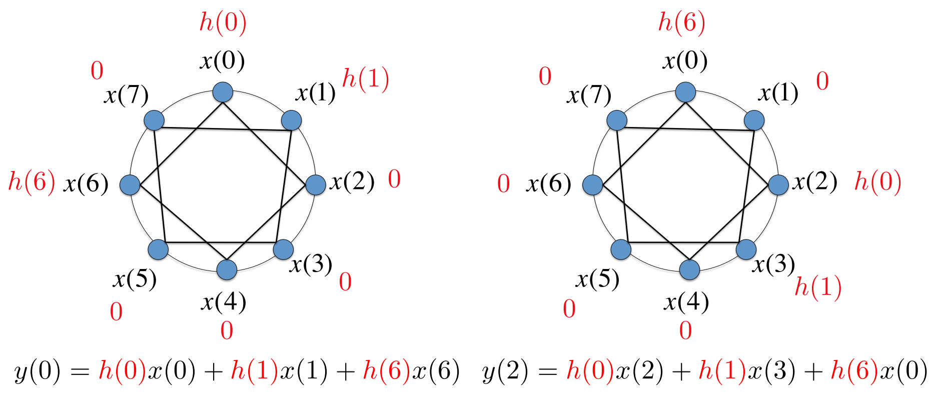


Figure 3.6: Example depicting the correlation operation. The output on two different nodes obtained after correlation is shown in the figure. Note that irrespective of the shift operation (i.e. the element of the generating set S) we choose, the output would remain the same

$\mathbf{H}$ containing the different shifts of $\boldsymbol{h}$ as follows,

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{h}^T \\ (\mathbf{P}\boldsymbol{h})^T \\ (\mathbf{P}^2\boldsymbol{h})^T \\ \cdots \\ (\mathbf{P}^{N-1}\boldsymbol{h})^T \end{bmatrix}. \tag{3.12}$$

The correlation between $\boldsymbol{h}$ and $\boldsymbol{x}$ is then defined as,

$$corr(\boldsymbol{h}, \boldsymbol{x}) \triangleq \mathbf{H}\boldsymbol{x}. \tag{3.13}$$

This definition holds for $s = 1 \in S$. In general for odd values of s, correlation can be defined in a similar manner by replacing $\mathbf{P}$ with $\mathbf{P}_s$. Fig 3.6 shows an example of a correlation operation on a circulant graph.

When s is even, $\mathbf{P}_s^k$ becomes identity for $k < N$. This calls for a new way of shifting an element through all the nodes. In the definition for the matrix $\boldsymbol{h}$, let $k < N$ be such that $\mathbf{P}_s^k = \mathbf{I_N}$. Then we modify the definition so that in the k^{th} row, instead of the identity shift, we have the shift $\mathbf{P}$. Going further, the next set of j shifts would be $\mathbf{P}_s^j\mathbf{P}$. This holds for k steps, and then we then replace $\mathbf{P}$

by $\mathbf{P}^2$ until all the elements are covered. The following matrix can be used to define this operation,

$$\mathbf{H}_s = \begin{bmatrix} \boldsymbol{h}^T \\ (\mathbf{P}_s\boldsymbol{h})^T \\ (\mathbf{P}_s^2\boldsymbol{h})^T \\ \cdots \\ (\mathbf{P}_s^{k-1}\boldsymbol{h})^T \\ (\mathbf{P}_s^0\mathbf{P}\boldsymbol{h})^T \\ (\mathbf{P}_s\mathbf{P}\boldsymbol{h})^T \\ \cdots \\ (\mathbf{P}_s^{k-1}\mathbf{P}\boldsymbol{h})^T \\ (\mathbf{P}_s^0\mathbf{P}^2\boldsymbol{h})^T \\ \cdots \end{bmatrix}. \quad (3.14)$$

One can see that there is a one-to-one mapping between the correlation operations defined with respect to different shifts. The rows of $\mathbf{H}_s$ are just a permuted version of the rows of $\mathbf{H}_1$. Consider the example of a eight node circulant graph with the set $S = \{1, 2\}$ (Fig. 3.5). Following are $\mathbf{H}_1$ and $\mathbf{H}_2$ as defined for this graph.

$$\mathbf{H}_1 = \begin{bmatrix} h(0) & h(1) & h(2) & h(3) & h(4) & h(5) & h(6) & h(7) \\ h(7) & h(0) & h(1) & h(2) & h(3) & h(4) & h(5) & h(6) \\ h(6) & h(7) & h(0) & h(1) & h(2) & h(3) & h(4) & h(5) \\ h(5) & h(6) & h(7) & h(0) & h(1) & h(2) & h(3) & h(4) \\ h(4) & h(5) & h(6) & h(7) & h(0) & h(1) & h(2) & h(3) \\ h(3) & h(4) & h(5) & h(6) & h(7) & h(0) & h(1) & h(2) \\ h(2) & h(3) & h(4) & h(5) & h(6) & h(7) & h(0) & h(1) \\ h(1) & h(2) & h(3) & h(4) & h(5) & h(6) & h(7) & h(0) \end{bmatrix}, \quad (3.15)$$

$$\mathbf{H}_2 = \begin{bmatrix} h(0) & h(1) & h(2) & h(3) & h(4) & h(5) & h(6) & h(7) \\ h(6) & h(7) & h(0) & h(1) & h(2) & h(3) & h(4) & h(5) \\ h(4) & h(5) & h(6) & h(7) & h(0) & h(1) & h(2) & h(3) \\ h(2) & h(3) & h(4) & h(5) & h(6) & h(7) & h(0) & h(1) \\ h(7) & h(0) & h(1) & h(2) & h(3) & h(4) & h(5) & h(6) \\ h(5) & h(6) & h(7) & h(0) & h(1) & h(2) & h(3) & h(4) \\ h(3) & h(4) & h(5) & h(6) & h(7) & h(0) & h(1) & h(2) \\ h(1) & h(2) & h(3) & h(4) & h(5) & h(6) & h(7) & h(0) \end{bmatrix}. \quad (3.16)$$

$\mathbf{H}_2$ is a permuted version of $\mathbf{H}_1$ wherein the second half of the rows of $\mathbf{H}_1$ is interleaved with the first half of the rows. In general, since $\mathbf{P}_s^k = I$, it holds that $\mathbf{H}_1(2,:) = \mathbf{H}_s(k+1,:)$, $\mathbf{H}_1(3,:) =$

$\mathbf{H}_s(2k,:)$, etc. In general one can verify that

$$\mathbf{H}_1(r,:) \quad = \quad \mathbf{H}_s(((r-1)k+1)_N,:). \tag{3.17}$$

Let the output of the correlation operation be given by $\boldsymbol{y}_s$, i.e.,

$$\boldsymbol{y}_s \quad = \quad \mathbf{H}_s\boldsymbol{x}. \tag{3.18}$$

Note that we need to define the mapping of the elements of $\boldsymbol{y}_s$ to the nodes in the graph. It is intuitive to have the mapping consistent with the shift operation i.e. $y_s(0)$ maps to node 0, $y_s(1)$ maps to node s, $y_s(2)$ maps to node $(2s)_N$ and so on. Once we have such a mapping it is clear that $\boldsymbol{y}_s = \boldsymbol{y}_1$, irrespective of the value of s. The reasoning is as follows. The correlation operation takes the weighted average of the signal $\boldsymbol{x}$ on the neighbors of each node where the weights are defined by the coefficients in $\boldsymbol{h}$. Hence it does not matter what shift we apply as long as the coefficients of $\boldsymbol{h}$ are applied to each node. Thus we can restrict ourselves to the case of $s = 1$.

3.7 Convolution

Analogous to classical convolution, we can define convolution here as a correlation of the signal $\boldsymbol{x}$ with a "graph-reversed" version of $\boldsymbol{h}$. For the same reasons as above, we can restrict ourselves to the case of $s = 1$. On a circulant graph, the natural reversed version of the signal $\widetilde{\boldsymbol{h}}$ is defined as follows,

$$\widetilde{h}(k) \quad = \quad h((N-k)_N). \tag{3.19}$$

Thus if $\widetilde{\mathbf{H}}$ is the matrix formed by the different shifts of $\widetilde{\boldsymbol{h}}$ as defined in Equation 3.12, then the convolution operation is defined as,

$$conv(\boldsymbol{h},\boldsymbol{x}) \quad = \quad \widetilde{\mathbf{H}}\boldsymbol{x}. \tag{3.20}$$

Following is an example of $\widetilde{\mathbf{H}}$ defined for the graph shown in Fig 3.5.

$$\widetilde{\mathbf{H}} \quad = \quad \begin{bmatrix} h(0) & h(7) & h(6) & h(5) & h(4) & h(3) & h(2) & h(1) \\ h(1) & h(0) & h(7) & h(6) & h(5) & h(4) & h(3) & h(2) \\ h(2) & h(1) & h(0) & h(7) & h(6) & h(5) & h(4) & h(3) \\ h(3) & h(2) & h(1) & h(0) & h(7) & h(6) & h(5) & h(4) \\ h(4) & h(3) & h(2) & h(1) & h(0) & h(7) & h(6) & h(5) \\ h(5) & h(4) & h(3) & h(2) & h(1) & h(0) & h(7) & h(6) \\ h(6) & h(5) & h(4) & h(3) & h(2) & h(1) & h(0) & h(7) \\ h(7) & h(6) & h(5) & h(4) & h(3) & h(2) & h(1) & h(0) \end{bmatrix}, \tag{3.21}$$

Lemma 2. *Convolution of $\boldsymbol{h}$ and $\boldsymbol{x}$ with respect to $s = 1$ in the graph domain corresponds to the multiplication of the GFT of $\boldsymbol{h}$ and $\boldsymbol{x}$, i.e.*

$$GFT(conv(\boldsymbol{h}, \boldsymbol{x})) \;=\; \mathbf{H}^{\mathsf{G}} \circ \mathbf{X}^{\mathsf{G}},$$

where $\circ$ is the element-wise product of the vectors.

Proof: Let $\boldsymbol{y} = conv(\boldsymbol{h}, \boldsymbol{x})$ or equivalently,

$$\boldsymbol{y} \;=\; \widetilde{\mathbf{H}}\boldsymbol{x}. \tag{3.22}$$

Note that the matrix $\widetilde{\mathbf{H}}$ is circulant. We thus get the following,

$$\begin{aligned}
\boldsymbol{y} &= \widetilde{\mathbf{H}}\boldsymbol{x}, && (3.23)\\
&= \mathbf{U}\text{diag}([H^{\mathsf{G}}(k)]_{k=0}^{N-1})\mathbf{U}^{\mathsf{H}}\boldsymbol{x}, && (3.24)\\
\mathbf{U}^{\mathsf{H}}\boldsymbol{y} &= \text{diag}([H^{\mathsf{G}}(k)]_{k=0}^{N-1})\mathbf{U}^{\mathsf{H}}\boldsymbol{x}, && (3.25)\\
\mathbf{Y}^{G} &= \mathbf{H}^{\mathsf{G}} \circ \mathbf{X}^{\mathsf{G}}, && (3.26)
\end{aligned}$$

where the second equality follows due to the circulant property of the matrix $\widetilde{\mathbf{H}}$. The matrix is therefore diagonalized by the GFT matrix $\mathbf{U}$ with the entries of the diagonal matrix given by $\mathbf{H}^{\mathsf{G}} = \mathbf{U}^{\mathsf{H}}\boldsymbol{h}$ which is the GFT of $\boldsymbol{h}$. ■

3.8 Chapter highlights

- Motivated by the impact of LTI signal processing, circulant graphs that are amenable to LSI signal processing, are introduced. The goal is to analyze these in depth and then extend relevant signal processing operations to general graphs through a circulant decomposition.
- The columns of the DFT matrix form a valid set of eigenvectors of the adjacency matrix of a circulant graph. However the ordering of the eigenvectors is different, i.e. the highest Fourier frequency basis vector is not necessarily the highest graph frequency vector. Nevertheless, many of the DFT properties carry over.
- Basic operations such as shifting, correlation and convolution are appropriately defined for circulant graphs and their corresponding GFT properties are derived.
- Fundamental principles such as the uncertainty principle are also shown to carry over from time domain to graphs in the circulant case.

Chapter 4

Sampling on circulant graphs

4.1 Sampling

Sampling is an important operation that merits a chapter of its own. There exists some literature on sampling graphs to retain different properties of the graph though not in this context [28]. In order to carry out multi-resolution analysis, it is fundamental to define sampling and understand its effect in the GFT domain. Sampling on a graph should be carefully defined since each node can have multiple neighbors, and it is not clear a priori which subset of nodes should be kept. Further, after sampling, the underlying graph on which the downsampled signal resides should also be defined. This is easy to do in linear and planar domains (like discrete-time signals or images), since it is natural for the underlying graph to also be defined as a line or a grid. However, this does not necessarily hold true for general graphs. In this section, we will mostly define the downsampling operation, i.e. given a large circulant graph, how should we sub-sample by a given factor? Any upsampling operation will always be assumed with respect to a downsampled graph and therefore one is assumed to know the original graph and hence can appropriately reconnect the nodes.

As with shifting, we define sampling according to different elements of the set S. Suppose we wish to downsample a graph signal $\boldsymbol{x}$ by two with respect to the element $s = 1$; then we keep every alternate value of $\boldsymbol{x}$ as shown in Fig. 4.1. In general, if we want to downsample by two with respect to an element $s \in S$, we start at node 0 and consider nodes $\{s, 2s, ...\}$, keeping only the alternate elements in this list. We then move to node 1 and repeat the same for the nodes $\{s+1, 2s+1, ...\}$ and so on until we exhaust all the nodes in the graph.

Fig. 4.1 shows the sampling operation with respect to different elements of the set $s \in S$. We use the notation $\downarrow_s m$, to represent the operation of downsampling by m with respect to $s \in S$. The following lemma mathematically characterizes the downsampling pattern obtained when downsampled with respect to different elements of the set S.

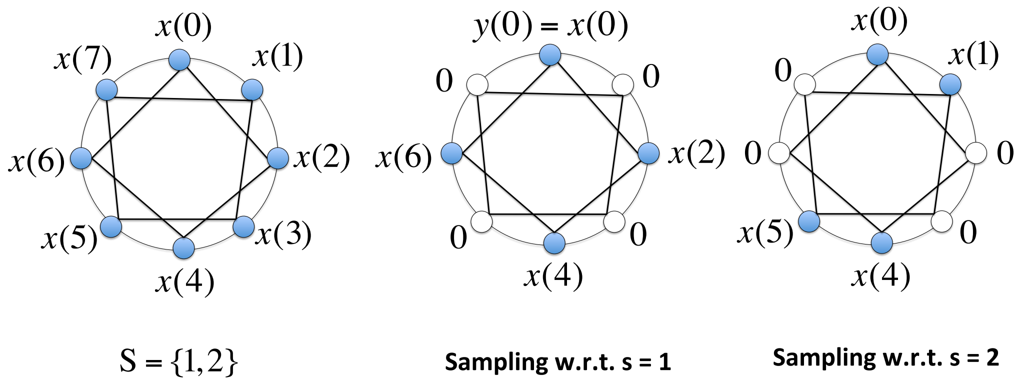


Figure 4.1: Downsample-upsample by 2 operation with respect to different neighbors i.e. elements of the set S.

Lemma 3. *Suppose a circulant graph G with 2^n nodes is defined by set S. Sampling a graph signal on G by a factor of two with respect to $s \in S$ creates a downsampling pattern $\boldsymbol{p} = [\mathbf{1}_r^T \mathbf{0}_r^T \mathbf{1}_r^T \mathbf{0}_r^T ...]$, where $\mathbf{1}_r$ and $\mathbf{0}_r$ are vectors of all-ones and all-zeros, respectively, in $\mathbb{R}^r$ and $r = \gcd(s, N)$.*

Proof: Let $s = 2^k\ell$, where $\ell \geq 1$ is an odd number and $k \geq 0$. Downsampling by a factor of two keeps every alternate node in the set $\{0, s, 2s, ...\}$. The cardinality of this set, is essentially the smallest number m such that $(sm)_N = 0$. Since $N = 2^n$, we have that $\text{lcm}(N, s) = 2^n\ell$ which gives $m = 2^{n-k}$. Thus the cardinality of the set is 2^{n-k} and we have 2^k such sets. Note that $r = \gcd(s, N) = 2^k$. Thus after downsampling, the first r node values are retained. Now consider node elements that are multiples of r. Consider a node with index $ar = a2^k$. This belongs to the set $\{0, s, 2s, ...\}$. This node is retained if its index is an even multiple of s, i.e. $a2^k = (2^k\ell b)_N$ where b is an even number. The following holds true for some q,

$$Nq + a2^k = 2^k\ell b, \tag{4.1}$$

$$2^{n-k}q = (\ell b - a). \tag{4.2}$$

Since b is even, a is also even as the LHS of the above equation is even. Thus every even indexed node of the form ar is retained after sampling from the set $\{0, s, 2s, ...\}$. Similarly if b is odd, since ℓ is odd, a is odd and the nodes whose indices are odd multiples of r are discarded. Repeating this for every other set, we get the sampling pattern $\boldsymbol{p} = [\mathbf{1}_r^\mathsf{T}\, \mathbf{0}_r^\mathsf{T}\, \mathbf{1}_r^\mathsf{T}\, \mathbf{0}_r^\mathsf{T} ...]$.

■

Remark: This does not hold true for general circulant graphs (e.g., the circulant graph with $N = 10$ and $S = \{1, 4\}$—downsampling by two with respect to $s = 4$ does not correspond to the above sampling pattern). Note that downsampling with respect to an odd $s \in S$ on a graph with 2^n

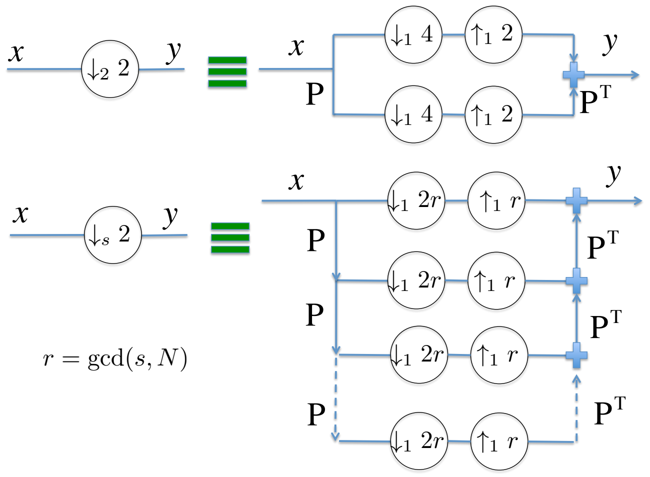


Figure 4.2: Downsampling with respect to $s = 2$ in the graph Fig. 4.1, shown as a function of downsampling with respect to $s = 1$. The second figure shows the same for a general $\downarrow_s$.

nodes would always give the same downsampling pattern.

The above downsampling pattern is reminiscent of block sampling in the time domain, which can be represented as a bank of individual regular samplers. Fig. 4.2 shows an example of representing downsampling with respect to a general $s \in S$ in terms of downsampling by $s = 1$. Note that $\mathbf{P}$ corresponds to the shift-matrix defined with respect to $s = 1$.

We have defined downsampling with respect to different elements of the set S. One wonders if would ever be required to down sample with respect to any other element s other than say $s = 1$. Further, how does one interpret downsampling with respect to different elements of the generating set ? In order to answer these questions, let us consider the simple example of images. Images can be associated with a rectangular grid graph with pixels representing nodes, and pixel adjacency in the horizontal and vertical directions indicating edges. Ignoring border effects, this can be represented as a circulant graph with the set $S = \{1, R\}$, where the image is of dimension $R \times R$ (see Fig 3.2). Intuitively, each $s \in S$ can be thought of as a different dimension of the graph along which we want to downsample. Then the usual downsampling operation on images would require us to downsample in both dimensions, which corresponds to downsampling with respect to $s = 1$ initially and then with respect to $s = R$.

The downsampling operation can also be interpreted as sampling the cosets of the associated Cayley group. Recall the group theoretic interpretation of circulant graphs (section 3.2.2). Each element of the generating set $s \in S$, gives rise to a subgroup that is generated by the set $\{0, s\}$. For the group G, given a sub-group G', the cosets of the sub-group are defined as follows,

$$\begin{aligned} g * G' &= \{g * g' : g' \in G'\} \text{ left coset of } G' \text{ in } G, && (4.3) \\ G' * g &= \{g' * g : g' \in G'\} \text{ right coset of } G' \text{ in } G. && (4.4) \end{aligned}$$

Thus the cosets of the sub-group generated by the set $\{0, s\}$ are of the form $\{0, s, 2s, \cdots\}, \{1, s + 1, 2s + 1, \cdots\}, \cdots, \{r - 1, s + r - 1, 2s + r - 1, \cdots\}$. Note that the left and right cosets of this sub-group are the same since the group operation is commutative i.e. the group is abelian. Thus sampling with respect to the element $s \in S$ is equivalent to sampling the associated cosets of the sub-group generated by this element as described above.

4.2 Aliasing in the GFT domain

Sampling in the graph domain leads to a loss of information in the general case. This loss needs to be reflected in the GFT domain which is explained by the phenomenon of "aliasing". In particular, downsampling in the graph domain causes a mixing up of some of the frequency coefficients in the GFT domain. The aliasing mixing pattern would depend on the element s according to which the graph was sampled. The following lemmas characterize the aliasing pattern in the GFT domain. Recall that $\sigma(\cdot)$ denotes a permutation that maps the standard ordering of DFT vectors to an ordering that arranges the graph's corresponding eigenvalues from smallest to largest.

Lemma 4. *Downsampling and upsampling by 2 with respect to an odd s creates the following aliasing pattern in the frequency domain:*

$$\widetilde{X}^{\mathsf{G}}(k) = \frac{1}{2}X^{\mathsf{G}}(k) + \underbrace{\frac{1}{2}X^{\mathsf{G}}\big(\sigma^{-1}\left[\left(\sigma(k) - N/2\right)_N\right]\big)}_{\text{ALIASING TERM}}, \tag{4.5}$$

where $\widetilde{\mathbf{X}}^{\mathsf{G}}$ is the signal's GFT after downsampling and upsampling.

Proof: Let $\mathbf{X}^{\mathsf{F}}$ denote the usual Fourier transform of the signal $\boldsymbol{x}$. Let $\widetilde{\boldsymbol{x}}$ be the signal obtained after the downsampling-upsampling operation. From classical Fourier transform theory, we know that the following holds true,

$$\widetilde{X}^{\mathsf{F}}(k) = \frac{1}{2}\left(X^{\mathsf{F}}(k) + X^{\mathsf{F}}\left(k - N/2\right)_N\right). \tag{4.6}$$

Further, we have that,

$$\begin{aligned} \widetilde{X}^{\mathsf{G}}(k) &= \widetilde{X}^{\mathsf{F}}(\sigma(k)), && (4.7)\\ &= \frac{1}{2}\left(X^{\mathsf{F}}(\sigma(k)) + X^{\mathsf{F}}\left((\sigma(k)-N/2)_N\right)\right), && (4.8)\\ &= \frac{1}{2}\left(X^{\mathsf{G}}(k) + X^{\mathsf{G}}\left(\sigma^{-1}\left[(\sigma(k)-N/2)_N\right]\right)\right). && (4.9)\end{aligned}$$

■

Fig 4.3 shows an example of the aliasing pattern obtained after downsampling-upsampling by two with respect to $s = 1$ for a circulant graph with 128 nodes with the generating set $\{1, 2\}$. The DFT of the signal is also plotted for contrast. One can observe that the aliasing patterns can be very different depending on the graph structure which determines the permutation.

The above lemma can be generalized for sampling with respect to any $s \in S$ as follows:

Lemma 5. *Downsampling and upsampling by 2 with respect to a general $s \in S$ creates the following aliasing pattern in the frequency domain,*

$$\widetilde{X}^{\mathsf{G}}(k) = \frac{1}{2r}\sum_{m=0}^{r-1}\sum_{\ell=0}^{2r-1} e^{-j\frac{2\pi\ell m}{2r}} X^{\mathsf{G}}\left(\sigma^{-1}\left(\sigma(k) - \ell\frac{N}{2r}\right)_N\right), \tag{4.10}$$

where $r = \gcd(s, N)$.

Proof: Follows on the similar lines as that of Lemma 4, by now considering the aliasing in the DFT domain due to the sampling pattern $\boldsymbol{p}$.

■

Fig. 4.3 shows an example of the aliasing pattern after downsampling-upsampling by two with respect to different $s \in S$. Note that the aliasing pattern can vary significantly depending on the element s according to which the graph is downsampled.

4.3 Reconnection strategies for a sampled graph

4.3.1 Desirable properties

After downsampling a graph signal $\boldsymbol{x}$, it is necessary to define the new, smaller graph on which the sampled signal resides. In time domain, it is easy to define the new time axis after downsampling. For instance, if the even samples are kept, then the new downsampled signal $\widehat{x}(k) = x(2k), k =$

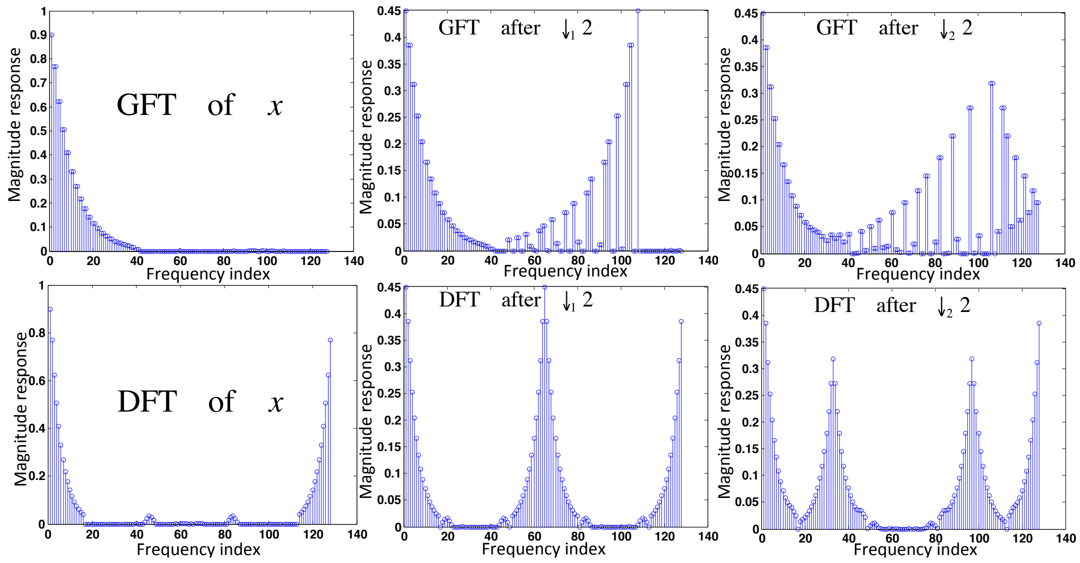


Figure 4.3: Example of a graph signal spectra (GFT) on a circulant graph with number of nodes, $N = 2^7$ and generating set $S = \{1, 2, 3\}$. The spectra obtained after downsampling-upsampling by two with respect to $s = 1$ and $s = 2$ is also shown in the figures. The DFT of the signal is also shown to visually analyze the aliasing pattern.

$0, 1, 2...$ connects the two-hop neighbors in the original line graph. Similarly for images, the two-hop pixels along the horizontal direction are connected after downsampling the image horizontally by a factor of two. This is then treated as a new image and the procedure can be recursed.

For general graphs, determining the connectivity pattern of subsampled nodes is more difficult than for structures like time series and images. Researchers have considered various methods to define the underlying graph structure after sampling [29]. Since we restrict ourselves to circulant graphs, it makes sense to define connectivity based on the graph structure.

Before we discuss the reconnection strategies, we first need to understand the desirable properties of the graph and the signal spectra that we might want to retain in the downsampled graph. Following is a list of desirable properties that one might be interested in retaining while defining the downsampled graph.

- **Closure:** Suppose we start with a structured graph like circulant graphs, it would be desirable for the downsampled graph to also retain the same property, i.e. be circulant in our case. This would help recurse any operations that are defined on the original graph. For example, multiscale analysis for time and images banks on the fact that the signal at every stage is again a time or image signal. This is mostly true irrespective of whether one uses a uniform or a non-uniform downsampling strategy in classical signal processing.

- **Connectivity:** One could always satisfy the Closure property by arbitrarily connecting the nodes so as to retain the original graph structure. However, intuitively it is necessary to maintain the relation between the nodes inherited from the original graph. For example, when we downsample a time signal (viewed as a path graph) we only connect those nodes that were originally connected through a sequence of nodes that were removed after downsampling. A similar property needs to be maintained for graph signals as well.

- **Spectral compaction:** If the original signal was low-pass in nature on the original graph, we would want the GFT of the signal on the new underlying graph to also be low-pass. This property could be maintained to some extent by having a good connectivity property.

- **Computational Efficiency:** In many large scale applications it is necessary that the reconnection algorithms be computationally efficient in order to have real-time processing.

Though all the above properties are desirable, it is difficult to have a single strategy satisfying all of them. In the following sections, we propose two graph-reconnection strategies for circulant graphs and analyze their properties.

4.3.2 Kron reduction

Kron-reduction or Schur-complementation is a popular method to reconnect a downsampled graph [9, 29, 30]. The operation is defined as follows for a general graph. Let α be the subset of the nodes to keep and α^C be the subset of nodes to discard. If $\mathbf{L}$ is the Laplacian matrix, then define the following submatrices,

$$\begin{aligned} \mathbf{L}_1 &= \mathbf{L}(\alpha, \alpha), & (4.11)\\ \mathbf{L}_2 &= \mathbf{L}(\alpha, \alpha^C), & (4.12)\\ \mathbf{L}_3 &= \mathbf{L}(\alpha^C, \alpha^C), & (4.13) \end{aligned}$$

where $\mathbf{L}(A, B)$ is the submatrix of the rows of $\mathbf{L}$ corresponding to the nodes A and columns corresponding to nodes B. The *Kron*-reduced Laplacian matrix of the downsampled graph $\widehat{\mathbf{L}}$ is defined as follows,

$$\widehat{\mathbf{L}} \triangleq \mathbf{L}_1 - \mathbf{L}_2\mathbf{L}_3^{-1}\mathbf{L}_2^{\mathsf{T}}. \quad (4.14)$$

In other words, the new Laplacian is the Schur complement of $\mathbf{L}_1$. The *Kron*-reduced matrix satisfies the following properties [29] which we state here without proof:

Proposition 4. *The Laplacian matrix is closed under* Kron*-reduction, i.e.* $\widehat{\mathbf{L}}$ *is a valid Laplacian.*

Proposition 5. *New edges are introduced only between the nodes in the original graph that were connected by a sequence of nodes that were removed by the downsampling operation.*

Proposition 6. *The eigenvalues of the* Kron-*reduced Laplacian are interlaced between the eigenvalues of the original Laplacian matrix.*

$$0 = \lambda_0(\widehat{\mathbf{L}}) = \lambda_0(\mathbf{L}) \leq \lambda_1(\widehat{\mathbf{L}}) \leq \lambda_1(\mathbf{L}) \leq \lambda_2(\widehat{\mathbf{L}}) \leq \lambda_2(\mathbf{L}) \leq .... \leq \lambda_{N-1}(\widehat{\mathbf{L}}) \leq \lambda_{N-1}(\mathbf{L}). \quad (4.15)$$

There are other properties of *Kron*-reduction that we do not detail here and refer the reader to [29, 30] for more details.

We now analyze the properties of the *Kron*-reduced graph when the original graph is circulant. Let us consider the case when we downsample with respect to odd $s \in S$. We then have the following,

Theorem 2. *(*Closure:*) A circulant graph G having Laplacian matrix $\mathbf{L}$ and generating set S is Kron-reduced with respect to an odd s in S. The Laplacian matrix $\widehat{\mathbf{L}}$ of the resulting graph $\widehat{G}$ is circulant.*

Proof: Recall that the node ordering is chosen such that the Laplacian matrix is circulant. In this case, when we downsample with respect to an odd $s \in S$, we have that the set α consists of even nodes, i.e. $\alpha = \{0, 2, 4, ...\}$. Since $\mathbf{L}_1$ only consists of edges between the even nodes, it is symmetric circulant matrix with the even elements of set S. In other words $L_1(i, j) = 1$ only when $j = (i + s/2)_{N/2}$ where s is some even element of S. Similarly $\mathbf{L}_3$ is also a symmetric circulant matrix since it consists of edges between the odd nodes. $\mathbf{L}_2$ consists of the edges between the even nodes and the odd nodes. Hence it is also a circulant matrix defined corresponding to the odd elements of set S. However this is not a symmetric matrix since this contains only the edges from α to α^C and not the other way around. We are given that,

$$\widehat{\mathbf{L}} \;=\; \mathbf{L}_1 - \mathbf{L}_2\mathbf{L}_3^{-1}\mathbf{L}_2^{\mathsf{T}}. \quad (4.16)$$

Since every matrix in the right hand side of the above equation is circulant, the resulting matrix $\widehat{\mathbf{L}}$ is also circulant. Thus the newly constructed downsampled graph is circulant.

■

Remark: Unfortunately, downsampling with respect to an even s in S does not yield a circulant Laplacian matrix. A counterexample is a circulant graph with $N = 8$ and $S = \{1, 2\}$. Suppose we downsample with respect to $s = 2$, one can verify that we do not even end up with a vertex-transitive graph i.e. edge weights are not symmetric across nodes. This holds true even for images

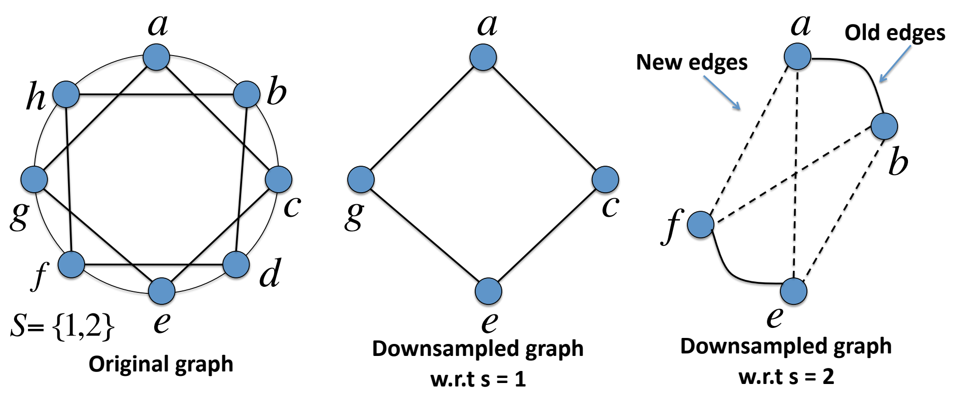


Figure 4.4: Graphs obtained after downsampling and reconnecting according the circulant strategy.

where Kron reduction does not maintain the closure property for different downsampling patterns even though in classical signal processing we connect the nodes in such a way that closure is maintained.

The computational complexity of $Kron$-reduction is mainly determined by the matrix multiplication/inversion operations which is $O(N^{2.3})$. $Kron$-reduction also works for weighted graphs and the resultant downsampled graph is also weighted (even for unweighted graphs).

In the next section, we describe a strategy that preserves the circulant structure of the parent graph irrespective of the downsampling strategy used.

4.3.3 Circulant-preserving reconnection strategy

Since the downsampling pattern depends on the element $s \in S$, we expect the downsampled graph structure to depend on s, too. Let us call the edges generated by odd elements of S as odd edges, and the rest even edges. When downsampled with respect to an odd s, it is easy to verify that all the odd edges are removed, and the even edges connecting the downsampled nodes are retained. The connectivity strategy that we adopt reconnects odd edges between two-hop neighbors. As an example, if the original graph has the set $S = \{1, 3, 4\}$ and we downsample w.r.t. $s = 1$, the downsampled graph has the set $S = \{1, 3, 2\}$. We do not add double-edges to any pair of nodes.

When downsampled with respect to an even s, a subset of the odd and even edges is removed, so it is unclear how to reconnect the nodes. To retain the circulant structure, we adopt the following strategy. We reintroduce edges to complete the cycle corresponding to the edges that were removed. This is best explained using the example in Fig. 4.4. We add new edges (b, e) and (a, f) to complete the cycle with respect to $s = 1$. We also add edges (a, e) and (b, f), which correspond to the two-hop neighbors along the edges generated by $s = 2$ in the parent graph.

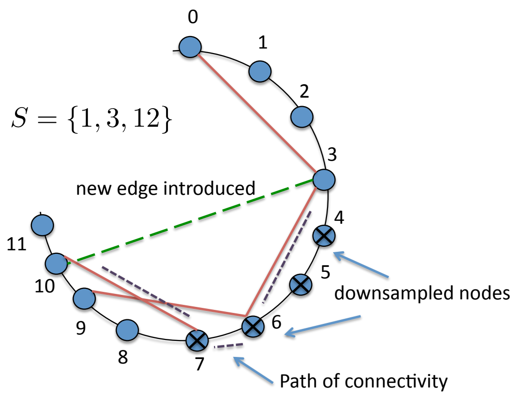


Figure 4.5: Example of a circulant graph with $N = 32$ and $S = \{1, 3, 12\}$, for Theorem 3 showing the path connecting nodes 3 and 10 that were newly connected in the downsampled graph, through a set of nodes that were removed in the original graph after downsampling with respect to $s = 12$.

The following theorem illustrates certain connectivity properties of this reconnection strategy.

Theorem 3. (CONNECTIVITY:) *The circulant-preserving reconnection strategy has the following property: Every pair of newly reconnected nodes was originally connected by a path whose nodes were all removed by downsampling.*

Proof: Without loss of generality we assume that $s = 1 \in S$ since the graph is connected. Suppose we are downsampling with respect to $s^* \in S$ and $r^* = \gcd(s^*, N)$. Due to the downsampling by s^*, we will be keeping the nodes (0 to $r^* - 1$), ($2r^*$ to $3r^* - 1$) and so on based on Lemma 3. Consider the edges in the original graph generated by element $s \in S$. If all such edges were removed (i.e. $\gcd(s, N) = r^*$), then we would be connecting the two-hop neighbors, which by definition were previously connected by a path consisting of the removed nodes and the edges generated by s. Hence we need to consider only those elements s for which a subset of the edges were removed.

Let $s < r^*$. Thus we must introduce new edges between nodes in adjacent pairs of sets that we are keeping, e.g. nodes in (0 to $r^* - 1$) and ($2r^*$ to $3r^* - 1$). It is easy to see that the nodes between which we are introducing new edges are connected through the nodes that were removed i.e. (r^* to $2r^* - 1$); this holds because each of the nodes in the removed set are connected by the edges due to $s = 1$. The following example illustrates this argument.

Consider a circulant graph with $S = \{1, 3, 12\}$ and $N = 32$ (see Fig. 4.5). Let us downsample with respect to $s = 12$. Since $\gcd(12, 32) = 4$, we would keep the nodes,

$$\{(0, 1, 2, 3), (8, 9, 10, 11), (16, 17, 18, 19)....\}.$$

In this case, some fraction of the edges corresponding to $s = 3$ are lost. In the new graph where $(8, 9, 10, 11)$ would be renumbered as $(4, 5, 6, 7)$, we have an edge between 0 and 3. Hence to make the new graph circulant, we introduce edges between 3 and 10 (new node 6). Clearly 3 and 10 were connected in the original graph through the path $(3, 6, 7, 10)$ where 6 and 7 are discarded after downsampling.

A similar argument can be carried out for $s > r^*$. In this case, we just need to choose a path that goes through multiple subsets that were removed before reaching the node to which a new edge was introduced. Such a path can always be found by hopping over edges generated by s and 1.

■

Remark : *Kron* reduction connects *every* pair of nodes that were originally connected through a sequence of removed nodes [29, 30], while our circulant-preserving reconnection strategy connects only a subset thereof. Further the *Kron* strategy usually results in a weighted Laplacian matrix, whereas the circulant strategy gives an unweighted Laplacian matrix by construction.

4.3.4 Examples

Fig 4.6 shows an example of downsampling and reconnection using the proposed strategies on a circulant graph with 128 nodes and $S = \{1, 3\}$. The downsampling is carried out with respect to $s = 1$, i.e. the sampling is carried out along the outer rings in the graph shown in the figure. Since we are downsampling with respect to an odd element of the generating set, Kron reduction gives a circulant graph that is also shown in the figure. Edges are only shown between the nodes when the weights exceed a threshold. The graph obtained after circulant reconnection is shown as well. A low-pass graph signal is associated with the original graph. In this example, the Kron reduced graph seems to better capture the spectral characteristics of the original graph signal.

Fig 4.7 shows an example of downsampling and reconnection using the proposed strategies on a circulant graph with 128 nodes and $S = \{1, 3, 4\}$. The downsampling is carried out with respect to $s = 4$. The Kron-reduced graph, though symmetric is not circulant in this case. The spectral properties of the original graph signal are not that well captured by either of the reconnection strategies in this case. This is an important property that needs further exploration.

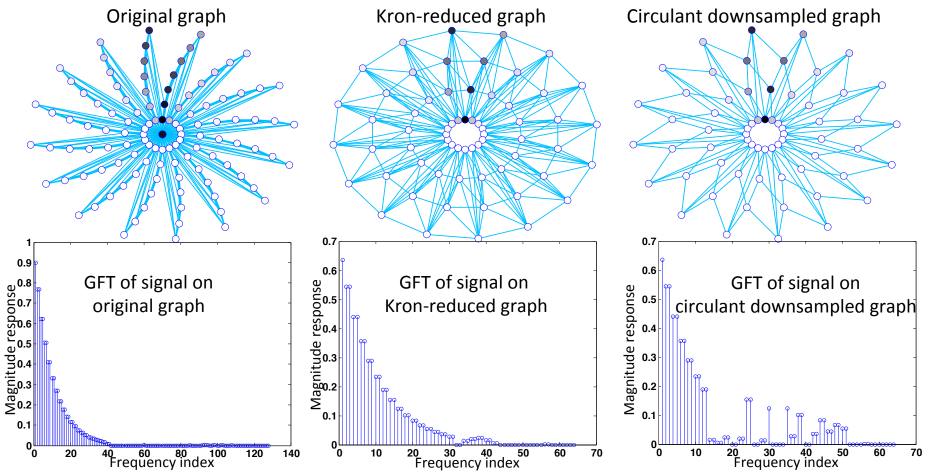


Figure 4.6: Example illustrating the downsampled graphs obtained after Kron-reduction and circulant reconnection. The original graph circulant graph has 128 nodes with the generating set $S = \{1, 3\}$. In this example, since we are downsampling by odd $s \in S$, the Kron reduction also gives a circulant graph. A low-pass graph signal is associated with the original graph and the corresponding spectra in the downsampled graphs is plotted. For this example, Kron reduction seems to retain the spectral characteristics of the signal on the downsampled graph better than the circulant preserving strategy.

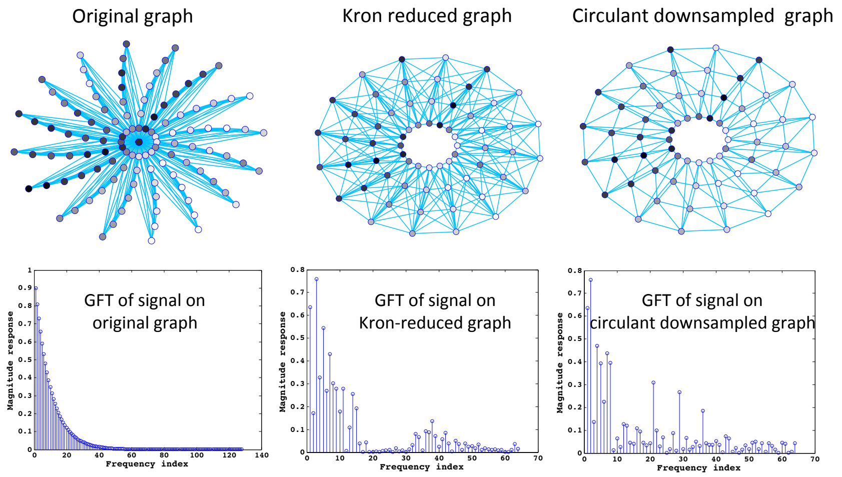


Figure 4.7: Example illustrating the downsampled graphs obtained after Kron-reduction and circulant reconnection. The original graph circulant graph has 128 nodes with the generating set $S = \{1, 3, 4\}$. The downsampling is carried out with respect to $s = 4$. The Kron-reduced graph even though looks symmetric is not circulant. A low-pass graph signal is associated with the original graph and the corresponding spectra in the downsampled graphs is plotted.

4.4 Sampling theorems

4.4.1 Alias-free Sampling

Definition 5. *The GFT bandwidth of a graph signal is defined as the maximum spectral spread in the GFT domain—that is, the maximum k such that $X^G(k) \neq 0$.*

The following theorems provide sufficient Nyquist-like conditions for alias-free downsampling.

Theorem 4. *(Sampling: Odd s) A signal having GFT bandwidth B can be downsampled by 2 with respect to an odd s, without aliasing, if $B \leq \max_{\widetilde{k} \in \{0,\cdots,N-1\}} \widetilde{k}$ such that*

$$\widetilde{k} < \min_{k \in \{0,\cdots,\widetilde{k}\}} \sigma^{-1}[(\sigma(k) - N/2)_N]. \tag{4.17}$$

Proof: The proof is based on the observation that the $\widetilde{k}^{\text{th}}$ GFT coefficient is guaranteed to *not* be aliased by any of the frequencies less than $\widetilde{k}$ after downsampling, if Inequality (4.17) is satisfied as seen from Lemma 4. ■

Corollary 1. *If S only has odd elements, then we have alias-free recovery provided that $B < N/2$.*

Proof: If S only contains odd elements, then the graph is bipartite and the proof follows from using the result in [31]. ■

The next theorem shows that we can define a *Nyquist* bandwidth for sampling with respect to the even elements of the generating set S.

Theorem 5. *(Sampling: Even s) A signal having GFT bandwidth B can be downsampled by 2 with respect to an even s, without aliasing, as long as we have that $B \leq \max_{\widetilde{k} \in \{0,\cdots,N-1\}} \widetilde{k}$ such that*

$$\widetilde{k} < \min_{k \in \{0,\cdots,\widetilde{k}\}} \min_{\ell \in \{1,\ldots,2r-1\}} \left(\sigma^{-1} \left[\left(\sigma(k) - \ell \frac{N}{2r} \right)_N \right] \right), \tag{4.18}$$

where $r = \gcd(s, N)$.

Proof: Similar to Theorem 4, the proof follows from Lemma 5.

■

Theorems 4 and 5 can be quite conservative depending on the graphs. For bipartite graphs, the results are tight. Tighter bounds for general circulant graphs could be obtained from multi-band sampling theory, which we discuss in the next section.

4.4.2 Optimal Perfect-Reconstruction Sampling

Restricting oneself to alias-free sampling can result in retaining more number of samples than is actually needed to losslessly represent the signal. We discuss tighter results here wherein for graph-signal with k-nonzero GFT coefficients, one only needs to retain k appropriately chosen graph-signal components.

Theorem 6. *A graph signal $\boldsymbol{x}$ defined on a circulant graph with N nodes, having a GFT $\mathbf{X}^G$ that has k non-zero coefficients at known locations, can be exactly recovered from any k sized subset of $\boldsymbol{x}$ when N is prime.*

Proof: Recall that $\boldsymbol{x} = \mathbf{U}\mathbf{X}^G$, where $\mathbf{U}$ is the eigenvector matrix of the Laplacian matrix $\mathbf{L}$, the columns of which are the permuted columns of the DFT matrix. We are given that $\mathbf{X}^G$ has only k non-zero coefficients at known locations. Let α be the index set of the coefficients of $\mathbf{X}^G$ that are non-zero. Let β be any k sized index set. We then have the relation, $\boldsymbol{x}(\beta) = \mathbf{U}(\beta, \alpha)\mathbf{X}^G(\alpha)$, where $\mathbf{U}(\beta, \alpha)$ is a $k \times k$ sub-matrix of $\mathbf{U}$ with row indices corresponding to the set β and the column indices corresponding to α. From Chebotarëv's theorem [32], we know that any square sub-matrix of a DFT matrix is full-rank when N is prime. Thus $\mathbf{X}^G$ can be recovered from $\boldsymbol{x}(\beta)$ since $\mathbf{U}(\beta, \alpha)$ is invertible, as long as $|\beta| = k$. ■

Theorem 7. *A graph signal $\boldsymbol{x}$ defined on a circulant graph with N nodes, having a GFT $\mathbf{X}^G$ that has k non-zero coefficients at known locations, can be exactly recovered by retaining signal values on any consecutive k nodes. Note that the choice of node indices assumes that the node ordering is chosen such that the adjacency matrix is circulant.*

Proof: Any $k \times k$ sub-matrix of the DFT matrix consisting of either k consecutive rows or columns is full rank [33]. Thus the linear transformation from the $\mathbf{X}^G$ to the retained consecutive

k elements of $\boldsymbol{x}$ is invertible giving us the desired result.

$\blacksquare$

Remark: There are various other subsets of rows of the DFT matrix which yield a full-rank matrix [33]. By keeping signal values corresponding to these node indices, one can again recover the original signal. In practice, the choice of the subset to keep depends on the application. In many a cases we might want to keep nodes that are well spread out and choose a sampling pattern that might be amenable to a simpler underlying sampled graph description. Further, there are also many other results in the literature corresponding to blind sampling wherein, one does not need to know the support of the non-zero coefficients of $\mathbf{X}^G$ to sample [34]. We do not discuss these results here.

4.5 Chapter highlights

- The sampling operation for circulant graph signals is defined based on different elements of the generating set of the circulant graph.
- For graph signals, it is important to define the new graph on which the downsampled signal resides. The desirable properties such as closure, connectivity, spectral compaction and computation efficiency of a reconnection strategy are listed and two graph reconnection strategies are described—*Kron* reconnection and Circulant-preserving reconnection strategy.
- *Kron* reconnection satisfies nice connectivity properties and results in a circulant graph under certain conditions.
- Circulant reconnection strategy always results in a circulant graph and has some restricted connectivity properties.
- Sampling in the graph domain leads to aliasing in the GFT domain. Aliasing expressions are derived for the signal after sampling under the proposed sampling strategies.
- Nyquist-like bandwidth conditions for alias-free sampling are derived, which are shown to be conservative in general. For bipartite circulant graphs, these are shown to be tight.
- Optimal perfect-reconstruction sampling is discussed and conditions are derived, under which a k-GFT sparse graph signal can be recovered from k-signal components.

Chapter 5

Multiscale analysis on circulant graphs

Multirate signal processing in time domain allows for compression and efficient processing of signals. Analogously in the graph domain, representations of graph signals at different resolutions can facilitate computationally heavy operations and reduce storage costs, while still giving detailed localized information. For instance, we might want a set of basis vectors that capture localized signal variations on the graph and also localized variations in the GFT domain. This concept corresponds to time domain wavelets; wavelets provide time-frequency tilings wherein bases range from being localized in time but spread out in frequency to localized in frequency but not in time. These are useful for representing natural signals comprised of slowly varying time content with interspersed high frequency events. In particular, they help characterize the location of high frequency events while simultaneously characterizing the frequency components of the slowly varying part.

As in time domain, it would be of interest to obtain bases that are localized in the graph domain to capture local signal variations. For example, consider a large Facebook social graph with the signal defined as number of ad clicks per user. Given this data, one might wish to understand which groups of users are most responsive to the ad and whether these users are strongly localized on the graph. Similarly, neural stimulation could result in localized brain activity, with the location corresponding to a subsection of the brain connectivity graph. For example, we might want to know the strength of the neuron activity in the thalamus. This information is contained in the energy of the wavelet components localized in the 'thalamus' region of the graph. Thus obtaining wavelet functions on graphs has been a topic of significant research interest.

In this section we will discuss the basic notions of linear filtering of graph signals and associated filter bank designs for multiscale analysis. The proposed filter banks generate wavelet-like functions on graphs that satisfy many properties of interest.

5.1 Linear Filters for Graph Signals

Filters form one of the basic tools of signal processing. In the most general form, a filter is a function that maps an input graph signal of length N to an output signal of length N. We will focus on linear filters where the function of interest is linear. Thus if $\boldsymbol{x}$ is the input signal and $\boldsymbol{y}$ the output signal, then we have that,

$$\boldsymbol{y} = \mathbf{H}\boldsymbol{x}, \tag{5.1}$$

where $\mathbf{H}$ is a matrix that represents the linear filter. In general, the filter $\mathbf{H}$ could be arbitrary. However we would be interested in imposing a structure on $\mathbf{H}$ for both ease of design and computational efficiency. For example, in the case of time signals, finite impulse response (FIR) filters constitute an important category of filters. We want an analogous definition of filters for graph signals.

A k-tap FIR filter takes weighted averages of signal values up to k-time instants. Similarly, we can define the notion of a k-hop filter on a graph as follows. Let $\mathcal{I}_{\mathbf{A}}$ denote the *modified indicator matrix* for $\mathbf{A}$ defined as follows,

$$\mathcal{I}_A(i,j) = \begin{cases} 1 & A(i,j) \neq 0, i \neq j, \\ 0 & A(i,j) = 0, i \neq j, \\ 0 & \quad i = j. \end{cases} \tag{5.2}$$

The diagonal entries are defined to be zero for simplicity, which will be useful when we consider higher powers of the adjacency matrix.

Definition 6. *(A 1-Hop Linear Filter): We say $\mathbf{H}$ is a 1-hop linear filter on a graph G if $\mathbf{H} = \mathbf{H} \circ (\mathcal{I}_{\mathbf{A}} + \mathbf{I})$, where $\mathbf{I}$ is the identity matrix.*

Thus we want the non-zero off-diagonal entries of $\mathbf{H}$ coincide with those of the graph's adjacency matrix $\mathbf{A}$. The symbol $\circ$ denotes the standard Hadamard (element-wise) product. It is known that the number of walks of length k from node i to node j is given by the entry (i,j) of $(\mathcal{I}_{\mathbf{A}})^k$ [35]. So, a k-hop linear filter is characterized by the k-hop indicator matrix $\mathcal{I}_{(\mathcal{I}_{\mathbf{A}})^k}$, which is the same as $\mathcal{I}_{\mathbf{A}^k}$ since $\mathbf{A}$ has no negative entries.

The primary reason for discussing circulant graphs has been that they are amenable to shift-invariant processing. Therefore it is of interest to formally define the notion of linear shift-invariant (LSI) filters.

Definition 7. *(Linear Shift-Invariant Filtering on Graphs): A filter $\mathbf{H}$ is LSI if $\mathbf{P}\,\mathbf{H}\,\boldsymbol{x} = \mathbf{H}\,\mathbf{P}\,\boldsymbol{x}$ for every graph signal $\boldsymbol{x}$.*

Here $\mathbf{P}$ is the shift-matrix as defined earlier. Note that it is sufficient to consider the shift matrix corresponding to $s = 1$ for reasons mentioned while defining correlation and convolution (see Section 3.6). Irrespective of the shift chosen, the intuition is that the filter weights remain the same for different nodes in the graph.

Proposition 8. *A filter* $\mathbf{H}$ *is LSI if and only if it can be written in the form,* $\mathbf{H} = \sum_{j=0}^{N-1} h(j)\mathbf{P}^j$*, where* $\boldsymbol{h} = [h(0), \cdots, h(N-1)]^{\mathsf{T}}$*. Thus* $\mathbf{H}$ *is a circulant matrix.*

Proof: From the definition of shift-invariance, we need $\mathbf{H}$ and $\mathbf{P}$ to commute, which requires $\mathbf{H}$ to be a polynomial in $\mathbf{P}$ [36]. By choosing $\boldsymbol{x}$ as $\mathbf{P}^j[1\ \mathbf{0}_{N-1}^{\mathsf{T}}]^{\mathsf{T}}$, for $j = 0, \cdots, N-1$, the above theorem can be verified. ■

Proposition 9. *For a LSI filter* $\mathbf{H}$ *on a circulant graph, if* $\boldsymbol{y} = \mathbf{H}\boldsymbol{x}$*, then we have that* $\mathbf{Y}^G = \mathbf{H}^G \circ \mathbf{X}^G$*, where* $\mathbf{Y}^G, \mathbf{H}^G$ *and* $\mathbf{X}^G$ *are the GFTs of* $\boldsymbol{y}, \boldsymbol{h}$ *and* $\boldsymbol{x}$ *respectively.*

Proof: Since $\mathbf{H}$ is circulant, we know that it is diagonalizable by the GFT basis and hence the proof follows.

■

$\mathbf{H}^{\mathsf{G}}$ denotes the graph-frequency response of the LSI filter $\mathbf{H}$, i.e.$\mathbf{H}^{\mathsf{G}} = \mathbf{U}^{\mathsf{T}}\boldsymbol{h}$.

Definition 8. *(Orthogonal filters): A filter* $\mathbf{H}$ *is said to be orthogonal if* $\mathbf{H}^{\mathsf{T}}\mathbf{H} = \mathbf{I}$.

Definition 9. *(Graph Independence): A filter* $\mathbf{H}$ *is said to be graph-independent if the form of the filters does not change with the graph.*

For example, consider the filter $\mathbf{H} = \mathbf{A} + 2\mathbf{A}^2$, where $\mathbf{A}$ is the adjacency matrix of the graph. Note that the form of the filter i.e. the coefficients of $\mathbf{A}$ and $\mathbf{A}^2$ are independent of the graph and only the matrix $\mathbf{A}$ changes with the graph.
As in classical signal processing, filters can be designed either in the signal (graph) domain or in the transform (GFT) domain. This corresponds to designing the filters either in the graph domain or in the GFT domain. A filter designed in the GFT domain would have the following structure,

$$\mathbf{H} = \sum_{k=0}^{N-1} H_G(k)\boldsymbol{u}_k\boldsymbol{u}_k^{\mathsf{H}}, \tag{5.3}$$

where one can choose the coefficients $\mathbf{H}_G$ to obtain a desired response. This is analogous to designing standard filters such as Chebyshev and Butterworth filters in classical signal processing. However, if care is not taken, the graph domain response might have undesired artifacts.

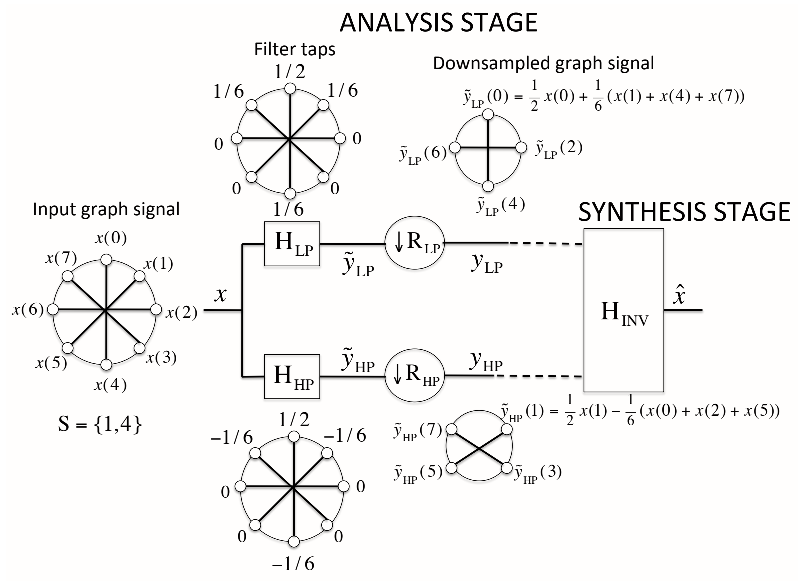


Figure 5.1: Two-channel linear filter bank for graph signals with an example of a 8-node circulant graph. The input signal is passed through two-filters—LP and HP separately in each of the branches and then downsampled. This forms the analysis stage. In the synthesis stage, the inverse filter obtains an estimate of the input signal based on the output of these two branches.

The other option is to design filters in the graph domain by restricting the number of hops of the response which would also limit the range of frequency responses that one might obtain in the frequency domain. The choice of the design methodology largely depends on the application at hand. We shall mainly focus on graph-domain designs.

5.2 Filter banks for multiscale analysis

Multiscale analysis using filter banks typically proceeds as follows. The input signal is appropriately filtered into low-pass (LP) and high-pass (HP) components using two filters and the outputs are downsampled. The reason for using LP and HP filters is that, the LP signal output typically captures a large fraction of the energy of the original signal and the HP captures the detailed coefficients. Hence if one wants to work on a downsampled version of the signal, then the LP branch would provide a reasonable approximation. This corresponds to level one. The same procedure is iterated on the LP branches in the successive levels to get representations at each of the different levels which correspond to multiple scales. Thus once we define filters for the first level, the same can potentially be iterated at the different levels. Hence the reason to consider two-channel filter banks.

Definition 10. *A two-channel linear filterbank for the graph signal $\boldsymbol{x}$ is a collection of filters $\mathbf{H}_{\mathsf{LP}}$, $\mathbf{H}_{\mathsf{HP}}$, and $\mathbf{H}_{\mathsf{INV}}$, where $\mathbf{H}_{\mathsf{LP}}$ and $\mathbf{H}_{\mathsf{HP}}$ retain the low-frequency and high-frequency content of the input signal, respectively, and $\mathbf{H}_{\mathsf{INV}}$ is an inverse filter that obtains an estimate of the input signal $\widehat{\boldsymbol{x}}$ from $\boldsymbol{y}_{\mathsf{LP}}$ and $\boldsymbol{y}_{\mathsf{HP}}$, the downsampled versions of $\widetilde{\boldsymbol{y}}_{\mathsf{LP}}$ and $\widetilde{\boldsymbol{y}}_{\mathsf{HP}}$, the LP and HP filter outputs, respectively.*

The filters $\mathbf{H}_{\mathsf{LP}}$ and $\mathbf{H}_{\mathsf{HP}}$ are collectively called as the *analysis* filters and the filter $\mathbf{H}_{\mathsf{INV}}$ is called as the *synthesis* filter. Fig. 5.1, is a block-diagram depiction of a two-channel linear filterbank with an example of a 8-node circulant graph.

We have different classes of filter banks depending upon various properties that are satisfied. Following is a list of desirable properties for a two-channel filter bank.

- **Perfect-reconstruction:** The filter bank is said to satisfy the perfect reconstruction property if it is lossless i.e., using the outputs of the analysis stage one can exactly reconstruct the original graph signal using the synthesis filter.
- **Critical-sampling:** The number of components retained at the output of the filter bank is equal to the length of the input signal. For example, in Fig. 5.1, the filter bank is said to be critically sampled if the downsampling rates R_{LP} and R_{HP} are such that $N/R_{\mathsf{LP}} + N/R_{\mathsf{HP}} = N$. Critical sampling helps control the problem dimensionality without explosion at multiple levels.
- **Localization:** The filters should be able to capture the properties of the graph signal, localized in the graph domain as well as the frequency domain at multiple scales. This is one of the major goals of multiscale analysis—obtain a good graph domain and graph frequency domain tiling.
- **Localized reconstruction:** It would be desirable for the inverse filter $\mathbf{H}_{\mathsf{INV}}$ to also be localized, since this leads to an efficient implementation. However, this is not fundamentally necessary for analyzing the signal properties.
- **Orthogonality:** The filter bank is said to be orthogonal if each of the filters in the filter bank are orthogonal.
- **Graph Independence:** The filter bank is said to be graph-independent, if each of the filters are graph-independent. This is a desirable property particularly for multiscale analysis, since the downsampled graphs obtained at each stage may have a different structure than the original graph we started with (e.g. in Fig. 5.1). In such a case, we might want to retain the same filter structure without having to redesign the filters at every stage.
- **Diagonalizability:** The filters should be diagonalizable by the eigenvectors of the Laplacian matrix. For circulant graphs, all LSI filters satisfy this property. However, for general graphs we shall later see that not all filter designs necessarily satisfy this property.

The above set of properties are desirable, but not mandatory for all filter bank designs. The choice depends on the application. It might be advantageous to compromise on certain properties to gain in others. In the sections to follow, we will discuss three different classes of filter banks that satisfy different properties.

5.3 Oversampled Filter banks—The Graph Laplacian Pyramid

The Laplacian pyramid [37] is one of the most popular filter bank structures in classical signal processing with an oversampled representation. It provides significant design flexibility at the loss of the critical sampling property. The filter bank is defined as follows[1]. For the LP stage, any suitable LP filter can be chosen and the output is downsampled to get the LP coefficients. The HP output is obtained by subtracting the LP filtered signal from the original signal. Formally, we have the following,

$$\begin{aligned} \widetilde{\boldsymbol{y}}_{\mathsf{LP}} &= \mathbf{H}_{\mathsf{LP}}\boldsymbol{x}, & (5.4)\\ \boldsymbol{y}_{\mathsf{LP}} &= (\widetilde{\boldsymbol{y}}_{\mathsf{LP}})\downarrow_2, & (5.5)\\ \boldsymbol{y}_{\mathsf{HP}} &= \boldsymbol{x} - \widetilde{\boldsymbol{y}}_{\mathsf{LP}}, & (5.6)\end{aligned}$$

Note that the LP filter response is chosen so that it can be downsampled by two without loss of information. The reconstruction filter is very simplistic. One needs to upsample the downsampled LP signal and pass it through a LP filter to get $\widetilde{\boldsymbol{y}}_{\mathsf{LP}}$. This is then added to the HP output, $\boldsymbol{y}_{\mathsf{HP}}$ to exactly reconstruct the original signal.

The Laplacian pyramid filter bank structure satisfies the following set of properties.

- The filter bank is perfect reconstruction by construction as long as the LP signal can be reconstructed from the downsampled output.
- The number of coefficients retained at each stage is $N + N/2 = 3N/2$ for an input signal of size N. Thus it is not critically sampled.
- One has considerable flexibility in designing the LP filter as long as the signal obtained after LP filtering can be recovered from its downsampled version. Thus the filter can be designed to satisfy most of the other desired properties detailed in Section 5.2.

[1] Similar filter bank structures have been proposed in the existing literature [38].

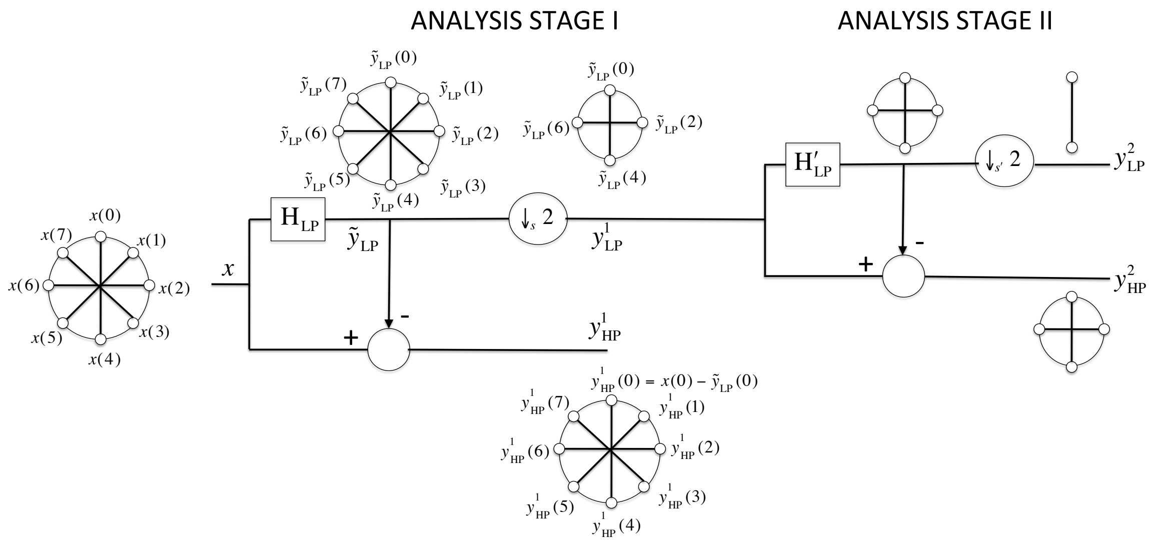

ANALYSIS STAGE I
ANALYSIS STAGE II
$x(0)$ $x(1)$ $x(2)$ $x(3)$ $x(4)$ $x(5)$ $x(6)$ $x(7)$
x
H_{LP}
$\tilde{y}_{\mathrm{LP}}(0)$ $\tilde{y}_{\mathrm{LP}}(1)$ $\tilde{y}_{\mathrm{LP}}(2)$ $\tilde{y}_{\mathrm{LP}}(3)$ $\tilde{y}_{\mathrm{LP}}(4)$ $\tilde{y}_{\mathrm{LP}}(5)$ $\tilde{y}_{\mathrm{LP}}(6)$ $\tilde{y}_{\mathrm{LP}}(7)$
$\tilde{y}_{\mathrm{LP}}$
$\downarrow_s 2$
y^1_{LP}
$\mathrm{H}'_{\mathrm{LP}}$
$\downarrow_{s'} 2$
y^2_{LP}
y^2_{HP}
y^1_{HP}
$y^1_{\mathrm{HP}}(0) = x(0) - \tilde{y}_{\mathrm{LP}}(0)$
$y^1_{\mathrm{HP}}(1)$ $y^1_{\mathrm{HP}}(2)$ $y^1_{\mathrm{HP}}(3)$ $y^1_{\mathrm{HP}}(4)$ $y^1_{\mathrm{HP}}(5)$ $y^1_{\mathrm{HP}}(6)$ $y^1_{\mathrm{HP}}(7)$

Figure 5.2: Block diagram description of the analysis stage of the Graph Laplacian Pyramid filter bank. Note that the downsampling operations are chosen with respect to some element $s \in S$ which would depend on the graph. Further the filters at different stages could be different depending on the graph. They could also be graph-independent depending on the design of the LP filter.

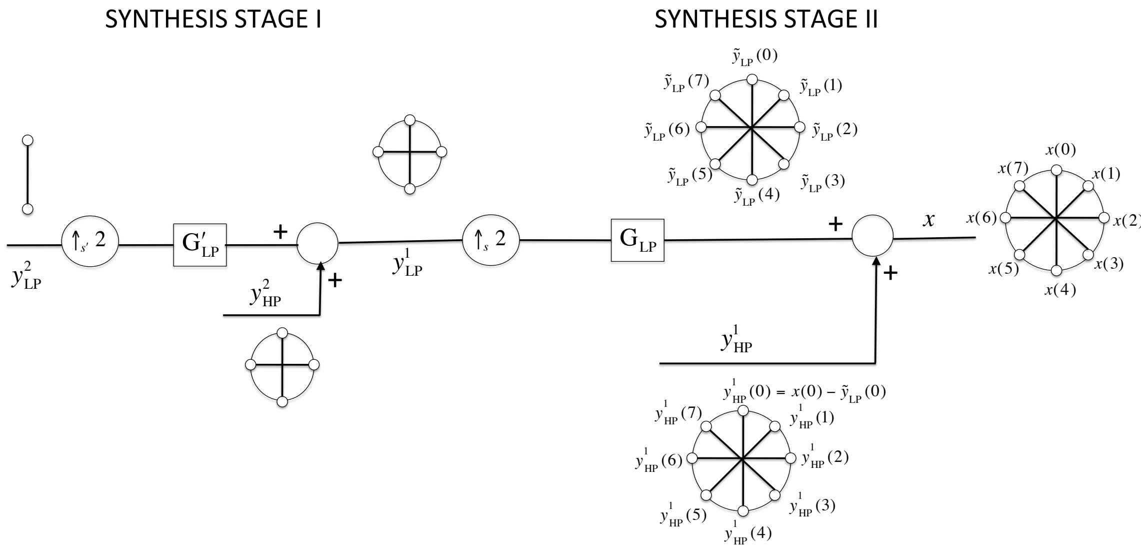

SYNTHESIS STAGE I
SYNTHESIS STAGE II
y^2_{LP}
$\uparrow_{s'} 2$
$\mathrm{G}'_{\mathrm{LP}}$
y^2_{HP}
y^1_{LP}
$\uparrow_s 2$
G_{LP}
$\tilde{y}_{\mathrm{LP}}(0)$ $\tilde{y}_{\mathrm{LP}}(1)$ $\tilde{y}_{\mathrm{LP}}(2)$ $\tilde{y}_{\mathrm{LP}}(3)$ $\tilde{y}_{\mathrm{LP}}(4)$ $\tilde{y}_{\mathrm{LP}}(5)$ $\tilde{y}_{\mathrm{LP}}(6)$ $\tilde{y}_{\mathrm{LP}}(7)$
y^1_{HP}
$y^1_{\mathrm{HP}}(0) = x(0) - \tilde{y}_{\mathrm{LP}}(0)$
$y^1_{\mathrm{HP}}(1)$ $y^1_{\mathrm{HP}}(2)$ $y^1_{\mathrm{HP}}(3)$ $y^1_{\mathrm{HP}}(4)$ $y^1_{\mathrm{HP}}(5)$ $y^1_{\mathrm{HP}}(6)$ $y^1_{\mathrm{HP}}(7)$
x
$x(0)$ $x(1)$ $x(2)$ $x(3)$ $x(4)$ $x(5)$ $x(6)$ $x(7)$

Figure 5.3: Block diagram description of the synthesis stage for the analysis filters shown in Fig 5.2. Note that the upsampling operations are fixed once the downsampling pattern is fixed in the analysis stage. The synthesis filters are chosen such that the LP filtered signal can be reconstructed from their downsampled versions.

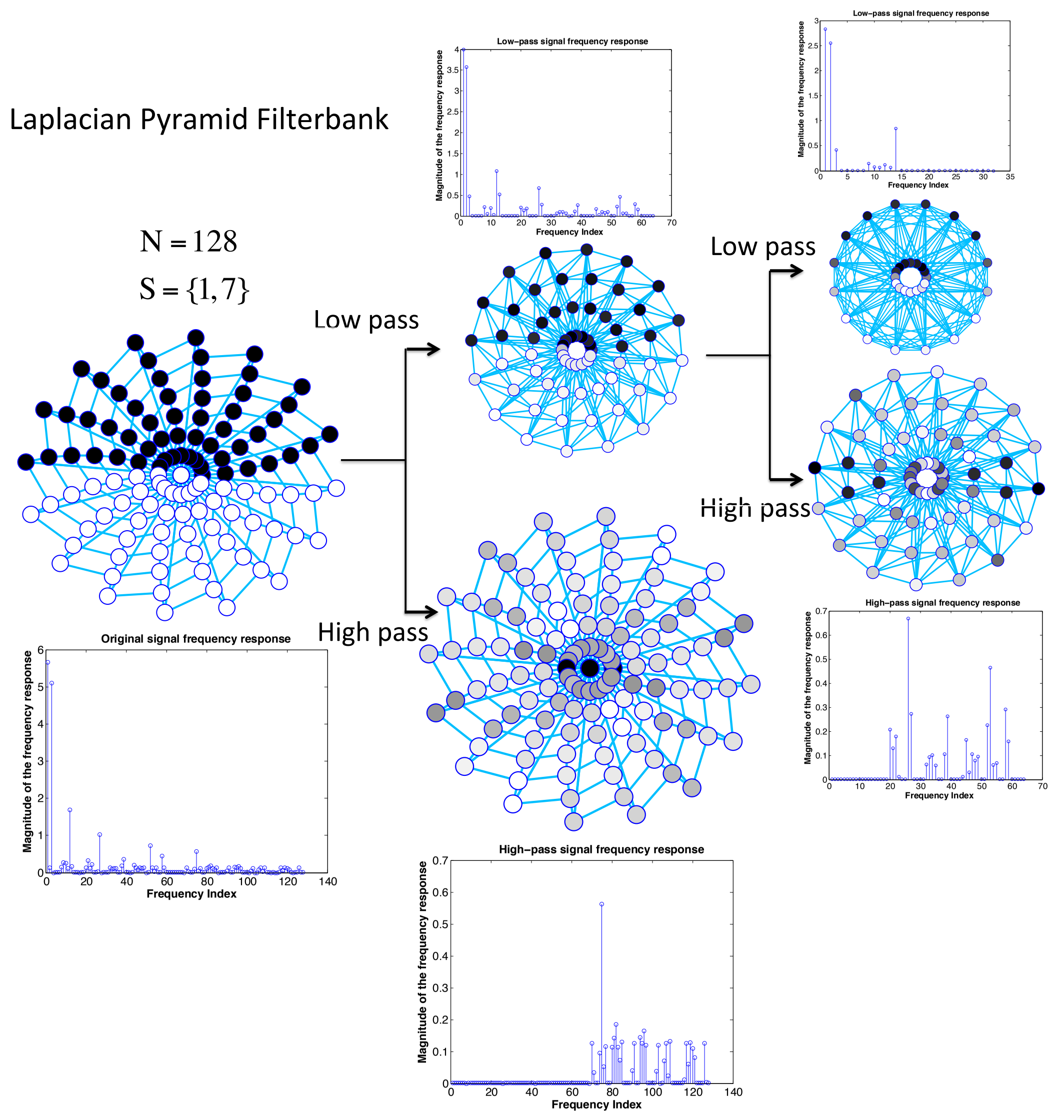

Laplacian Pyramid Filterbank
N = 128
S = {1,7}
Low pass
High pass
Low pass
High pass
Original signal frequency response
Low−pass signal frequency response
High−pass signal frequency response
Magnitude of the frequency response
Frequency Index

Figure 5.4: Two-stage multiscale decomposition of a circulant graph signal using the Graph Laplacian pyramid. The circulant graph chosen here has the parameters $N = 128$ and $S = \{1, 7\}$. The signal is binary with all ones on one half of the nodes and zeros on the others. The LP filter used in both the stages is an ideal LP filter with cutoff frequency equal to the alias-free sampling bandwidth for this graph when downsampled with respect to the first element in the set S. Note that the alias-free bandwidth here is $N/2$ since the graph is also bipartite. The HP output is obtained after subtracting the LP output from the original signal and the effect of this can be observed in the frequency response.

Fig 5.2 and Fig 5.3 are block diagram descriptions of the analysis and synthesis stages of a two-stage Graph Laplacian pyramid filter bank with an example of a 8-node circulant graph. There are a few points to be noted here. The downsampling operation depends on the element of the generator set S which can be chosen according to the application of interest. The LP filter is to be chosen such that the LP filtered signal can be reconstructed after downsampling. For example, it could be chosen such that the output signal bandwidth is less than the bandwidth required for alias free sampling on the given graph for the chosen downsampling pattern. The filter design could be such that the filters are graph-independent and could be reused in the subsequent stages. The synthesis stage shown in Fig 5.3 is fixed once the design choices in the analysis stage have been made.

Fig 5.4 shows an example multiscale decomposition of a synthetic circulant graph signal. The circulant graph shown here has 128 nodes and the generating set is given by $S = \{1, 7\}$. The signal on this graph is binary with all ones on one half of the nodes (top half) and zeros otherwise. The LP filter in both the stages has an ideal response with cutoff at the Nyquist bandwidth for signals on this graph (see Theorem 4) when downsampled with respect to the element $s = 1$ since the downsampling is carried out with respect to this element in both the stages. Note that the Nyquist bandwidth in this case is $N/2$ since the graphs are also bipartite. The downsampled graphs are obtained through Kron-reduction and are circulant since we are downsampling with respect to an odd element of the generating set.

5.4 Spline-like filterbank structures

Oversampled filter bank representations provide certain flexibility in design. However, the primary disadvantage is that the dimensionality increases with every subsequent stage which is undesirable especially when we are dealing with large scale graph datasets spanning millions of nodes where the dimensionality needs to be kept under control. This motivates us to look for alternative designs that are critically sampled. In this section we will discuss a class of filter banks inspired by the classical first order spline-filters [39]. The first class of filters we discuss have a fixed structure which we later generalize to provide some flexibility in design.

5.4.1 Simple Spline-like filter banks

The simplest set of filters that one can think of for graph signals would be those that take weighted averages with neighboring nodes for the LP filter and weighted differences for HP. The natural choice of weights is to take them to be uniform for all the neighbors. This is the basic underlying idea for the simple Spline-like filter banks. Formally, the filter bank outputs can be defined as follows,

$$\widetilde{y}_{\mathsf{LP}}(k) \;=\; \frac{1}{2}\left[x(k)+\frac{1}{d}\sum_{j\in\mathcal{N}(k)} x(j)\right], \tag{5.7}$$

$$\widetilde{y}_{\mathsf{HP}}(k) \;=\; \frac{1}{2}\left[x(k)-\frac{1}{d}\sum_{j\in\mathcal{N}(k)} x(j)\right], \tag{5.8}$$

where $\boldsymbol{x}$ is the input signal, $\widetilde{\boldsymbol{y}}_{\mathsf{LP}}$ and $\widetilde{\boldsymbol{y}}_{\mathsf{HP}}$ are the filtered LP and HP signals, d is the degree of each node in the circulant graph, and $\mathcal{N}(k) = \{m \mid A(k,m) \neq 0\}$ is the one-hop neighborhood of node k. We can express the LP and HP filters succinctly as follows:

$$\begin{aligned}
\mathbf{H}_{\mathsf{LP}} &= \frac{1}{2}\left(\mathbf{I}_N + \frac{1}{d}\mathbf{A}\right), && (5.9)\\
&= \frac{1}{2d}(\mathbf{D}+\mathbf{A}), && (5.10)\\
&= \frac{1}{2d}(2\mathbf{D}-\mathbf{L}), && (5.11)\\
&= \left(\mathbf{I}_N - \frac{1}{2d}\mathbf{L}\right). && (5.12)\\
\mathbf{H}_{\mathsf{HP}} &= \frac{1}{2}\left(\mathbf{I}_N - \frac{1}{d}\mathbf{A}\right), && (5.13)\\
&= \frac{1}{2d}(\mathbf{D}-\mathbf{A}), && (5.14)\\
&= \frac{1}{2d}\mathbf{L}, && (5.15)
\end{aligned}$$

where $\mathbf{I}_N$ is the identity matrix. Thus we also have that,

$$\mathbf{H}_{\mathsf{LP}} \;=\; \mathbf{I}_N - \mathbf{H}_{\mathsf{HP}}. \tag{5.16}$$

The example shown in Fig 5.1 is an example of a simple Spline-like filter bank on an 8-node circulant graph. The LP and HP filters defined above are graph generalizations of three-tap FIR filters F_{LP} and F_{HP} in classical discrete-time signal processing, which have impulse responses characterized by $f_{\mathsf{LP}}(\ell) = 0.25\,\delta(\ell+1) + 0.5\,\delta(\ell) + 0.25\,\delta(\ell-1)$ and $f_{\mathsf{HP}}(\ell) = -0.25\,\delta(\ell+1) + 0.5\,\delta(\ell) - 0.25\,\delta(\ell-1)$, respectively, where $\delta(\cdot)$ is the Kronecker impulse. Filters F_{LP} and F_{HP} are first-order discrete-time spline wavelets which motivates the Spline terminology we use here.

The simple Spline-like filter banks are clearly shift-invariant by definition for circulant graphs. It is straightforward to see that the filters $\mathbf{H}_{\mathsf{LP}}$ and $\mathbf{H}_{\mathsf{HP}}$ are circulant and hence commute with the shift

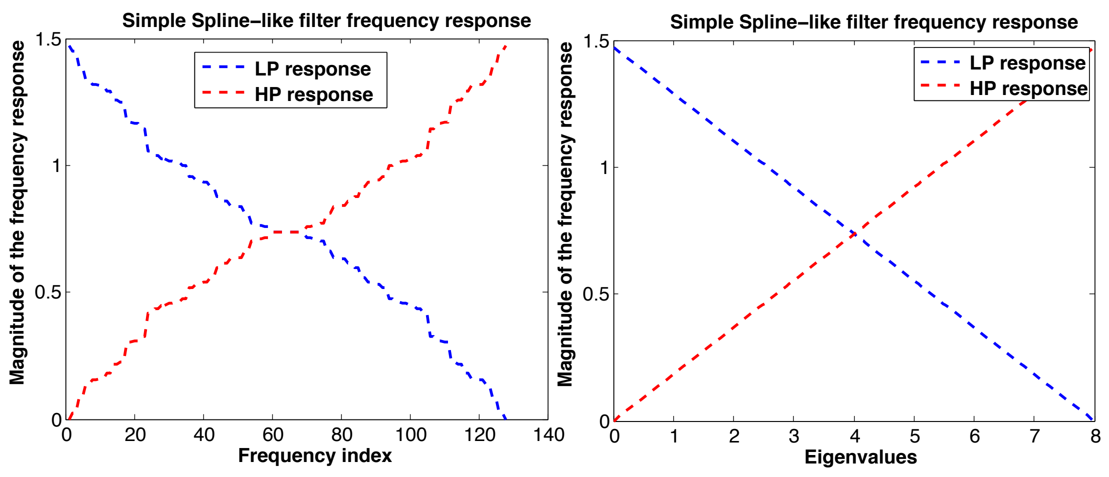


Figure 5.5: Example of a simple spline-like filter response for a circulant graph with $N = 128$ and $S = \{1, 7\}$. The LP and HP filter responses are shown in the figures. Fig (a) shows the response plotted as a function of the frequency index. The LP and HP responses are complementary as expected. However, the shape of the filters is quite non-intuitive. On the other hand, if one observes the frequency response plotted as a function of the eigenvalues as in Fig (b), then we get an intuitively pleasing response.

matrix $\mathbf{P}$. Given the forms of the filters in Equation 5.12 and Equation 5.15, the filter response can be easily derived as follows,

$$\mathbf{H}_{\mathsf{HP}} = \frac{1}{2d}\mathbf{L}, \tag{5.17}$$

$$= \frac{1}{2d}\mathbf{U}\Lambda\mathbf{U}^{\mathsf{H}}. \tag{5.18}$$

$$\mathbf{H}_{\mathsf{LP}} = \mathbf{I}_N - \mathbf{H}_{\mathsf{HP}}, \tag{5.19}$$

$$= \mathbf{U}\left(\mathbf{I}_N - \frac{1}{2d}\Lambda\right)\mathbf{U}^{\mathsf{H}}. \tag{5.20}$$

Thus the corresponding filter responses are given as follows,

$$H^{\mathsf{G}}_{\mathsf{LP}}(k) = 1 - \frac{\lambda_k}{2d}, \tag{5.21}$$

$$H^{\mathsf{G}}_{\mathsf{HP}}(k) = \frac{\lambda_k}{2d}. \tag{5.22}$$

Fig 5.5 shows an example of a simple spline-like filter response for a circulant graph with $N = 128$ and $S = \{1, 7\}$. The LP and HP filter responses are shown in the figures. Fig (a) shows the response plotted as a function of the frequency index. The LP and HP responses are complementary as expected. However, the shape of the filters is quite non-intuitive. On the other hand, if one observes the frequency response plotted as a function of the eigenvalues as in Fig (b), then we get an intuitively pleasing response. There are some works in the literature that treat the eigenvalues as the frequencies [9]. However, it is unclear as to how the eigenvalues need to be interpreted (see Section 2.3 for a discussion on this). But in this specific case, given the structure of the filters, one obtains a linear response when the x-axis is taken to be the eigenvalues.

Fig 5.6 shows an example of a multistage decomposition of a circulant graph using the simple Spline filter bank. The HP signal does have some LP signal content given that the frequency response of the filters is not that sharp as compared to an ideal filter that was employed in the Laplacian pyramid case (see Fig 5.4). However, visually the LP signal still retains most of the information of the original signal and the advantage we gain over the Laplacian pyramid is that we need to only keep N components at any stage to recover the original signal.

Theorem 10. *For a connected circulant graph, the spline filters defined in Eqns. 5.12 and 5.15 form a critically-sampled perfect-reconstruction LSI filterbank for any downsampling pattern as long as at least one LP component is retained and the complementary set of nodes retain the HP component.*

Proof: The key step here is to show that the linear transformation from the input to the output is invertible i.e. the synthesis filter provides an exact reconstruction of the input signal. Let $\mathbf{K}$ be a diagonal matrix with $K(i,i) = 1$ if node i has the LP output after sampling, and -1 otherwise. The output $\boldsymbol{y}$ of the two-channel filterbank after downsampling can be written as follows:

$$\boldsymbol{y} \;=\; \frac{1}{2}\left(\mathbf{I}_N + \frac{1}{d}\mathbf{K}\mathbf{A}\right)\boldsymbol{x}, \tag{5.23}$$

where $\boldsymbol{y}$ consists of an interleaving of vectors $\boldsymbol{y}_{\mathsf{LP}}$ and $\boldsymbol{y}_{\mathsf{HP}}$ obtained after downsampling $\widetilde{\boldsymbol{y}}_{\mathsf{LP}}$ and $\widetilde{\boldsymbol{y}}_{\mathsf{HP}}$, respectively, as illustrated in Fig. 5.1. For the filterbank to be invertible, $\left(\mathbf{I}_N + \frac{1}{d}\mathbf{K}\mathbf{A}\right)$ must be full-rank. We prove this by showing that the null space of this matrix is trivial.

Consider the spectral factorization $\frac{1}{d}\mathbf{A} = \mathbf{V}\boldsymbol{\Gamma}\mathbf{V}^{\mathsf{H}}$. Suppose that $\boldsymbol{z} = \mathbf{V}\boldsymbol{r}$ lies in the null space of $\left(\mathbf{I}_N + \frac{1}{d}\mathbf{K}\mathbf{A}\right)$, where $\boldsymbol{r}$ is just the representation of $\boldsymbol{z}$ in the basis set comprised of the columns of

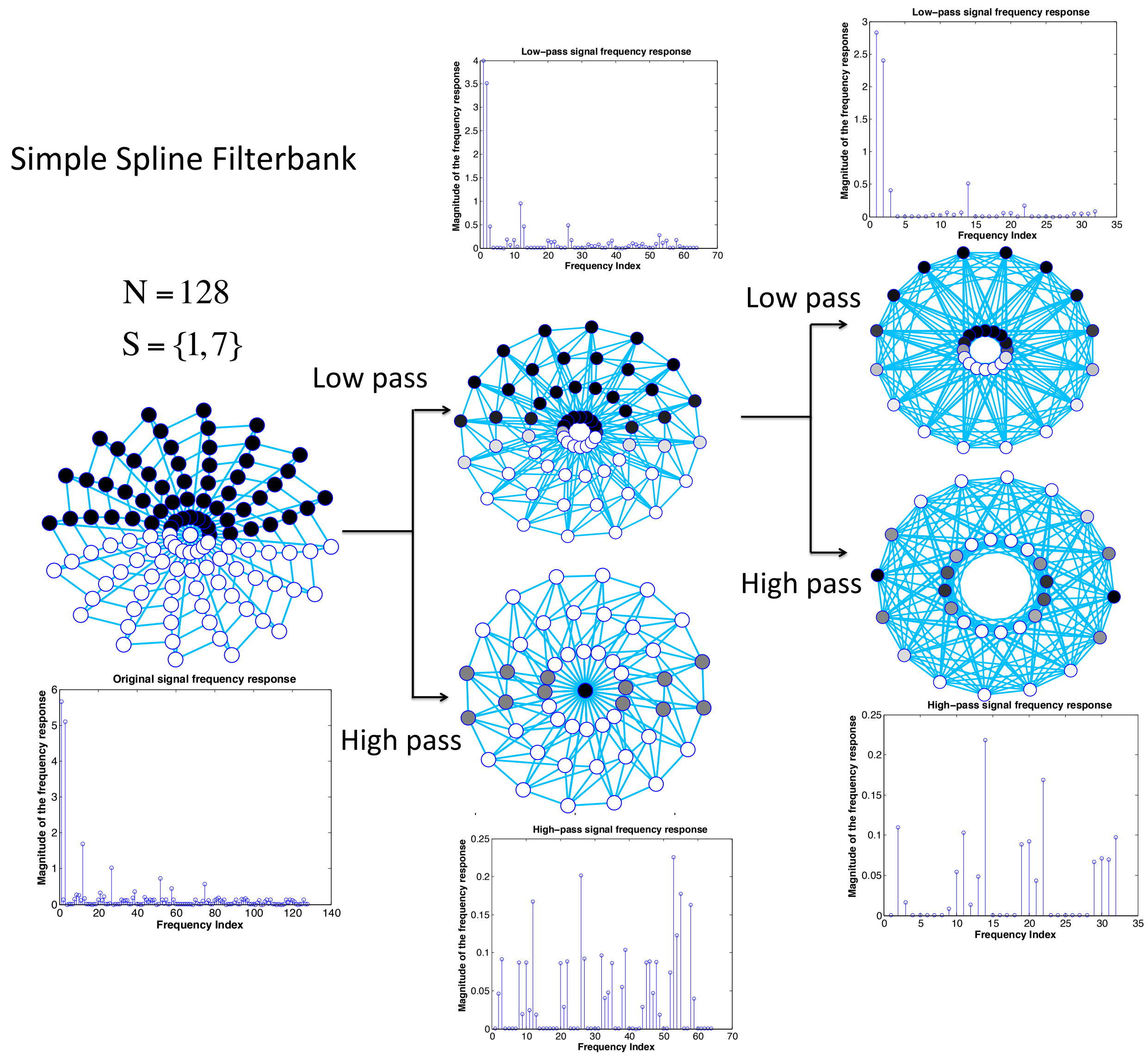

Simple Spline Filterbank
N = 128
S = {1,7}
Low pass
High pass
Low pass
High pass
Original signal frequency response
Low−pass signal frequency response
High−pass signal frequency response
Magnitude of the frequency response
Frequency Index

Figure 5.6: Two stage decomposition of a circulant graph with $N = 128$ and $S = \{1, 7\}$. The simple Spline-like filters are used for the LP and HP analysis filters. The downsampling is carried out with respect to $s = 1$ and the *Kron*-reconnection strategy is employed to determine the downsampled graphs.

$\mathbf{V}$. We then have the following,

$$\begin{aligned}
\left(\mathbf{I}_N + \frac{1}{d}\mathbf{KA}\right)\mathbf{V}\boldsymbol{r} &= 0, && (5.24)\\
\left(\mathbf{I}_N + \mathbf{KV\Gamma V}^{\mathsf{H}}\right)\mathbf{V}\boldsymbol{r} &= 0, && (5.25)\\
\mathbf{V}\boldsymbol{r} &= -\mathbf{KV\Gamma}\boldsymbol{r}, && (5.26)\\
||\mathbf{V}\boldsymbol{r}||^2 &= ||\mathbf{KV\Gamma}\boldsymbol{r}||^2, && (5.27)\\
\sum_{i=0}^{N-1}(1-\gamma_i^2)r(i)^2 &= 0, && (5.28)
\end{aligned}$$

where γ_i is the i^{th} eigenvalue of $\frac{1}{d}\mathbf{A}$. Gershgorin's Circle Theorem [40] stipulates that $|\gamma_i| \leq 1$, so it must be that $|r(i)| > 0$ only if $|\gamma_i| = 1$, and $r(i) = 0$ otherwise. Thus, $\boldsymbol{z}$ takes the form $\boldsymbol{z} = \frac{r(0)}{\sqrt{N}}\mathbf{1}_N + \widetilde{\mathbf{V}}\widetilde{\boldsymbol{r}}$, where $\widetilde{\mathbf{V}}$ is the set of eigenvectors corresponding to the eigenvalue -1 of the adjacency matrix.

We now consider two cases—one where the graph is bipartite and the other where the graph is non-bipartite. Let us start with the non-bipartite case which is easier.

Case (a) Non-bipartite graphs:
It is known that $\gamma > -1$ for non-bipartite graphs [41]. Thus $\boldsymbol{z}$ takes the form $\boldsymbol{z} = \frac{r(0)}{\sqrt{N}}\mathbf{1}_N$. This gives us the following,

$$\begin{aligned}
\left(\mathbf{I}_N + \frac{1}{d}\mathbf{KA}\right)\boldsymbol{z} &= \mathbf{0}_N, && (5.29)\\
\left(\mathbf{I}_N + \frac{1}{d}\mathbf{KA}\right)\frac{r(0)}{\sqrt{N}}\mathbf{1}_N &= \mathbf{0}_N, && (5.30)\\
\frac{r(0)}{\sqrt{N}}(\mathbf{1}_N + \mathbf{K}\mathbf{1}_N) &= \mathbf{0}_N. && (5.31)
\end{aligned}$$

As long as at least one of the diagonal entries of $\mathbf{K}$ is positive (i.e. at least one LP component is retained), we get that $r(0) = 0$ and hence $\boldsymbol{z} = 0$.

Case (b) Bipartite graphs:
In this case, it is known that if γ is an eigenvalue of $\mathbf{A}$, then $-\gamma$ is also an eigenvalue [41]. Let B be the indices of the nodes in one set of the bipartite graph. Then if $[\mathbf{v}_B; \mathbf{v}_{B^C}]$ is the eigenvector of γ, one can verify that $[\mathbf{v}_B; -\mathbf{v}_{B^C}]$ is the eigenvector of $-\gamma$. Thus for a connected graph, since the multiplicity of $\gamma = 1$ is one, the multiplicity of $\gamma = -1$ is also one and the corresponding

eigenvectors are $\mathbf{1}_N$ and $\left[\mathbf{1}_{\frac{N}{2}} \mid -\mathbf{1}_{\frac{N}{2}}\right]$. Using these properties we get that,

$$\frac{r(0)}{\sqrt{N}} \left(\mathbf{I}_N + \mathbf{K}\right) \mathbf{1}_N + r(1) \left(\mathbf{I}_N - \mathbf{K}\right) \left[\mathbf{1}_{\frac{N}{2}} \mid -\mathbf{1}_{\frac{N}{2}}\right] \quad = \quad \mathbf{0}_N,$$

where we are choosing $r(1)$ to be the coefficient associated with the eigenvector corresponding to $\gamma = -1$ without loss of generality. Note that $(\mathbf{I}_N + \mathbf{K})$ has zeros on the diagonals for all those elements that are discarded after sampling and $(\mathbf{I}_N - \mathbf{K})$ has zeros on the complementary set. Thus the above equation would require that $r(0) = 0$ and $r(1) = 0$, which again implies that $\boldsymbol{z} = 0$, completing the proof.

■

Remark: Note that the proof does not depend on any particular downsampling pattern and thus would work irrespective of the subset of nodes we choose for the low pass representation. The aliasing pattern however depends on the downsampling pattern.

The analysis LP and HP filters defined above are 1-hop filters. However synthesis filters are not necessarily one-hop. One can generalize these to k-hop filters by differencing and averaging over all neighbors up to k hops. The above proof holds since the adjacency matrix from before would instead consist of k-hop neighbors obtained by taking the k^{th} power of the adjacency matrix. These powers of the adjacency matrix are also circulant. By increasing the number of hops, the spread of the filter in graph domain grows, and the spread in the frequency domain shrinks. The k-hop simple Spline filters can be described as follows,

$$\mathbf{H}_{\text{LP,k}} \quad = \quad \frac{1}{2}\left(\mathbf{I}_N + \frac{1}{d_k}\mathbf{A}_k\right), \tag{5.32}$$

$$\mathbf{H}_{\text{HP,k}} \quad = \quad \frac{1}{2}\left(\mathbf{I}_N - \frac{1}{d_k}\mathbf{A}_k\right), \tag{5.33}$$

where $\mathbf{A}_k$ is the adjacency matrix of the k-hop graph given by $\mathbf{A}_k = \mathcal{I}_{\mathbf{A}^\mathbf{k}}$, where the modified indicator operator $\mathcal{I}$ is defined in Eqn 5.2.

Corollary 11. *For a connected circulant graph, the spline filters defined in Eqns. 5.32 and 5.33 form a critically-sampled perfect-reconstruction LSI filterbank for any downsampling pattern as long as at least one low-pass component is retained and the complementary set of nodes retain the high-pass component.*

Proof: The proof follows from that of Theorem 10 since the new matrix $\mathbf{A}_k$ is also a circulant adjacency matrix. ■

5.4.2 Generalized weighted Spline-like filter banks

To achieve greater control over the frequency response, one option is to vary the filter's averaging weights. It is well known that a circulant matrix $\mathbf{A}$ can be expressed as,

$$\mathbf{A} \;=\; \sum_{\ell=0}^{N-1} a(\ell)\,\mathbf{P}^{\ell}, \tag{5.34}$$

where $\boldsymbol{a} = [a(0), a(1), ..., a(N-1)]^{\mathsf{T}}$ is the first column of $\mathbf{A}$ (see Proposition 8). Define a new matrix $\widetilde{\mathbf{A}}$ as follows,

$$\widetilde{\mathbf{A}} \;=\; \sum_{\ell=0}^{N-1} w(\ell)a(\ell)\,\mathbf{P}^{\ell}, \tag{5.35}$$

where $\boldsymbol{w} = [w(0), w(1), ..., w(N-1)]^{\mathsf{T}}$ is a set of weights with $w(\ell) \geq 0$ and $\widetilde{d} = \sum_{\ell=0}^{N-1} w(\ell) > 0$. Note that $\widetilde{\mathbf{A}}$ is not symmetric in general, but it is circulant. The LP and HP filters are now defined by replacing the adjacency matrix $\frac{1}{d}\mathbf{A}$ in Eqns. 5.12 and 5.15 with $\frac{1}{\widetilde{d}}\widetilde{\mathbf{A}}$.

$$\widetilde{\mathbf{H}}_{\mathsf{LP}} \;=\; \frac{1}{2}\left(\mathbf{I}_N + \frac{1}{\widetilde{d}}\widetilde{\mathbf{A}}\right), \tag{5.36}$$

$$\widetilde{\mathbf{H}}_{\mathsf{HP}} \;=\; \frac{1}{2}\left(\mathbf{I}_N - \frac{1}{\widetilde{d}}\widetilde{\mathbf{A}}\right). \tag{5.37}$$

Note that since $\widetilde{\mathbf{A}}$ is a circulant matrix, it is diagonalizable by the GFT matrix. Hence we can compute the filter responses as follows.

$$\widetilde{\mathbf{H}}_{\mathsf{LP}} \;=\; \frac{1}{2}\mathbf{U}\left(\mathbf{I}_N + \frac{1}{\widetilde{d}}\mathrm{diag}\left(\mathbf{W}'^{\mathsf{G}}\right)\right)\mathbf{U}^{\mathsf{H}}, \tag{5.38}$$

$$\widetilde{\mathbf{H}}_{\mathsf{HP}} \;=\; \frac{1}{2}\mathbf{U}\left(\mathbf{I}_N - \frac{1}{\widetilde{d}}\mathrm{diag}\left(\mathbf{W}'^{\mathsf{G}}\right)\right)\mathbf{U}^{\mathsf{H}}, \tag{5.39}$$

$$\widetilde{H}^{\mathsf{G}}_{\mathsf{LP}}(k) \;=\; \frac{1}{2}\left(1 + \frac{1}{\widetilde{d}}W'^{\mathsf{G}}(k)\right), \tag{5.40}$$

$$\widetilde{H}^{\mathsf{G}}_{\mathsf{HP}}(k) \;=\; \frac{1}{2}\left(1 - \frac{1}{\widetilde{d}}W'^{\mathsf{G}}(k)\right), \tag{5.41}$$

where $\mathbf{W}'^{\mathsf{G}}$ is the GFT of the vector $\boldsymbol{w}' = (\boldsymbol{w} \circ \boldsymbol{a})$. The weights can be suitably optimized to approximate a desired frequency response depending on the application of interest. For example, one could minimize a least squares cost function to approximate a desired response $\mathbf{H}^{\mathsf{G}}_{\mathsf{LP\,des}}$ as follows,

$$\min_{\boldsymbol{w}} \quad \left|\left|\mathbf{H}^{\mathsf{G}}_{\mathsf{LP\ des}} - \widetilde{\mathbf{H}}^{\mathsf{G}}_{\mathsf{LP}}\right|\right|^2, \tag{5.42}$$

$$\text{s.t} \quad \boldsymbol{w}^{\mathsf{T}}\mathbf{1_N} > \mathbf{0}, \tag{5.43}$$

$$w(i) \geq 0, \quad \forall i = 0, \cdots, N-1. \tag{5.44}$$

Note that the optimization problem is convex once we impose the structure of $\boldsymbol{a}$ in the problem and null out the coefficients of $\boldsymbol{w}$ corresponding to the zero values of $\boldsymbol{a}$.

Theorem 12. *For a connected circulant graph, the generalized spline filters defined in Eqns 5.36 and 5.37 form a critically-sampled, perfect-reconstruction, LSI filterbank for every downsampling pattern that retains an equal number of low-pass and high-pass components in complementary node sets.*

Proof: By Gershgorin circle theorem, the magnitude of the eigenvalues of $\frac{1}{\tilde{d}}\widetilde{\mathbf{A}}$ are bounded by unity. Hence the proof is similar to that of Theorem 10. Although the matrix $\widetilde{\mathbf{A}}$ need not be symmetric, it is circulant and therefore diagonalized by the Fourier matrix. Abusing the notation for simplicity, let $\frac{1}{\tilde{d}}\widetilde{\mathbf{A}} = \mathbf{V}\boldsymbol{\Gamma}\mathbf{V}^{\mathsf{H}}$. Following along the same lines of the proof of Theorem 10, we get that $|r(i)| \geq 0$ if and only if $|\gamma_i| = 1$. Let $\boldsymbol{z} = \mathbf{V}\boldsymbol{r}$ lie in the null space of $\left(\mathbf{I}_N + \frac{1}{\tilde{d}}\mathbf{K}\widetilde{\mathbf{A}}\right)$. Then $\boldsymbol{z}$ takes the form,

$$\boldsymbol{z} \quad = \quad \widehat{\mathbf{V}}\widehat{\boldsymbol{r}} + \widetilde{\mathbf{V}}\widetilde{\boldsymbol{r}}, \tag{5.45}$$

where $\widehat{\mathbf{V}}$ are the eigenvectors corresponding to the eigenvalue unity, $\widetilde{\mathbf{V}}$ are the eigenvectors corresponding to the eigenvalue -1. Further, $\widehat{\boldsymbol{r}}$ and $\widetilde{\boldsymbol{r}}$ are the non-zero entries of $\boldsymbol{r}$ corresponding to $\widehat{\mathbf{V}}$ and $\widetilde{\mathbf{V}}$ respectively. Note that the multiplicity of the eigenvalue 1 need not be unity since the matrix $\widetilde{\mathbf{A}}$ is not the adjacency matrix of an undirected graph.

Using this we get the following,

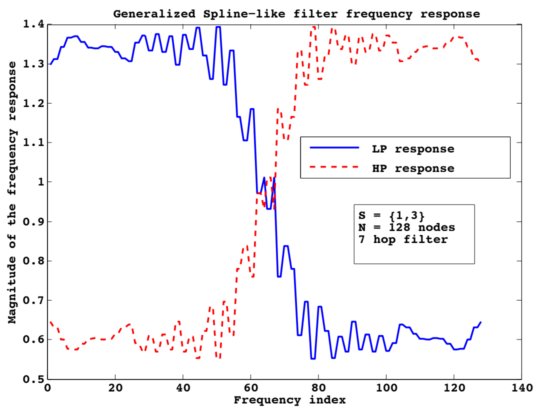
$$\left(\mathbf{I}_N + \frac{1}{\tilde{d}}\mathbf{K}\widetilde{\mathbf{A}}\right)\left(\widehat{\mathbf{V}}\widehat{\boldsymbol{r}} + \widetilde{\mathbf{V}}\widetilde{\boldsymbol{r}}\right) \quad = \quad \mathbf{0}, \tag{5.46}$$

$$\left(\mathbf{I}_N + \mathbf{K}\right)\widehat{\mathbf{V}}\widehat{\boldsymbol{r}} + \left(\mathbf{I}_N - \mathbf{K}\right)\widetilde{\mathbf{V}}\widetilde{\boldsymbol{r}} \quad = \quad \mathbf{0}. \tag{5.47}$$

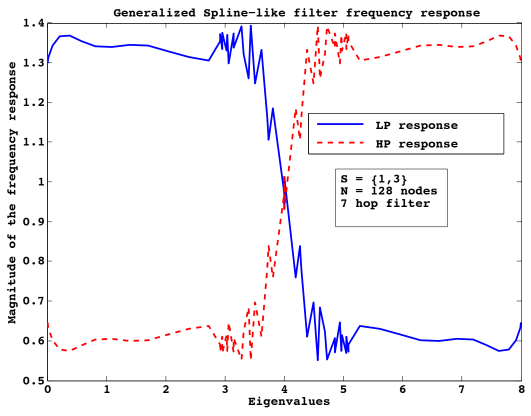
Note that $(\mathbf{I}_N + \mathbf{K})$ has zeros on the diagonals for all those elements that are discarded after sampling and $(\mathbf{I}_N - \mathbf{K})$ has zeros on the complementary set. Let α be the set of nodes that are retained and α^C be the complementary node set. We then get the following,

$$\widehat{\mathbf{V}}_{\alpha}\widehat{\boldsymbol{r}} \quad = \quad \mathbf{0}, \tag{5.48}$$

$$\widetilde{\mathbf{V}}_{\alpha^C}\widetilde{\boldsymbol{r}} \quad = \quad \mathbf{0}, \tag{5.49}$$



(a)



(b)

Figure 5.7: Example of a generalized spline-like filter bank response for a circulant graph with $N = 128$ and $S = \{1, 3\}$. The LP and HP filter responses are shown in the figures. Fig (a) shows the response plotted as a function of the frequency index. The response is obtained for a 7 hop filter where the coefficients are optimized to approximate an ideal LP response with cutoff frequency $N/2$. One observes considerable ripples in the transition band. This could be reduced by introducing a penalty function in the optimization routine.The frequency response plotted as a function of the eigenvalues as in Fig (b), seems to exhibit a sharper cutoff.

where $\widehat{\mathbf{V}}_\alpha$ and $\widetilde{\mathbf{V}}_{\alpha^C}$ are the sub-matrices of $\widehat{\mathbf{V}}$ and $\widetilde{\mathbf{V}}$ with the rows retained corresponding to the sets α and α^C respectively. Note that $|\alpha| = |\alpha^C| = N/2$ from the assumption in the theorem. In order to have non-trivial solutions for $\widehat{\boldsymbol{r}}$ and $\widetilde{\boldsymbol{r}}$, the column dimension of $\widehat{\mathbf{V}}_\alpha$ or $\widetilde{\mathbf{V}}_{\alpha^C}$ needs to be greater than $N/2$. This is because, $\mathbf{V}$ is a Vandermonde matrix and any sub-matrix of $\mathbf{V}$ has full-rank. Thus, for this condition to hold, the multiplicity of $\gamma = 1$ or $\gamma = -1$ need to be greater than $N/2$. However, this is not possible since the trace of $\widetilde{\mathbf{A}} = 0$ which is equal to the sum of the eigenvalues and we know that the magnitude of the eigenvalues are bounded by unity.

Thus we have $\widehat{\boldsymbol{r}} = 0$ and $\widetilde{\boldsymbol{r}} = 0$ and hence $\boldsymbol{z} = 0$.

■

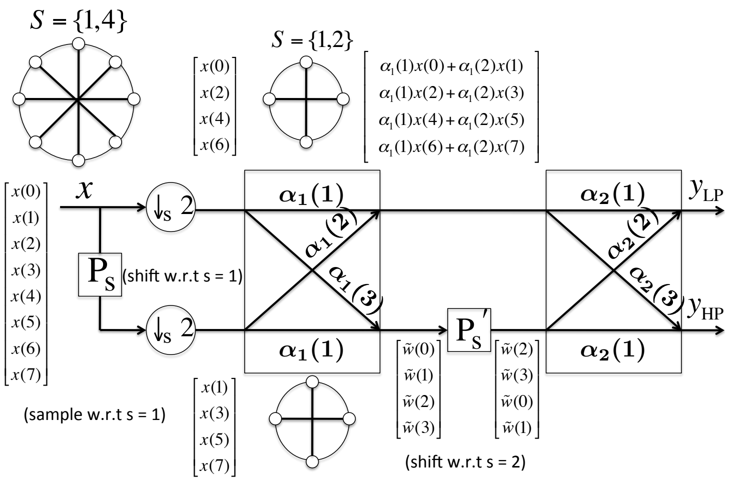
Remark: Compared to the unweighted Spline filters, here we need to keep $N/2$ components of the lowpass and the high pass filter responses. Note that this is only a sufficient condition.

Fig 5.7 shows an example of the frequency responses for a circulant graph with $N = 128$ and $S = \{1, 3\}$. The filter designed is a 7 hop filter, with the weights optimized to approximate an ideal LP response with cutoff frequency $N/2$. One observes considerable ripples in the transition band. This could be reduced by introducing a penalty function in the optimization routine if required. As before, one again observes that the response plotted as a function of the eigenvalues is intuitively pleasing and seems to exhibit a sharper cutoff.

5.4.3 Properties of the Spline-like two-channel filter bank

The Spline-like filter banks defined in the previous two sections satisfy some of the desirable properties that we had listed out for filter banks. These are detailed below,

- The filter banks satisfy the critical-sampling and perfect reconstruction properties as asserted in Theorems 10 and 12.
- The filters are localized in the graph domain given their explicit representation as a function of the adjacency matrix. As we move to higher hops of the adjacency matrix, we would get a better localization in the GFT domain as opposed to that in the graph domain.
- The filters are LSI on circulant graphs by design.
- The filters are graph-independent since the form of the filter is the same for all graphs. The generalized filters are not graph-independent since the weights need to be optimized for the given graph. However for fixed weights, they are graph-independent.
- The inverse filters are not necessarily localized in nature.
- The filters are not orthogonal in general. Note that one could use the modified orthogonalization procedure proposed in [42] that retains the local support of the filters. However, the theoretical guarantees on the perfect reconstruction would no longer be valid.



$S = \{1,4\}$
$S = \{1,2\}$
$x(0)$ $x(2)$ $x(4)$ $x(6)$
$\alpha_1(1)x(0)+\alpha_1(2)x(1)$
$\alpha_1(1)x(2)+\alpha_1(2)x(3)$
$\alpha_1(1)x(4)+\alpha_1(2)x(5)$
$\alpha_1(1)x(6)+\alpha_1(2)x(7)$
$x(0)$ $x(1)$ $x(2)$ $x(3)$ $x(4)$ $x(5)$ $x(6)$ $x(7)$
x
$\downarrow_S 2$
P_S
(shift w.r.t s = 1)
$\alpha_1(1)$
$\alpha_1(2)$
$\alpha_1(3)$
$\alpha_2(1)$
$\alpha_2(2)$
$\alpha_2(3)$
y_{LP}
y_{HP}
P_S'
$\tilde{w}(0)$ $\tilde{w}(1)$ $\tilde{w}(2)$ $\tilde{w}(3)$
$\tilde{w}(2)$ $\tilde{w}(3)$ $\tilde{w}(0)$ $\tilde{w}(1)$
(sample w.r.t s = 1)
$x(1)$ $x(3)$ $x(5)$ $x(7)$
(shift w.r.t s = 2)

Figure 5.8: Lattice structure based perfect reconstruction filter banks for circulant graphs with an example. The coefficients at each stage are chosen such that the transform is invertible and optimized to approximate a desired frequency response. The shift and sampling operators depend on the graph and can be chosen accordingly.

5.5 Lattice filter bank structures

Lattice structures in classical filter bank theory are a class of two-channel filter bank structures that are popular due to their simple implementation structure and a host of other advantages such as linear-phase [37]. The structure of the filter bank guarantees critical sampling as well as perfect reconstruction. For time-signals, the design involves dividing the signal into even and odd components and taking invertible linear combinations of the values across the two branches. We adopt a similar design philosophy for designing Lattice filter banks for circulant graph signals.

Fig. 5.8 shows the lattice structure framework for circulant graphs with an example. The downsample and shift operators are as defined on the circulant graph and depend on the element $s \in S$. The two branches of the filter in the first stage keep complementary copies of the input signal depending on the element s in S, according to which the shifts and downsampling operations are defined. After the downsampling operation in the first stage, the shifts in the subsequent stages are defined on the downsampled graph.

As long as each of the blocks in the lattice structure is invertible, the overall filter bank is invertible. The block diagram of the filter bank including the analysis and synthesis stages are shown in Fig 5.9. One way to guarantee invertibility is to fix $\alpha_j(1) = 1$ and $\alpha_j(3) = -\alpha_j(2)$ at stage j. This reduces the number of parameters to be optimized, and the filters are now orthogonal; the filter

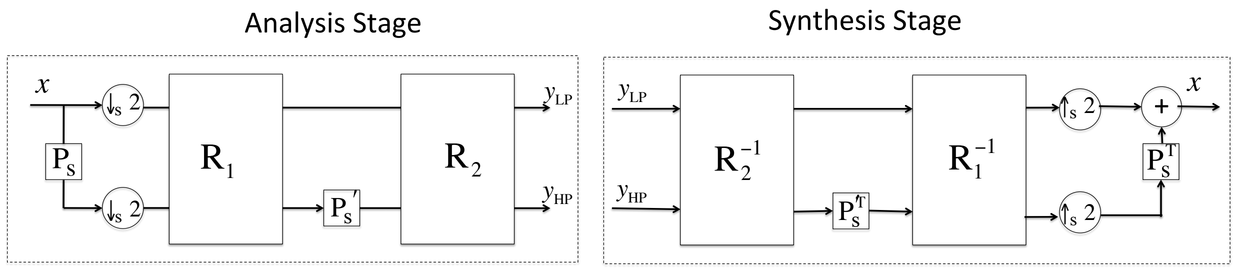


Figure 5.9: Block diagram description of the analysis and synthesis stages for the Lattice filter. Each of the stages in the filter bank is chosen to be invertible and hence the synthesis stage mimics the analysis stage.

is shift invariant for a fixed choice of coefficients. The number of stages used would define the number of hops for the filter.

The filter coefficients for the Lattice structure can be calculated in closed form once the number of stages and the shifts are fixed. For example, the filter responses using a 8-node circulant graph in Fig 5.8 can be calculated to be the following assuming the shifts and sampling are with respect to $s = 1$,

$$\boldsymbol{h}_{\text{LP}} = \begin{bmatrix} \alpha_1(1)\alpha_2(1) \\ \alpha_1(2)\alpha_2(1) \\ 0 \\ 0 \\ 0 \\ 0 \\ \alpha_1(3)\alpha_2(2) \\ \alpha_1(1)\alpha_2(2) \end{bmatrix}, \tag{5.50}$$

$$\boldsymbol{h}_{\text{HP}} = \begin{bmatrix} \alpha_1(1)\alpha_2(3) \\ \alpha_1(2)\alpha_2(3) \\ 0 \\ 0 \\ 0 \\ 0 \\ \alpha_1(3)\alpha_2(1) \\ \alpha_1(1)\alpha_2(1) \end{bmatrix}. \tag{5.51}$$

The filter responses can now be optimized over the coefficients $\alpha()$ to approximate a desired response. Note that we need to add a constraint in the optimization such that the matrices are invertible for perfect reconstruction to hold (see Fig 5.9). Further, the optimization is not convex

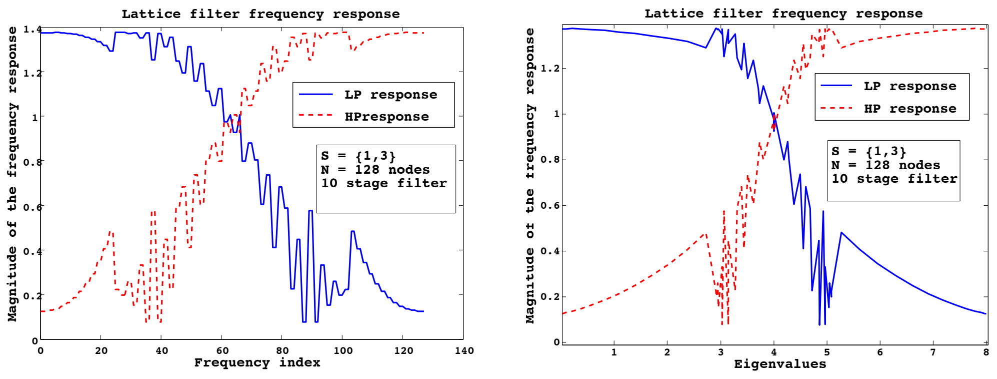


Figure 5.10: Example response of a 10-stage lattice filter designed over a circulant graph with 128 nodes generated by the set $S = \{1, 3\}$. The downsampling and shifting in all the stages is taken with respect to $s = 1$.

given the form of the filter responses. Hence approximation techniques are needed to solve for the coefficients.

Fig. 5.10 shows an example filter response for a circulant graph defined by $N = 128$ and $S = \{1, 3\}$. The number of stages in the filter is taken to be 10. The coefficients are optimized to approximate an ideal LP filter response. A least squares cost function is optimized using a gradient-descent algorithm and the solution so obtained is one of the local minima. The response is shown as a function of both frequency indices and graph eigenvalues. Downsampling and shifting in all stages is taken with respect to $s = 1$. The LP and HP responses appear to be complementary in this case. However, note that this need not be the case in general. Depending on the choice of the coefficients one can obtain different sets of responses for LP and HP. Hence the optimization problem needs to be carefully formulated. We do not explore this here.

Fig 5.11 shows an example of a multiscale decomposition of a circulant graph signal into two stages using Lattice filters. The filter coefficients are optimized to approximate ideal LP and HP responses. Qualitatively the LP branch captures the information in the original signal of interest. The HP output captures the signal variations at the boundary of transition where the signal value changes from unity to zero. The downsampling is carried out with respect to $s = 1$ and *Kron* reduction is employed to define the downsampled graph.

Following are some interesting properties of the Lattice filter banks.

- The filter bank satisfies the critical-sampling and perfect-reconstruction properties by design. However, note that the optimization of the filter coefficients is tricky.

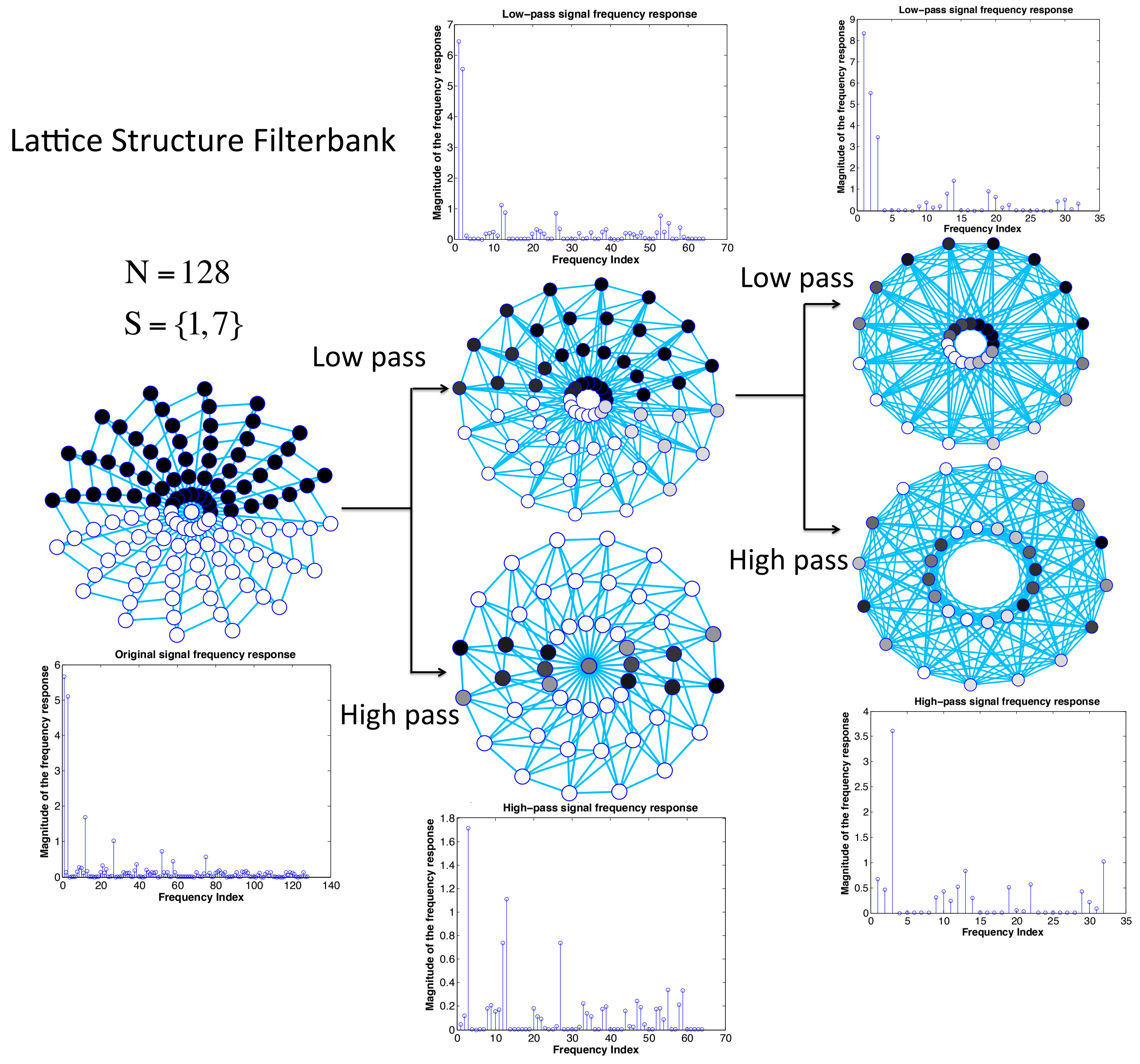

Lattice Structure Filterbank
N = 128
S = {1, 7}
Low pass
High pass
Low pass
High pass
Original signal frequency response
Low−pass signal frequency response
High−pass signal frequency response
Low−pass signal frequency response
High−pass signal frequency response
Magnitude of the frequency response
Frequency Index

Figure 5.11: Two stage decomposition of a circulant graph signal with $N = 128$ and $S = \{1, 7\}$ using a 10-stage Lattice filter. The downsampling is carried out with respect to $s = 1$ and *Kron*-reduction is employed to define the downsampled graph. The filter coefficients are optimized to approximate ideal responses.

- The filter banks are localized in the graph domain. The number of stages used defines the spread of the filter response in the graph domain.
- The filters are also LSI for circulant graphs once the coefficients are fixed since these remain the same for every shift.
- The filters are not necessarily graph-independent. The downsampling and shift matrices need to be defined depending on the graph and the weights have to be accordingly optimized.
- The inverse filters are localized by design.
- The filters can be chosen to be orthogonal by placing further restrictions on the coefficients $\alpha()$ at each stage.

5.6 Chapter highlights

Following is a summary of the main points of this chapter.

- Two channel graph filter banks lay the foundations for multi resolution analysis by successively decomposing the input graph signal into different frequency bands.
- The important desirable properties of filter banks are critical sampling, perfect reconstruction, graph-domain GFT-domain localization, orthogonality, local reconstruction and graph-invariance.
- Three filter bank structures with varying properties are discussed—The Graph Laplacian Pyramid, Spline-like filter banks, Lattice structure filter banks.
- The Graph Laplacian Pyramid offers considerable flexibility in the filter design at the cost of oversampling.
- Spline-like filter banks are critically sampled and perfect reconstruction filter banks that also offer some flexibility in the design and satisfy most of the desirable properties except local-reconstruction.
- Lattice filter structure are also critically-sampled perfect-reconstruction filter banks that also offer flexibility in their design. However the optimization of the filter coefficients is tricky and they are not graph-independent.

Part III

Signal Processing on General Graphs

Chapter 6

Circulant decomposition of a general graph

6.1 Introduction

In the previous chapters, we discussed circulant graphs at length with emphasis on fundamental signal processing operations and filter bank designs. One of the major advantages of circulant graphs is that the graph is symmetric with respect to different nodes. Hence it is possible to define fundamental operations like shifting, convolution, and sampling. This enables design of LSI filters on these graphs analogous to LTI filters in the time domain.

Some real-world network models such as small-world networks (Watts-Strogatz model [23]) are inspired by circulant graphs. However, for most other real-world applications we usually encounter graphs that might not have as much structure as circulant graphs. Thus we need to extend our signal analysis and filter design tools to general graphs. General graphs being asymmetric, one cannot have LSI filters as defined in Section 5.1. They are only amenable to Linear Shift Varying (LSV) signal processing. Linear Time Varying (LTV) filters in classical signal processing are designed by decomposition into a bank of LTI filters [14]. Analogously, our approach is to decompose a general graph into a bank of circulant graphs and extend the filtering and sampling operations from circulant graphs to general graphs.

In this chapter, we discuss various decompositions of a general graph into circulant graphs. In particular, we shall represent the Laplacian matrix of a general graph as a suitable sum of the Laplacian matrices of circulant graphs. The decomposition is motivated by the following fact that a circulant matrix $\mathbf{C}$ can be expressed as follows,

$$\mathbf{C} = \sum_{j=0}^{N-1} c(j)\mathbf{P}^j, \tag{6.1}$$

where $\boldsymbol{c} = [c(0), c(1), ..., c(N-1)]^\mathsf{T}$ is the first column of $\mathbf{C}$, and $\mathbf{P}$ is the shift matrix defined in

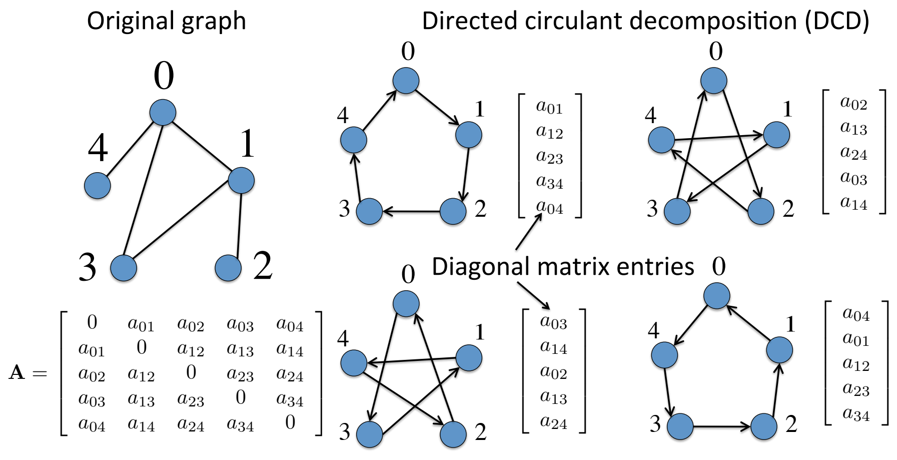

Original graph
Directed circulant decomposition (DCD)
Diagonal matrix entries
0
1
2
3
4
$\mathbf{A} = \begin{bmatrix} 0 & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{01} & 0 & a_{12} & a_{13} & a_{14} \\ a_{02} & a_{12} & 0 & a_{23} & a_{24} \\ a_{03} & a_{13} & a_{23} & 0 & a_{34} \\ a_{04} & a_{14} & a_{24} & a_{34} & 0 \end{bmatrix}$
$\begin{bmatrix} a_{01} \\ a_{12} \\ a_{23} \\ a_{34} \\ a_{04} \end{bmatrix}$
$\begin{bmatrix} a_{02} \\ a_{13} \\ a_{24} \\ a_{03} \\ a_{14} \end{bmatrix}$
$\begin{bmatrix} a_{03} \\ a_{14} \\ a_{02} \\ a_{13} \\ a_{24} \end{bmatrix}$
$\begin{bmatrix} a_{04} \\ a_{01} \\ a_{12} \\ a_{23} \\ a_{34} \end{bmatrix}$

Figure 6.1: Example of the directed circulant decomposition (DCD) of a given graph. Each of the directed circulant graphs correspond to different powers of the shift matrix $\mathbf{P}$. The diagonal matrices correspond to each of the rows of the original Laplacian matrix.

Equation 3.2:

$$\mathbf{P} \;=\; \begin{bmatrix} \mathbf{0}_{N-1}^{\mathsf{T}} & 1 \\ \mathbf{I}_{N-1} & \mathbf{0}_{N-1} \end{bmatrix}. \tag{6.2}$$

$\mathbf{P}$ can also be viewed as the adjacency matrix of a directed cycle graph. Thus, in essence, we are "decomposing" a circulant graph into many directed cycle graphs. Analogously one can ask if the Laplacian matrix of a general graph has such a decomposition into circulant Laplacian matrices[1]. We show that such a decomposition is feasible with the scalar factors replaced by diagonal matrices. Such a decomposition is not unique unless other restrictions are imposed. We show three such decompositions in the sections to follow.

6.2 Directed Circulant Decomposition

Here we will express the Laplacian matrix of a general graph as an appropriately defined sum of Laplacian matrices of directed circulant graphs. The following theorem illustrates this decomposition.

[1]The terminology, graph decomposition, is also classically used in the graph theory literature to refer to the problem of partitioning the edges of a given graph into different sets such that the edges in each set correspond to a graph that is isomorphic to a given graph. For example, each set could be isomorphic to a given circulant graph. However this is known to be a NP-complete problem [43].

Theorem 13. *(*Directed Circulant Decomposition (DCD):*) The Laplacian matrix* $\mathbf{L}$ *of the given graph can be written as,*

$$\mathbf{L} = \sum_{k=1}^{N-1} \mathbf{Q}_k \mathbf{L}_k, \tag{6.3}$$

where $\mathbf{L}_k = \mathbf{I} - \mathbf{P}^{N-k}$ *and* $\mathbf{Q}_k = -\mathrm{diag}(\mathbf{L}\mathbf{P}^k) = \mathrm{diag}(\mathbf{A}\mathbf{P}^k)$.

Proof: We will start by analyzing the adjacency matrix and then extend the analysis to the Laplacian matrix. Let us look for a decomposition of the following form for the adjacency matrix of a given graph:

$$\mathbf{A} = \sum_{k=0}^{N-1} \mathbf{D}_k \mathbf{A}_k, \tag{6.4}$$

where $\mathbf{D}_k$'s are diagonal matrices of the form,

$$\mathbf{D}_k = \mathrm{diag}\left([d_k(0), d_k(1), ..., d_k(N-1)]\right) = \mathrm{diag}\left(\boldsymbol{d}_k^{\mathsf{T}}\right), \tag{6.5}$$

and $\mathbf{A}_k$'s are circulant adjacency matrices of the form,

$$\mathbf{A}_k = \mathrm{circ}\left([a_{k,0}(0), a_{k,0}(1), ..., a_{k,0}(N-1)]\right) = \mathrm{circ}\left(\boldsymbol{a}_{k,0}^{\mathsf{T}}\right), \tag{6.6}$$

where $\boldsymbol{a}_{k,0}^{\mathsf{T}}$ is the first row of the circulant matrix $\mathbf{A}_k$ and the m^{th} row of $\mathbf{A}_k$ is $\boldsymbol{a}_{k,0}^{\mathsf{T}}\mathbf{P}^{m-1^{\mathsf{T}}}$. We deviate from the convention of using the columns to represent the circulant matrix and use the rows since that simplifies the argument for the proof. Each row of the adjacency matrix can be expressed as follows.

$$\boldsymbol{a}_0^{\mathsf{T}} = [d_0(0)\ d_1(0)\ , ..., d_{N-1}(0)] \begin{bmatrix} \boldsymbol{a}_{0,0}^{\mathsf{T}} \\ \boldsymbol{a}_{1,0}^{\mathsf{T}} \\ \cdots \\ \boldsymbol{a}_{N-1,0}^{\mathsf{T}} \end{bmatrix}, \tag{6.7}$$

$$= \boldsymbol{d}(0)^{\mathsf{T}}\widetilde{\mathbf{A}}, \tag{6.8}$$

where $\widetilde{\mathbf{A}} = \begin{bmatrix} \boldsymbol{a}_{0,0}^{\mathsf{T}} \\ \boldsymbol{a}_{1,0}^{\mathsf{T}} \\ \cdots \\ \boldsymbol{a}_{N-1,0}^{\mathsf{T}} \end{bmatrix}$, is the collection of the first rows of all the circulant adjacency matrices in the decomposition. Similarly it is easy to see that the m^{th} row is given by,

$$\begin{aligned} \boldsymbol{a}_{m-1}^{\mathsf{T}} &= \boldsymbol{d}(m-1)^{\mathsf{T}}\,\widetilde{\mathbf{A}}\,\mathbf{P}^{m-1^{\mathsf{T}}}, && (6.9) \\ \boldsymbol{a}_{m-1}^{\mathsf{T}}\mathbf{P}^{m-1} &= \boldsymbol{d}(m-1)^{\mathsf{T}}\widetilde{\mathbf{A}}, && (6.10) \end{aligned}$$

where $\boldsymbol{d}(m-1)$ is a vector of the m^{th} diagonal elements of each of the diagonal matrices in the decomposition. The above set of $m-1$ equations can be written in a matrix form as follows,

$$\begin{bmatrix} \boldsymbol{a}_0^{\mathsf{T}} \\ \boldsymbol{a}_1^{\mathsf{T}}\mathbf{P} \\ \cdots \\ \boldsymbol{a}_{N-1}^{\mathsf{T}}\mathbf{P}^{N-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}(0)^{\mathsf{T}} \\ \boldsymbol{d}(1)^{\mathsf{T}} \\ \cdots \\ \boldsymbol{d}(N-1)^{\mathsf{T}} \end{bmatrix} \widetilde{\mathbf{A}}. \qquad (6.11)$$

As long as we have an invertible $\widetilde{\mathbf{A}}$, we can get a valid solution for the diagonal matrices. The simplest choice for $\widetilde{\mathbf{A}}$ is the identity matrix i.e., $\widetilde{\mathbf{A}} = \mathbf{I}_N$. We then have,

$$\boldsymbol{d}(m-1)^{\mathsf{T}} = \boldsymbol{a}_{m-1}^{\mathsf{T}}\mathbf{P}^{m-1}. \qquad (6.12)$$

One can easily verify that these equations can be rearranged to get,

$$\mathbf{D}_k = \text{diag}(\mathbf{A}\mathbf{P}^k). \qquad (6.13)$$

Since the diagonals of the adjacency matrix are zero, we have that $\boldsymbol{d}_0(m) = 0,\ \forall\, m$. Hence the first term in the summation has $\boldsymbol{d}_0 = 0$. Let us now consider the k^{th} term in the summation. Here $\mathbf{A}_k$ is a circulant matrix with the first row equal to the k^{th} row of $\widetilde{\mathbf{A}} = \mathbf{I}$. Hence, we have that $\mathbf{A}_k = \mathbf{P}^{N-k}$ which is a valid adjacency matrix. Note that since the first term is zero, we need not bother about $\mathbf{A}_0$ not being a valid adjacency matrix.

It is now easy to extend this to the Laplacian matrix. Note that we have $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal matrix of the degrees of each of then nodes. Also note that we have,

$$\sum_{m=1}^{N-1} \mathbf{d}(m) = D(m,m). \qquad (6.14)$$

Thus we get that,

$$\begin{aligned} \mathbf{L} &= \mathbf{D} - \mathbf{A}, && (6.15) \\ &= \mathbf{D} - \sum_{k=1}^{N-1} \mathbf{D}_k \mathbf{P}^{N-k}, && (6.16) \\ &= \sum_{k=1}^{N-1} \mathbf{D}_k (\mathbf{I} - \mathbf{P}^{N-k}), && (6.17) \\ &= \sum_{k=1}^{N-1} \mathbf{Q}_k \mathbf{L}_k. && (6.18) \end{aligned}$$

■

Fig. 6.1 shows an example of this decomposition with each of the circulant graphs and the associated diagonal matrices. One of the advantages of this decomposition is that each of the individual circulant graphs are independent of the given graph and hence can be precomputed and stored.The drawback of the above decomposition is that each of the circulant graphs in the decomposition is a directed graph. Unfortunately, the theory we have developed so far has been for undirected graphs. It is not yet clear on how to deal with directed graphs, which poses additional challenges.

It is thus natural to ask whether we can have a decomposition with undirected circulant graphs. If we restrict ourselves to decompositions where the summations contain a pre-multiplication by a diagonal matrix and a Laplacian matrix, we can show that it is never possible to decompose a general Laplacian matrix in terms of undirected circulant Laplacians.

Corollary 2. *If we restrict the decomposition of the Laplacian to the form*

$$\mathbf{L} = \sum_{k=1}^{M-1} \mathbf{Q}_k \mathbf{L}_k, \qquad (6.19)$$

where $\mathbf{Q}_k$ are diagonal matrices and $\mathbf{L}_k$ are circulant Laplacian matrices, then $\mathbf{L}_k$ cannot be symmetric (and hence undirected), even when $M \geq N$.

Proof: The proof follows from our derivation in the above theorem. Consider the following key step (Eqn 6.11) where we had the flexibility of choosing a circulant graph.

$$\begin{bmatrix} \boldsymbol{a}_0^\mathsf{T} \\ \boldsymbol{a}_1^\mathsf{T}\mathbf{P} \\ \cdots \\ \boldsymbol{a}_{N-1}^\mathsf{T}\mathbf{P}^{N-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}(0)^\mathsf{T} \\ \boldsymbol{d}(1)^\mathsf{T} \\ \cdots \\ \boldsymbol{d}(N-1)^\mathsf{T} \end{bmatrix} \widetilde{\mathbf{A}}. \qquad (6.20)$$

Note that this equation is fundamental for any decomposition with a pre-diagonal matrix and circulant matrix in the summations. For a valid decomposition, we needed an invertible $\widetilde{\mathbf{A}}$. Now suppose we restrict ourselves to undirected circulants, then each row of the matrix $\widetilde{\mathbf{A}}$ only has $N/2$ free entries, since the circulant matrices $\mathbf{A}_k$ need to be symmetric. Thus the rank of the matrix $\widetilde{\mathbf{A}}$ is at most $N/2$ and hence cannot be inverted. This is true irrespective of the number of terms we take in the summation as this would only increase the number of columns of the matrix $\widetilde{\mathbf{A}}$ while the rank is governed by the flexibility in the rows. ■

6.3 Undirected Circulant Decomposition

In order to have a decomposition in terms of undirected circulant graphs, we need to have additional flexibility in the form of the summation we have considered. Let us now introduce diagonal matrices that pre and post multiply every term in the summation. Under this decomposition structure, we consider different representations. In particular, we could have an over-complete representation with more than N terms in the summation or we can have a representation with N terms. The following sections illustrate these representations.

6.3.1 Overcomplete Undirected Circulant Decomposition

In this decomposition, we will consider more than N terms in the decomposition. The flexibility we gain would be that the circulant graphs so obtained are independent of the original graph and hence can be pre-computed. The following theorem illustrates the decomposition.

Theorem 14. *(*Overcomplete Undirected Circulant Decomposition (OUCD)*:) The Laplacian matrix* $\mathbf{L}$ *of the given graph can be written as,*

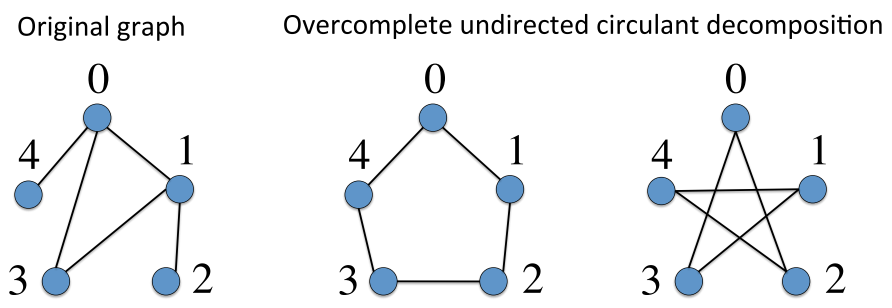
$$\mathbf{L} \;=\; \mathbf{D} + \overbrace{\sum_{k=0}^{N-1} \sum_{m=1}^{N/2-1} \mathbf{Q}_k \mathbf{L}_m \widetilde{\mathbf{Q}}_{k,m}}^{=-\mathbf{A}}, \tag{6.21}$$

where $\mathbf{L}_m = 2\mathbf{I} - \big(\mathbf{P}^m + (\mathbf{P}^m)^\mathsf{T}\big)$, $\mathbf{Q}_k$ *is a diagonal matrix with* $Q_k(k,k) = 1$, *and the rest of the entries are zeros.* $\mathbf{D} = \operatorname{diag}(\mathbf{L})$. $\widetilde{\mathbf{Q}}_{k,m}$ *is also a diagonal matrix with,*

$$\widetilde{Q}_{k,m}((k+m)_N,(k+m)_N) \;=\; L(k,(k+m)_N), \tag{6.22}$$

$$\widetilde{Q}_{k,m}((k-m)_N,(k-m)_N) \;=\; L(k,(k-m)_N). \tag{6.23}$$

Proof: It is straightforward to see that by pre and post multiplying a matrix by a diagonal matrix, we can pick every entry of the matrix. For example, if the first diagonal entry of the pre diagonal matrix is unity then the first row is picked. Similarly a column is picked by the post diagonal matrix. Thus all the N^2 entries of $\mathbf{L}$ can be individually picked by having N^2 terms in



$$\mathbf{A} = \begin{bmatrix} 0 & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{01} & 0 & a_{12} & a_{13} & a_{14} \\ a_{02} & a_{12} & 0 & a_{23} & a_{24} \\ a_{03} & a_{13} & a_{23} & 0 & a_{34} \\ a_{04} & a_{14} & a_{24} & a_{34} & 0 \end{bmatrix} \quad \tilde{\mathbf{Q}}_{0,1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & a_{01} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{04} \end{bmatrix} \quad \tilde{\mathbf{Q}}_{0,2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{02} & 0 & 0 \\ 0 & 0 & 0 & a_{03} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 6.2: Example showing the different component graphs in the Oversampled Undirected Circulant Decomposition of the given graph. Even though there can potentially be a large number of component graphs, only few of the terms in the summation are active if the original graph is sparse.

the summation. The number of terms can be reduced by recognizing that each row of the circulant graphs are symmetric since they are undirected.

Let us begin by choosing,

$$\mathbf{L}_m = 2\mathbf{I}_N - (\mathbf{P}^m + (\mathbf{P}^m)^{\mathsf{T}}). \tag{6.24}$$

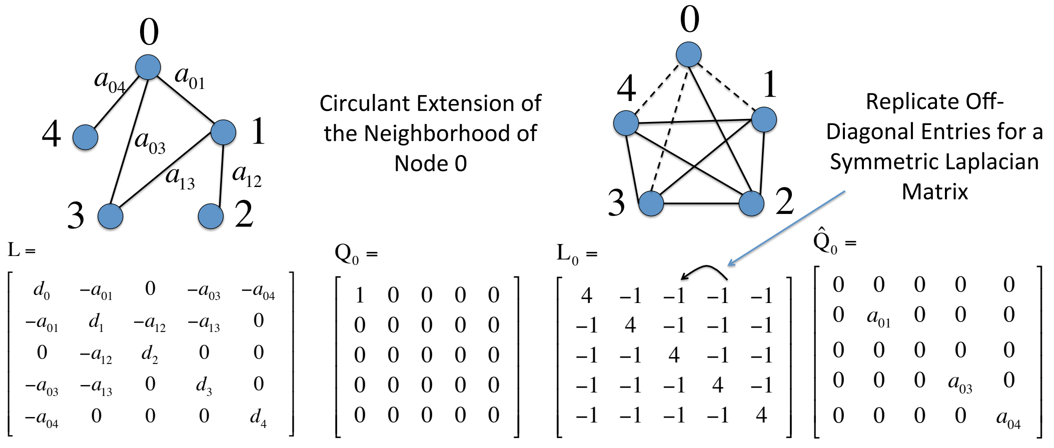
By construction $L_m(0, m) = L_m(0, N - m) = 1$ and the rest of the entries are zeros. By choosing,

$$\begin{aligned} \widetilde{Q}_0(0, m) &= L_0(0, m), \\ \widetilde{Q}_0(0, N - m) &= L_0(0, N - m), \end{aligned}$$

and $Q_0(0, 0) = 1$, the terms in the summation for $k = 0$, would essentially reconstruct the first row of $\mathbf{L}$, barring the diagonal entry. Similarly every other term in the summation reconstructs the corresponding row of $\mathbf{L}$. $\mathbf{D}$ is chosen to fill in the diagonal entries of the given Laplacian matrix.

■

Remark: Each of the $\mathbf{L}_m$'s are undirected circulant graphs of degree two. Further these graphs are independent of the given graph that needs to be decomposed. The number of circulant graphs in this decomposition grows quadratically with the size of the graph. However note that if the original graph is sparse, we would only have $O(Nd)$ non-zero terms in the summation where d is the average degree of the nodes in the graph.


Circulant Extension of the Neighborhood of Node 0
Replicate Off-Diagonal Entries for a Symmetric Laplacian Matrix

$$L = \begin{bmatrix} d_0 & -a_{01} & 0 & -a_{03} & -a_{04} \\ -a_{01} & d_1 & -a_{12} & -a_{13} & 0 \\ 0 & -a_{12} & d_2 & 0 & 0 \\ -a_{03} & -a_{13} & 0 & d_3 & 0 \\ -a_{04} & 0 & 0 & 0 & d_4 \end{bmatrix} \quad Q_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$L_0 = \begin{bmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{bmatrix} \quad \hat{Q}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & a_{01} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{03} & 0 \\ 0 & 0 & 0 & 0 & a_{04} \end{bmatrix}$$

Figure 6.3: Example of a circulant graph in the decomposition of the given graph. The circulant graph can be viewed as a circulant extension of the Node 0 in this graph. The diagonal matrices that pre and post multiply the circulant Laplacian matrix is also shown here.

Fig 6.2 shows an example of the OUCD decomposition for the given graph. In this example, one can see that all the component graphs belong to one of the two categories shown in the figure. Further, not all terms in the summation are active. The number of active terms is proportional to the number of edges in the graph and hence if the graph is sparse the decomposition is also sparse. Thus we see that even though there are potentially $O(N^2)$ terms in the summation only $O(Nd)$ are active for any given graph where d is the average degree of the graph. The natural question of interest is whether there exists a decomposition with only $O(N)$ terms? It turns out that this is possible, but the resulting circulant graphs are constructed depending on the given graph.

The key observation is that we can choose the circulant graphs in the summation. Each circulant graph offers $N/2$ degrees of freedom, which we are not fully exploiting in the OUCD representation. The degrees of freedom are calculated based on the fact that the first row of the circulant matrix completely defines the matrix. Since this matrix must be symmetric for the graph to be undirected, it is easy to see that the i^{th} and $(N-i)^{\text{th}}$ entries have to be equal for $i = 1, 2, ..., N-1$. In the sections to follow, we will discuss two decompositions. One that contains $N+1$ terms and the other with $2N+1$ terms. The decomposition with a larger number of terms helps extend arbitrary filtering operations from circulant graphs to general graphs.

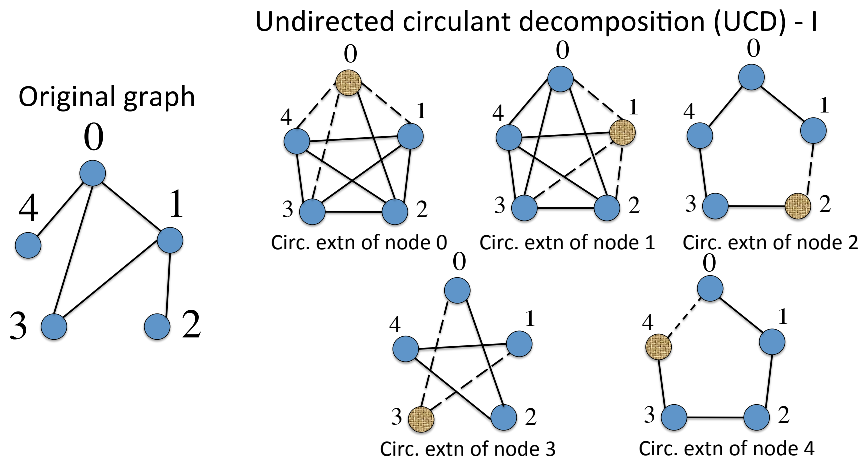


Figure 6.4: Example showing the different component graphs in the Undirected Circulant Decomposition I of the given graph. Each of the component circulant graphs can be viewed as circulant extensions of the neighborhood of each node.

6.3.2 Undirected Circulant Decomposition-I

In this section, we will consider a decomposition with $N+1$ terms. Each of the $N+1$ circulant graphs in the decomposition will be derived based on the structure of the given graph.

Theorem 15. *(*Undirected Circulant Decomposition-I (UCD-I)*:) Consider a graph G whose Laplacian and adjacency matrices are $\mathbf{L}$ and $\mathbf{A}$, respectively. Then $\mathbf{L}$ can be written as,*

$$\mathbf{L} \;=\; \mathbf{D} + \overbrace{\sum_{k=0}^{N-1} \mathbf{Q}_k \, \mathbf{L}_k \, \widehat{\mathbf{Q}}_k}^{=-\mathbf{A}}, \tag{6.25}$$

where $\mathbf{D}$ is the degree matrix of G; each $\mathbf{Q}_k$ is a matrix whose only nonzero entry is $Q_k(k,k)=1$; and $\widehat{\mathbf{Q}}_k$ is a diagonal matrix constructed from the k^{th} row of $\mathbf{L}$, i.e. $\widehat{Q}_k(i,i) = L(k,i),\ i = 0,\cdots,N-1$. The Laplacian matrix $\mathbf{L}_k$ represents a simple, unweighted, and undirected circulant graph defined by the set S_k characterized below:

$$S_k \;=\; \Big\{ i \;\Big|\; i \in \{1,\ldots,N-1\} \wedge L(k,(k+i)_N) \neq 0 \Big\} \tag{6.26}$$

Proof: The proof builds on the idea for OUCD. In OUCD, the circulant graphs are fixed and the entries of $\widehat{\mathbf{Q}}_{k,m}$ are chosen to match the entries of $\mathbf{L}(k,:)$ wherever $\mathbf{L}_m(k,:)$ has non-zero entries. This can now be modified to choose the entries of the k^{th} row of $\mathbf{L}_k$ in such a way that it matches the non-zero entries of $\mathbf{L}(k,:)$. The only constraint is that the first row of $\mathbf{L}_k$ has to be symmetrical and this can be imposed in the construction. This construction is best illustrated using a simple example. Consider the graph in 6.4. The Laplacian matrix of this graph is given by,

$$\mathbf{L} = \begin{bmatrix} 3 & -1 & 0 & -1 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 2 & 0 \\ -1 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.27}$$

Consider the first row. In order to reconstruct the off-diagonal entries of this row, we need that

$$L_0(0,1) = L_0(0,3) = L_0(0,4) = -1. \tag{6.28}$$

However this would not be an undirected circulant matrix unless $L_0(0,2) = -1$. Hence setting this to be -1, we get that

$$\mathbf{L}_0 = \begin{bmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{bmatrix}. \tag{6.29}$$

Now choose $\mathbf{Q}_0$ and $\widehat{\mathbf{Q}}_0$ as follows to reconstruct the first row,

$$\mathbf{Q}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{6.30}$$

$$\widehat{\mathbf{Q}}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.31}$$

Fig 6.3 shows an example of the matrices in the decomposition for Node 0 in the graph. Let us consider another row of $\mathbf{L}$ to make this concrete, say $\mathbf{L}(2,:)$. In order to reconstruct the off-diagonal entries, we need that $L_2(2,1) = -1$, which in turn requires that $L_2(0,4) = -1$. In order to make this undirected, we need $L_2(0,1) = -1$. Thus we have

$$\mathbf{L}_2 = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}, \tag{6.32}$$

$$\mathbf{Q}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{6.33}$$

$$\widehat{\mathbf{Q}}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{6.34}$$

The degree matrix $\mathbf{D}$ reconstructs the diagonal entries of $\mathbf{L}$. Fig 6.4 illustrates the rest of the graphs in this decomposition.

■

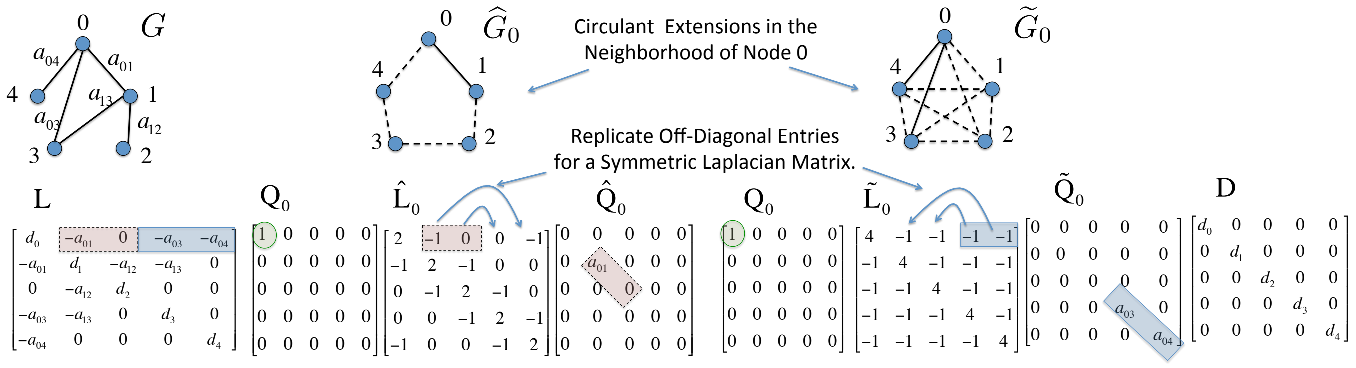
Remark: This decomposition is better than OUCD in that the number of terms is smaller. However, we lose the flexibility of having predetermined circulant graphs, and each of the circulant graphs depends on the original given graph. Conceptually, the circulant graphs can be thought of as circulant extensions of the neighborhood of each node in the original graph as shown in Fig 6.4. Note that we can interchange the roles of $\mathbf{Q}_k$ and $\widehat{\mathbf{Q}}_k$ by symmetrizing the columns. Since $\mathbf{L}$ and $\mathbf{L}_k$ are symmetric, we have that,

$$\mathbf{L} = \sum_{k=0}^{N-1} \mathbf{Q}_k \mathbf{L}_k \widehat{\mathbf{Q}}_k + \mathbf{D}, \tag{6.35}$$

$$= \sum_{k=0}^{N-1} \widehat{\mathbf{Q}}_k \mathbf{L}_k \mathbf{Q}_k + \mathbf{D}. \tag{6.36}$$

6.3.3 Undirected Circulant Decomposition-II

In UCD-I, we assumed that the input Laplacian is undirected, which has been our assumption throughout. However, as we will see later, when we want to define filters on the graph, the above decomposition restricts them to be symmetric for each node. In order to overcome this restriction, we slightly modify UCD-I, to get a $2N+1$ term decomposition that allows the flexibility of defining



G
Ĝ₀
G̃₀
Circulant Extensions in the Neighborhood of Node 0
Replicate Off-Diagonal Entries for a Symmetric Laplacian Matrix.
L
Q₀
L̂₀
Q̂₀
Q₀
L̃₀
Q̃₀
D

Figure 6.5: Example of the circulant graphs in the UCD-II decomposition of the given graph. The circulant graphs can be viewed as a circulant extensions of the Node 0 in this graph. Each of the circulant graphs reconstruct one half of Row 0 of the original Laplacian matrix.

general filters. The following theorem illustrates this decomposition.

Theorem 16. *(*Undirected Circulant Decomposition-II (UCD-II)*:) Consider a graph G whose Laplacian and adjacency matrices are* $\mathbf{L}$ *and* $\mathbf{A}$*, respectively. Then* $\mathbf{L}$ *can be written as*

$$\mathbf{L} \quad = \quad \mathbf{D} + \overbrace{\sum_{k=0}^{N-1}(\mathbf{Q}_k\,\widehat{\mathbf{L}}_k\,\widehat{\mathbf{Q}}_k + \mathbf{Q}_k\,\widetilde{\mathbf{L}}_k\,\widetilde{\mathbf{Q}}_k)}^{=-\mathbf{A}}, \tag{6.37}$$

where $\mathbf{D}$ *is the degree matrix of G; each* $\mathbf{Q}_k$ *is a matrix whose only non-zero entry is $Q_k(k,k) = 1$; and each* $\widehat{\mathbf{Q}}_k$ *and* $\widetilde{\mathbf{Q}}_k$ *is a diagonal matrix whose non-zero entries we now describe in their most general form. For $i = 1, \ldots, \lfloor N/2 \rfloor$,*

$$\widehat{Q}_k((k+i)_N,(k+i)_N) \quad = \quad -L(k,(k+i)_N) = A(k,(k+i)_N). \tag{6.38}$$

For $i = 1 + \lfloor N/2 \rfloor, \ldots, N-1$,

$$\widetilde{Q}_k((k+i)_N,(k+i)_N) \quad = \quad -L(k,(k+i)_N) = A(k,(k+i)_N). \tag{6.39}$$

The Laplacian matrices $\widehat{\mathbf{L}}_k$ *and* $\widetilde{\mathbf{L}}_k$ *represent simple, unweighted, and undirected circulant graphs defined by the sets $\widehat{S}_k$ and $\widetilde{S}_k$, respectively, characterized below:*

$$\widehat{S}_k \quad = \quad \Big\{i \;\Big|\; i \in \{1, \ldots, \lfloor N/2 \rfloor\} \wedge L(k,(k+i)_N) \neq 0\Big\}, \tag{6.40}$$

$$\widetilde{S}_k \quad = \quad \Big\{i \;\Big|\; i \in \big\{1 + \lfloor N/2 \rfloor, \ldots, N-1\big\} \wedge L(k,(k+i)_N) \neq 0\Big\}. \tag{6.41}$$

Proof: The proof follows on the lines of UCD-I. Instead of symmetrizing each of the rows completely for UCD-I, in this decomposition, we split the row into two halves and symmetrize them individually. This would give us two circulant graphs for every node and its neighbors. We will again outline the proof through an example. Fig. 6.5 depicts a circulant decomposition of the five-node graph G.

To illustrate the method, we reconstruct row 0 of $\mathbf{L}$. Each of the matrices $\widehat{\mathbf{L}}_0$ and $\widetilde{\mathbf{L}}_0$ in Fig. 6.5 inherits half of the off-diagonal entries of row 0 of $\mathbf{L}$. We want $\widehat{\mathbf{L}}_0$ and $\widetilde{\mathbf{L}}_0$ to be the Laplacian matrices of undirected circulant graphs, labeled $\widehat{G}_0$ and $\widetilde{G}_0$ in Fig. 6.5, respectively. The graphs $\widehat{G}_0$ and $\widetilde{G}_0$ can be viewed as circulant extensions of the neighborhood of node 0 of G. Per force, half of the row entries of $\widehat{\mathbf{L}}_0$ and $\widetilde{\mathbf{L}}_0$ completely specify each full matrix. For example, once we fix $\widehat{L}_0(0,1)$ and $\widehat{L}_0(0,2)$, it must be that $\widehat{L}_0(0,3) = \widehat{L}_0(0,2)$ and $\widehat{L}_0(0,4) = \widehat{L}_0(0,1)$. The last two entries of row 0 in $\widetilde{\mathbf{L}}_0$, too, must be mirror images of the corresponding first two off-diagonal entries.

Each successive row of $\widehat{\mathbf{L}}_0$ and $\widetilde{\mathbf{L}}_0$ is obtained by a right circular shift of its preceding row. The matrix $\mathbf{Q}_0$ selects row 0 of $\widehat{\mathbf{L}}_0$ and $\widetilde{\mathbf{L}}_0$. The matrices $\widehat{\mathbf{Q}}_0$ and $\widetilde{\mathbf{Q}}_0$ select the corresponding entries of row 0 of $\widehat{\mathbf{L}}_0$ and $\widetilde{\mathbf{L}}_0$, respectively—which match the corresponding off-diagonal entries in $\mathbf{L}$. Since $\mathbf{Q}_0\,\widehat{\mathbf{L}}_0\widehat{\mathbf{Q}}_0 + \mathbf{Q}_0\,\widetilde{\mathbf{L}}_0\widetilde{\mathbf{Q}}_0$ does *not* reproduce $L(0,0)$, we add $D(0,0)$ to the appropriate entry of $\mathbf{D}$.

■

6.4 Properties of the circulant decompositions

Each of the circulant decompositions have their own advantages and disadvantages. We will now list a set of desirable properties of a circulant graph decomposition and analyze each of the decompositions from this perspective. We only focus on the undirected decompositions.

- **Minimality:** The number of terms in the decomposition should be minimal. For the OUCD decomposition, we had $O(N^2)$ terms in the decomposition. However, if the graph is sparse with each node having an average degree d, then the number of terms is $O(Nd)$. UCD-I and UCD-II have $O(N)$ terms in the decomposition. This can still be prohibitively high for large graph datasets with millions of nodes. In such a case, it might be infeasible to decompose the graph into a million terms.

- **Sparsity:** If the given graph is sparse, then we might want each of the component graphs in the decomposition to be sparse as well. Note that in OUCD, all the component graphs are fixed a priori and also sparse since each node only has degree 2. The component graphs in UCD-I and UCD-II closely follow the structure of the original graph. Hence if the node degree in the original graph is small then the decomposed graphs too have a small node degree. Note that one could always decompose a given graph into fully connected graphs (replace the graphs in each of the decompositions by fully connected graphs and verify that the decomposition still holds). But then this would not inherit any meaningful structure from the original graph.

- **Graph independence:** It might be desirable for certain applications if the component graphs in the decomposition are independent of the given graph (i.e. this would in some sense behave as a standard basis). This would be helpful when we are processing graph structured data in real-time when the decomposition itself can be a bottleneck in terms of the computational cost. Note that the OUCD satisfies this property while the other decompositions do not.

- **Amenable to filter designs:** One of the main goals of the graph decompositions is to be able to extend filter designs from circulant graphs to general graphs. The decompositions should be able to help us generalize the filter operations. As we shall see in the next chapter, UCD-II will help us decompose a general LSV filter on a general graph to component LSI filters on each of the circulant graphs. The other two decompositions place restrictions on the filters that can be generalized.

- **Permutation Invariance:** It would be highly desirable for the decomposition to be invariant to the node ordering, i.e. we should get the same component graphs irrespective of the node ordering we choose. Classically, in the graph theory literature, decomposition of a given graph G with respect to a graph H, is defined to be a partitioning of the edges of G, with each partition corresponding to a graph that is isomorphic to H. By definition this decomposition is invariant to node ordering. However, it is well known that this problem is NP-complete [43].

  Unfortunately, the decompositions we have presented here are not invariant to the node ordering. Any heuristic algorithm for graph decomposition would likely give a decomposition that is dependent on the initial state which is the node ordering here. Hence, one can hope to obtain a reasonably good decomposition by starting with a good initial point. In the next section, we will discuss an algorithm that obtains a particular node ordering which we will heuristically argue that is good for our decompositions. We use this algorithm in all our simulation and application examples.

6.4.1 Reverse Cuthill-McKee Ordering

We saw that the initial node ordering is important for the proposed circulant decompositions. Intuitively, if the local neighborhoods of the nodes in the given graph are approximately circulant, then it is likely that the component graphs in the decomposition well-capture the local properties of the original graph. By this, we mean that if we look at nodes neighboring one other, then their local neighborhoods should look similar. It might be reasonable to expect this of many graphs, since

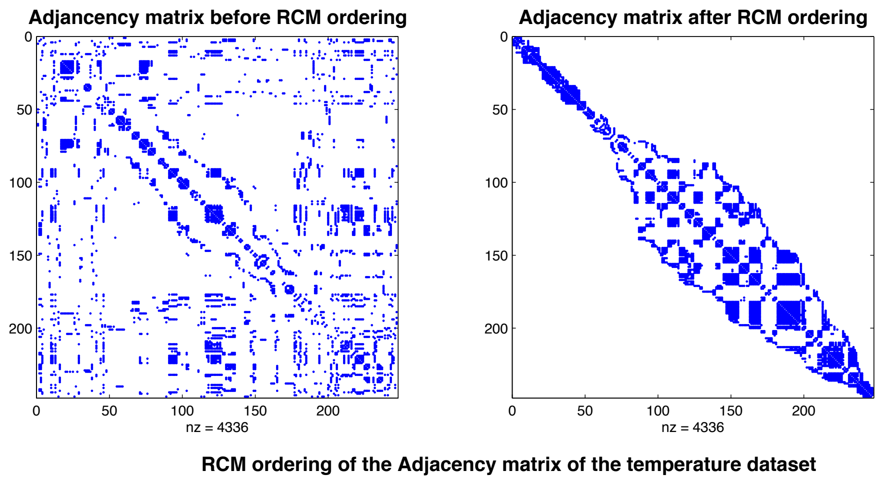


Figure 6.6: RCM ordering of the adjacency matrix of a United States temperature dataset [2](more details of the dataset in Section 8.5.1). The original matrix and the RCM ordered matrices are shown. The shaded entries are the non-zero entries of the matrix.

local structures tend to be similar. For example, in the Watts-Strogatz model [23] for small world networks, it is quite likely that nodes that are neighboring to each other see the same or similar neighborhoods. In other words, if we look at the circulant decomposition, then the circulant graphs that we get for neighboring nodes is likely to be the same. Thus we could design filters that slowly vary across the neighborhoods.

One of the major challenges in this approach is that even though a given graph might have such a property of local similarity, the node ordering might not reflect it. Finding the best node ordering to fit the model is likely to be a very hard problem. In order to capture some of these properties, we resort to a node ordering called as the Reverse-Cuthill-McKee (RCM) ordering [44]. This ordering tries to obtain a permutation of the rows and columns such that the entries are concentrated towards the diagonal i.e. reduce the *bandwidth* of the matrix. The *bandwidth* of a matrix is the smallest number of adjacent diagonals to which all the non-zero entries of the matrix are confined to. In other words, the ordering tries to make the matrix as close as possible to a banded matrix. The RCM algorithm is a heuristic that obtains a matrix with small bandwidth.

Notice that a matrix with b bands, having all the band entries to be non-zero corresponds to a circulant graph with generator set $S = \{1, 2, \cdots, b-1\}$. Thus intuitively, an approximately banded matrix such as the one obtained by the RCM ordering, is likely to have locally circulant structures that can be captured by the decomposition. Fig 6.6 shows an example of the RCM ordering of a given adjacency matrix.

6.5 Chapter highlights

- We consider different decompositions of a general graph into circulant components where the decomposition is defined as expressing the Laplacian matrix of the original graph as a linear summation of the Laplacian matrices of circulant graphs, where each Laplacian matrix term in the summation is pre and post multiplied by diagonal matrices.
- If one restricts the decompositions to only have either pre multiplying diagonal matrices or post multiplying diagonal matrices but not both, then we can show that the circulant graphs so obtained have to be directed in nature.
- For decompositions into undirected circulant graphs, we propose three decompositions,
  - Overcomplete Undirected Circulant Decomposition (OUCD)
  - Undirected Circulant Decomposition- I (UCD-I)
  - Undirected Circulant Decomposition-II (UCD-II)
- The OUCD has the advantage that the component graphs are invariant to the original graph, but is overcomplete in nature. There are $O(Nd)$ terms in the decomposition where d is the average degree of the nodes.
- UCD-I has $N+1$ terms in the decomposition, but the component graphs depend on the given graph.
- UCD-II has $2N+1$ terms in the decomposition and the component graphs depend on the given graph. The advantage over UCD-I is that in LSV filter design, we will see in the next chapter that UCD-II allows to decompose arbitrary LSV filters into LSI filters over the component graphs.
- All the decompositions have the primary drawback that they depend on the node ordering. In general, obtaining an ordering independent decomposition is known to be a hard problem. The initial node ordering plays an important role in the decomposition. In order to have a good initial point, we use a heuristic algorithm called the Reverse-CutHill-McKee (RCM) ordering algorithm, that seems to provide reasonably good results in terms of capturing the local neighborhoods of the graph in the decomposition.

Chapter 7

Sampling on general graphs

In this chapter we discuss sampling and the associated reconnection strategies for general graphs. These are essential tools required for multi resolution analysis on graphs. We have defined these operations for circulant graphs in Chapter 4 and wish to extend these to general graphs based on the circulant decompositions.

7.1 Sampling on general graphs

There are various graph sampling algorithms defined in the literature for different applications [28]. For example, the second eigenvector or the Fiedler vector provides a partition of the graph into two disjoint components which is used in applications like segmentation [45]. Here we define sampling on a noncirculant graph, using the circulant decompositions from Chapter 6.

Consider the UCD-I decomposition 6.3.2 for simplicity. Let $\{G_0, ..., G_{N-1}\}$ be the set of circulant graphs into which a noncirculant graph G has been decomposed, and let $\{S_0, ..., S_{N-1}\}$ be their corresponding generating sets. The key idea is to downsample on each of the circulant graphs, which corresponds to a downsampling pattern on the parent graph. For this, it is important to ensure that the downsampling pattern is consistent across the individual circulant graphs.

We have defined sampling with respect to every element s in S_k for a circulant graph G_k. Let $\mathbf{R}_k = \gcd(S_k, N)$, where the $\gcd$ acts on every element of S_k. The elements of $\mathbf{R}_k$ determine the sampling pattern (Lemma 3). Let $\mathbf{R}^* = \bigcap \mathbf{R}_k \neq \emptyset$. We choose the sampling pattern defined by any $r^* \in \mathbf{R}^*$. This corresponds to sampling the circulant graph G_k with respect to $s_k^* \in S_k$ such that $\gcd(s_k^*, N) = r^*$. Note that for some graphs, $\mathbf{R}^*$ might be the empty set. In that case, we choose the downsampling pattern that is consistent with the largest subset of the circulant graphs.

Given the downsampling strategy, one can now obtain conditions for the lossless recovery of the original signal. Define the signals $\boldsymbol{x}_k = \widehat{\mathbf{Q}}_k \boldsymbol{x}$ that are made to reside on the individual circulant graphs in the decomposition (we will see why this is the case when we discuss filtering in the next chapter). Assume that the edge weights are unity. Note that $\boldsymbol{x}_k$ consists of the signal values at node k and its neighbors. Based on the spectral properties of the signals $\boldsymbol{x}_k$, we can obtain a condition for the lossless recovery of the original signal after downsampling.

Theorem 17. (Alias-Free Recovery) *A signal $\boldsymbol{x}$ on a graph G can be downsampled according to the sampling pattern defined by r^*, and recovered without loss, if each signal $\boldsymbol{x}_k$ defined on the circulant graph G_k, for all k that are retained, can be downsampled with respect to s_k^* and subsequently recovered.*

Proof: Let $\boldsymbol{y} = \boldsymbol{x} \circ \boldsymbol{p}$ and $\boldsymbol{y}_k = \boldsymbol{x}_k \circ \boldsymbol{p}$, where $\circ$ represents element-wise multiplication and $\boldsymbol{p}$ is the downsampling pattern vector as defined in Lemma 3. We are given that $\boldsymbol{x}_k$'s can be recovered from $\boldsymbol{y}_k$ $\forall$ k such that $p(k) \neq 0$. We are interested in recovering $\boldsymbol{x}$ from $\boldsymbol{y}$. Note that we can get all the $\boldsymbol{y}_k$'s from $\boldsymbol{y}$ since $\boldsymbol{x}$ is a superset of the $\boldsymbol{x}_k$'s. Thus $\forall$ k such that $p(k) \neq 0$, we can recover $\boldsymbol{x}_k$ from $\boldsymbol{y}_k$ and hence $x(k)$. Now we need to see how to recover $x(k)$ such that $p(k) = 0$. We will show this by considering two different cases.

Case 1: $r^* = s^* \in S_k$.
If $p(k) = 0$, we know that $p(k + s^*) \neq 0$ by definition. From the construction of the circulant graphs and the associated signals, we have that $x(k) \in \boldsymbol{x}_{k+s^*}$ since k is a neighbor of the node $(k + s^*)$. Hence $x(k)$ can be recovered from $\boldsymbol{x}_{k+s^*}$ which is in turn recoverable from $\boldsymbol{y}_{k+s^*}$.

Case 2: $r^* \neq s^* \in S_k$.
We need to show that whenever $p(k) = 0$ there exists k' such that $p(k') \neq 0$ and $(k - k')$ mod $N \in S_{k'}$, i.e. k is a neighbor of k'. In this case we are done using the same argument as in Case 1. Let $s_k^* \in S_k$ be such that $\gcd(s_k^*, N) = r^*$. Consider nodes $k_1 = (k + s_k^*)_N$ and $k_2 = (k - s_k^*)_N$. Then we have that,

- $p(k_1) \neq 0$ and $p(k_2) \neq 0$.
  This follows from the sampling strategy since we are sampling with respect to s_k^*.

- Either $s_k^* \in S_{k_1}$ or $s_k^* \in S_{k_2}$ or both.
  For s_k^* to belong to S_k, we need at least one of the nodes k_1 or k_2 to be a neighbor of node k in the original graph. Correspondingly at least one of the sets S_{k_1} or S_{k_2} will have s_k^* due to the node k being a neighbor (note that we are dealing with undirected graphs).

Thus $x(k) \in \boldsymbol{x}_{k_1}$ or $x(k) \in \boldsymbol{x}_{k_2}$ or both and hence is recoverable from $\boldsymbol{y}_{k_1}$ or $\boldsymbol{y}_{k_2}$.

■

Remark: A sufficient condition for perfect signal recovery after sampling is that the GFT bandwidth of each of the signals $\boldsymbol{x}_k$, satisfies the maximum sampling bandwidth with respect to s_k^* on the graph $\mathbf{G}_k$, as defined in Theorems 4 and 5. This alias-free recovery condition is conservative, since the signals must be recoverable on $N/2$ separate circulant graphs. Nevertheless, this might be useful for general graphs having local neighborhoods with a circulant structure (e.g., small-world networks based on the Watts-Strogatz model [23]).

7.2 Reconnection strategies

As before, after downsampling, the new underlying graph on which the sampled signal resides needs to be defined. In the circulant case, we defined two strategies for reconnection—one based on *Kron*-reduction and the other a circulant preserving reconnection strategy. One of the major goals in the circulant case was to retain the circulant structure of the graph. For an arbitrary general graph there is no explicit structure that can be carried over to the component graphs. Hence there is no closure property (Section 4.3.1) that one can hope to retain. However, the other properties such as connectivity, spectral compaction and computational efficiency are still desirable of a reconnection algorithm.

Kron-reduction as defined in Section 4.3.2 is generic and can be applied to any graph. Thus, given the Laplacian matrix $\mathbf{L}$ of a graph and α the subset of nodes retained after downsampling based on any strategy, the Laplacian matrix of the new graph $\widehat{\mathbf{L}}$ is defined as follows,

$$\widehat{\mathbf{L}} \; = \; \mathbf{L}(\alpha,\alpha) - \mathbf{L}(\alpha,\alpha^C)\mathbf{L}(\alpha^C,\alpha^C)^{-1}\mathbf{L}(\alpha^C,\alpha)^{\mathsf{T}}. \tag{7.1}$$

The properties of *Kron*-reduction detailed in Section 4.3.2 hold for any general graph i.e. connectivity, spectral interlacing and the closure of the Laplacian matrix.

The circulant-preserving strategy was defined in order to retain the circulant structure of the graphs after downsampling. However, such a property is not defined for arbitrary graphs. Nevertheless we can extend the strategy to general graphs using the circulant decomposition in the following manner. For each circulant graph G_k in the decomposition, we know how to get the downsampled Laplacian matrix $\widehat{\mathbf{L}}_k$ from Section 4.3.3. Since the circulant graphs in the decomposition essentially reflect the neighborhood structure of each node, we can retain this in the new Laplacian $\widehat{\mathbf{L}}$ as follows,

$$\widehat{\mathbf{L}}(k,:) \; = \; \widehat{\mathbf{L}}_k(k,:), \tag{7.2}$$

i.e., the k^{th} row of the new Laplacian matrix is the k^{th} row of the Laplacian matrix of the corresponding circulant graph. Note that this construction gives a valid Laplacian matrix. However, the drawback of this approach is that the new Laplacian need not be symmetric and hence we might end up with a directed graph, which we currently do not know how to deal with. A naive method to overcome this would be to symmetrize the Laplacian matrix by making each of the edges undirected. However, the connectivity properties are not satisfied under this strategy. Hence we use the *Kron*-reconnection strategy in our simulations and example applications.

Fig 7.1 shows an example mesh graph [46] of a *Tapir*[1]. A synthetic LP signal is associated with

[1]A tapir is a large browsing mammal, similar in shape to a pig, with a short, prehensile snout

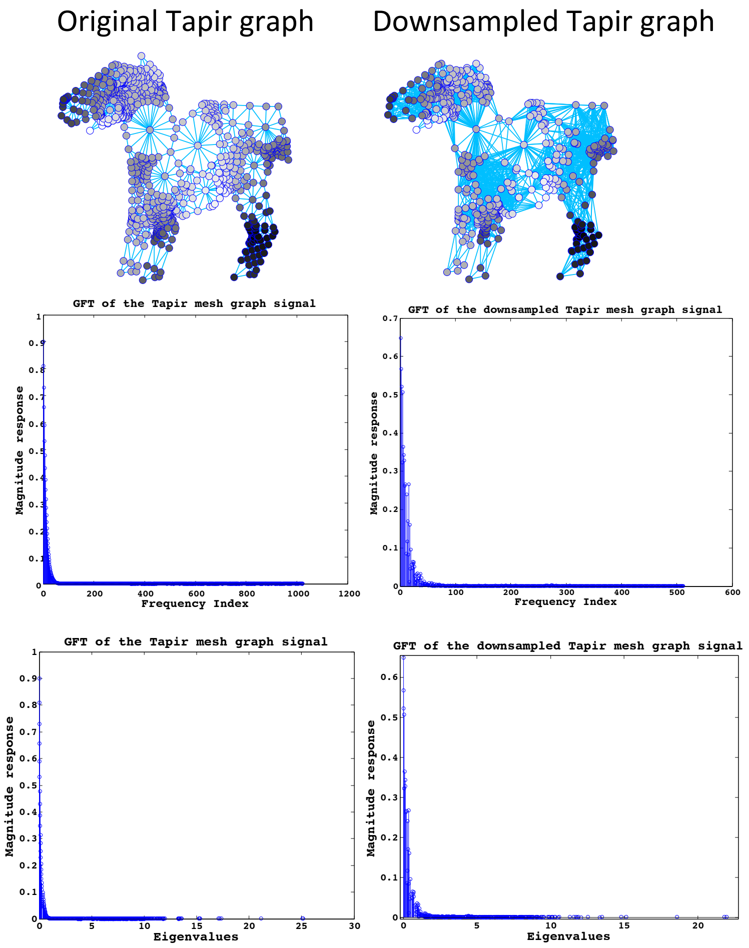


Figure 7.1: Example of a mesh graph of a *Tapir*. A synthetic LP signal is associated with the graph. The original graph is downsampled by two and reconnected using the *Kron*-reconnection strategy. The GFT of the signals before and after downsampling are shown in the figure.

the graph. The GFT of the signal is shown both as a function of the frequency index and the eigenvalues. The graph is downsampled by two after an RCM ordering of the nodes and reconnected using the *Kron*-reconnection strategy. Note that *Kron*-reconnection introduces edge weights which we explicitly do not shown in the figure. Edges are drawn between nodes whenever the edge weight is above a threshold. One can observe that the LP nature of the graph signal is retained in the downsampled graph and the downsampled graph is also visually representative of the original graph. The circulant reconnection strategy on the other hand does not provide any reasonable reconnections and hence we do not plot the graphs here.

7.3 Chapter highlights

- Sampling strategies for general graphs are defined based on the circulant decomposition. In particular, the generating set for each circulant graph is calculated and the sampling is chosen with respect to that element of the generating set for each graph which results in a consistent downsampling pattern for all the component graphs.
- Alias-free sampling is obtained as long as the signal bandwidth on each of the component circulant graphs is small enough for alias-free sampling on the corresponding circulant graph.
- *Kron*-reconnection can be used for general graphs and many of the properties carry over. The circulant preserving reconnection strategy does not retain many of the interesting properties and can result in a directed graph.

Chapter 8

Filter bank design on general graphs

Filtering is one of the important tools required for signal analysis. So far we discussed the basic operations required to define and design filters and filter banks for signals on general graphs. In particular, we discussed decompositions of general graphs into circulant graphs and extensions of the sampling and reconnection strategies. In this chapter, we will use the derived tools to define Linear Shift Varying (LSV) filters on general graphs and then transition to filter bank design for multi resolution signal analysis.

8.1 Shift-varying filters on general graphs

In this section, we derive the relation between filters on general graphs and the associated filters on the circulant graphs in the decomposition. Recall the definitions of a k-hop localized filter on a general graph from Section 5.1. We are aware that filters on general graphs are shift-varying (LSV) and we need the relation to LSI filters on the circulant graphs. The following lemma gives this relationship.

Theorem 18. *If* $\mathbf{H}$ *is a 1-hop LSV filter on* $\mathbf{G}$*, then we get,*

$$\mathbf{H} \;=\; \mathbf{D}_H + \sum_{k=0}^{N-1} \left(\mathbf{Q}_k \left(\mathcal{I}_{\widehat{\mathbf{L}}_k} \circ \widehat{\mathbf{H}}_k \right) \widehat{\mathbf{Q}}_k + \mathbf{Q}_k \left(\mathcal{I}_{\widetilde{\mathbf{L}}_k} \circ \widetilde{\mathbf{H}}_k \right) \widetilde{\mathbf{Q}}_k \right), \tag{8.1}$$

where each of the $\widehat{\mathbf{H}}_k$*'s and* $\widetilde{\mathbf{H}}_k$*'s are LSI filters on the circulant graphs, constructed from* $\mathbf{H}$ *in exactly the same way as the matrices* $\widehat{\mathbf{L}}_k$*'s and* $\widetilde{\mathbf{L}}_k$*'s are constructed from* $\mathbf{L}$ *in the UCD-II decomposition (Theorem 16).* $\mathcal{I}_{()}$ *is the modified indicator function defined in Equation 5.2 and* $\mathbf{D}_H = \text{diag}(\mathbf{H})$.

Proof: The proof follows by decomposing the filter matrix $\mathbf{H}$ as a linear combination of circulants in exactly the same way we decomposed the Laplacian of a general graph in the UCD-

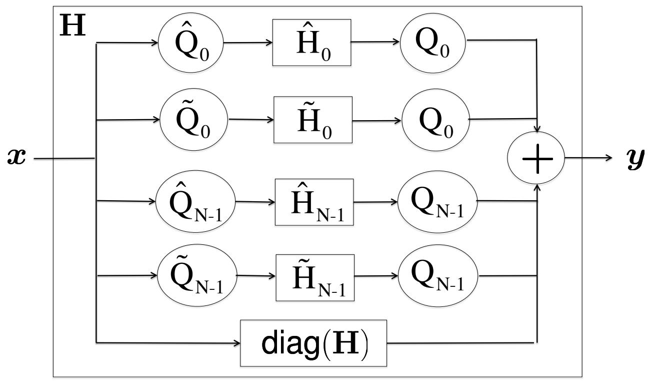


Figure 8.1: Decomposition of a LSV filter on a general graph as a bank of LSI filters on individual circulant graphs.

II decomposition (see Theorem 16). Note that the UCD-II decomposition is useful here since in general the filter $\mathbf{H}$ need not be symmetric.

■

Fig. 8.1 shows a block diagram description of this decomposition. Each of the filters $\widehat{\mathbf{H}}_k$ and $\widetilde{\mathbf{H}}_k$ act on the signals $\widehat{\mathbf{Q}}_k\boldsymbol{x}$ and $\widetilde{\mathbf{Q}}_k\boldsymbol{x}$ defined on the corresponding circulant graph. This definition extends to a k-hop filter by considering the circulant decomposition of the k-hop adjacency matrix.

Given a decomposition of the LSV filters into LSI filters, one approach is to design each of the LSI filters to have a desired filter response on each of the individual circulant graphs. This is similar to LTV filter banks where the individual LTI filters are designed to have a particular response for a fixed time duration and then the response changes. Similarly, suppose that the given graph have approximately circulant neighborhoods, then one can design filters whose response slowly varies across the graph.

Analogous to our discussion on filter banks for multi resolution analysis for circulant graphs, we shall design filter banks for general graphs satisfying some of the desirable properties we have listed in Section 5.2. The definition of a two-channel filter bank is similar to that we have for circulant graphs, with the analysis stage defined by the filters $\mathbf{H}_{\mathsf{LP}}$ and $\mathbf{H}_{\mathsf{HP}}$ and the synthesis stage defined by the filter $\mathbf{H}_{\mathsf{INV}}$. Fig 8.2 shows the filter bank structure. The downsampling operation are now carried out based on the circulant decomposition as we have discussed in Chapter 6. However one could use any other downsampling and reconnection strategy. We are interested in designing filter banks having the following desirable properties (Section 5.2)—perfect-reconstruction, critical-sampling, localization, localized reconstruction, orthogonality, graph-invariance and diagonalizability.

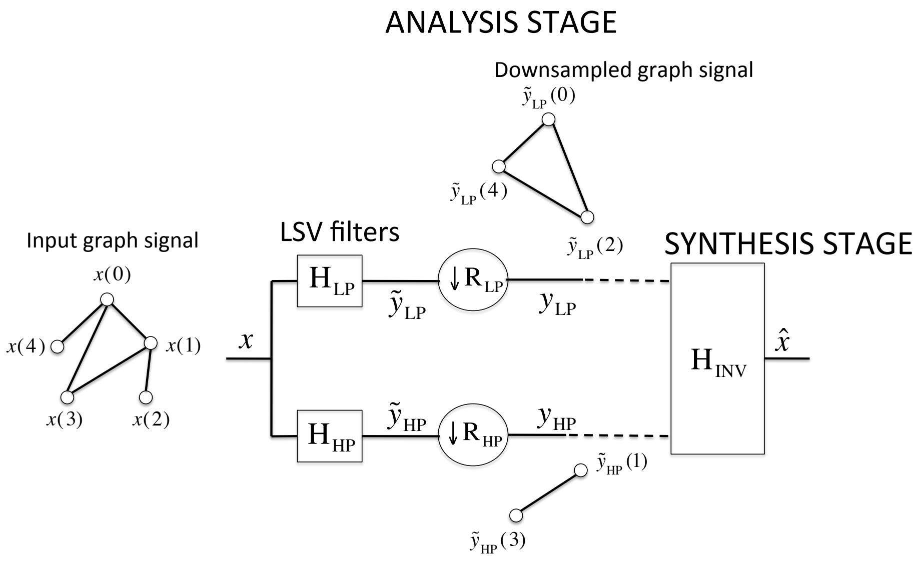


Figure 8.2: Example of a two-channel filter bank for signals defined on general graphs. The downsampling operation is carried out using the circulant decomposition. In this example, the downsampling factors in the high-pass and low-pass branches is chosen so as to have critical sampling. The filters $\mathbf{H}_{\mathsf{LP}}$ and $\mathbf{H}_{\mathsf{HP}}$ are LSV filters appropriately designed.

We would like to explicitly point out that the Laplacian Pyramid filter bank structure and the Spline-like filter bank structure that we discuss in the following sections are not dependent on the circulant decomposition of general graphs.

8.2 The Laplacian Pyramid for general graphs

The Laplacian pyramid, as defined in Section 5.3, is a generic oversampled filter bank structure that can be used for arbitrary graphs. The LP and HP outputs are generated as follows,

$$\widetilde{\boldsymbol{y}}_{\mathsf{LP}} = \mathbf{H}_{\mathsf{LP}}\boldsymbol{x}, \tag{8.2}$$

$$\boldsymbol{y}_{\mathsf{LP}} = (\widetilde{\boldsymbol{y}}_{\mathsf{LP}})\downarrow_2, \tag{8.3}$$

$$\boldsymbol{y}_{\mathsf{HP}} = \boldsymbol{x} - \widetilde{\boldsymbol{y}}_{\mathsf{LP}}, \tag{8.4}$$

As before, we have considerable flexibility in designing the LP filter, $\mathbf{H}_{\mathsf{LP}}$. However, note that one should be able to reconstruct the signal $\widetilde{\boldsymbol{y}}_{\mathsf{LP}}$ from its downsampled version $\boldsymbol{y}_{\mathsf{LP}}$. We do not have sharp aliasing and recovery results for general graphs. One conservative option is to design the filter in such a way that the alias-free sampling condition is met as derived in Theorem 17. Thus the LSI filters on each of the component circulant graphs will be designed to satisfy the alias-free recovery condition on these graphs.

One could also generalize the filters to allow aliasing and yet obtain perfect recovery. This follows from ideas in compressive sensing where enough linear combinations of the data is obtained for successful recovery given that it is known the data is sparse in some domain [47]. However, we might also want to maintain the underlying structure of the data in the downsampled signal which will help in multi resolution. Hence one might need to choose the sensing matrices (related to the filters in our context) carefully to allow for this. We do not explore this aspect in any further detail.

As discussed before, the Laplacian pyramid structure satisfies the perfect reconstruction property but is oversampled. The LP filter can be designed to meet the other desirable properties. For example, in order to satisfy the diagonalizability property, the filter needs to be a polynomial in the adjacency matrix which would also render it to be localizable on the graph domain. Further, if the coefficients of the polynomial are fixed, we would also satisfy the graph-invariance property.

Fig 8.3 shows an example of a Laplacian pyramid decomposition for a *Tapir* mesh graph. The LP filter is chosen to have an ideal filter response with cut-off frequency $N/2$. The Reverse-Cuthill-McKee algorithm is used to choose the node ordering. The downsampling and reconnection methodologies are carried out according to the methods described for general graphs. Note that in this example, the LP is not optimized for perfect-reconstruction and is only chosen for illustrative purposes.

8.3 Shift-varying Spline-like filter bank structures

The Laplacian pyramid structure offers flexibility in design at the loss of the critical-sampling property. Critical-sampling is necessary while dealing with very large scale datasets. In this section we will focus on critically-sampled filter banks motivated by Spline filters in classical signal processing.

8.3.1 Simple Spline-like filter banks

These are similar to the spline filters on circulant graphs wherein we take a normalized average of the neighboring nodes. Since each node can have different number of neighbors, we need a normalized adjacency matrix. The symmetric normalized $\mathbf{A}_n^{\mathsf{S}}$ and random-walk normalized $\mathbf{A}_n^{\mathsf{R}W}$ adjacency matrices and the corresponding Laplacian matrices have been defined in Section 2.1.

The LP and HP filters are now defined by replacing the adjacency matrix $\frac{1}{d}\mathbf{A}$ in Eqn. 5.12 and

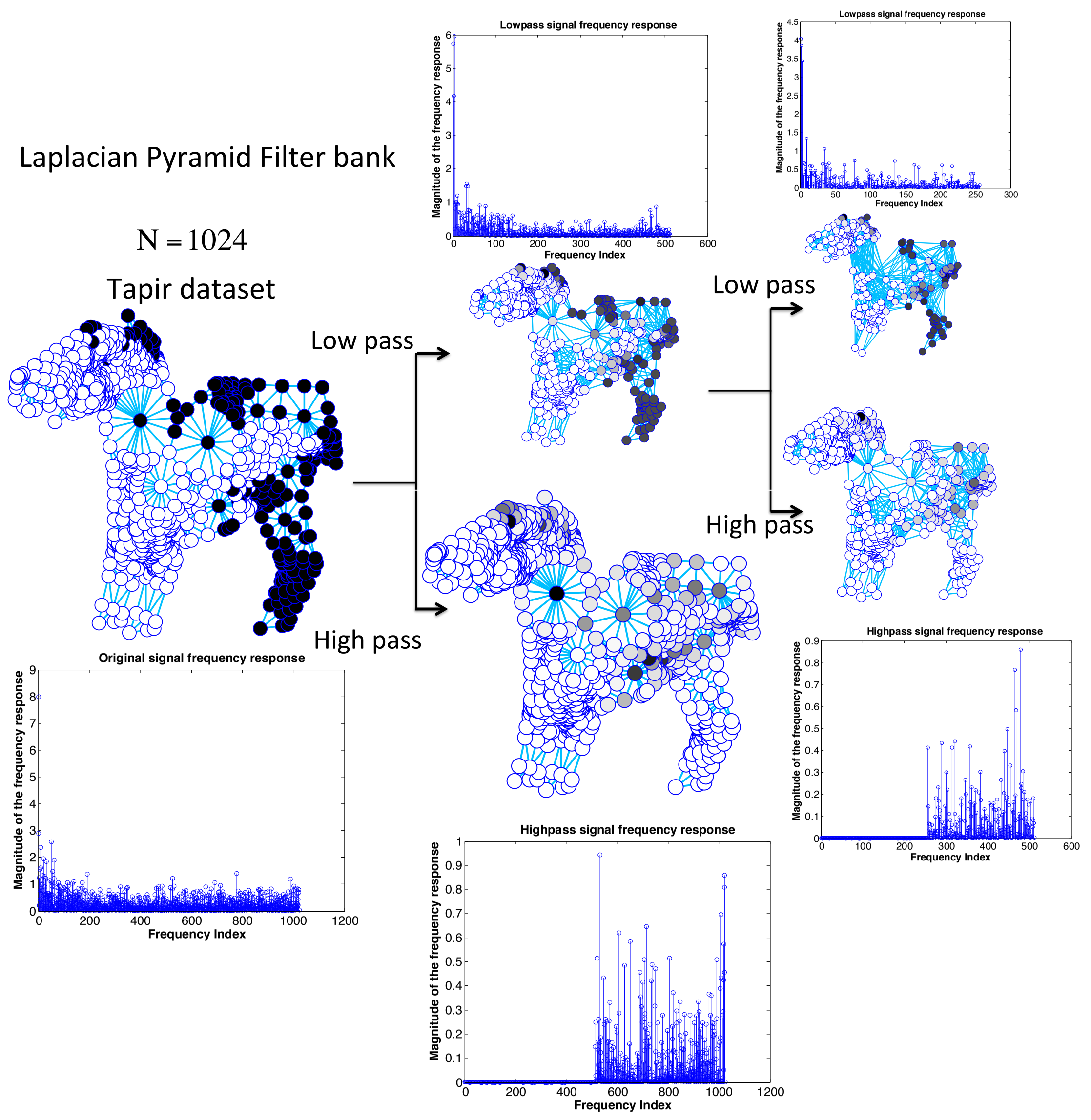

Laplacian Pyramid Filter bank
N = 1024
Tapir dataset
Low pass
High pass
Low pass
High pass
Lowpass signal frequency response
Highpass signal frequency response
Original signal frequency response
Magnitude of the frequency response
Frequency Index

Figure 8.3: Multiscale decomposition of a Tapir mesh graph using the Laplacian pyramid. The LP filter is chosen to have an ideal response. The graph signal is binary with one fourth of the nodes having the value one and the rest are zero.The Reverse-Cuthill-McKee algorithm is used to choose the node ordering. The downsampling and reconnection methodologies are carried out according to the methods described for general graphs.

Eqn. 5.15 by $\mathbf{A}_n^{\mathsf{S}}$ or $\mathbf{A}_n^{\mathsf{R}W}$. For example,

$$\begin{aligned}
\mathbf{H}_{\mathsf{LP}} &= \frac{1}{2}\left(\mathbf{I}_N + \mathbf{A}_n^{\mathsf{S}}\right), && (8.5)\\
&= \frac{1}{2}\left(2\mathbf{I}_N - \mathbf{L}_n^{\mathsf{S}}\right), && (8.6)\\
&= \left(\mathbf{I}_N - \frac{1}{2}\mathbf{L}_n^{\mathsf{S}}\right). && (8.7)\\
\mathbf{H}_{\mathsf{HP}} &= \frac{1}{2}\left(\mathbf{I}_N - \mathbf{A}_n^{\mathsf{S}}\right), && (8.8)\\
&= \frac{1}{2}\mathbf{L}_n^{\mathsf{S}}, && (8.9)
\end{aligned}$$

Theorem 19. *For a noncirculant graph, the spline filters defined in Eqn 8.7 and Eqn 8.9 form a critically-sampled perfect-reconstruction filterbank for any downsampling pattern, as long as at least one of the nodes retains a low-pass output.*

Proof: We will first prove this for the filters designed using the symmetric-normalized adjacency matrix and then consider the random-walk normalized adjacency matrix. Note that the proof of Theorem 10 did not explicitly make use of the fact that the graphs are circulant in nature. The proof relied on the following important properties of the eigenvalues of the adjacency matrix which hold true for the symmetric-normalized adjacency matrix as well [41].

- The eigenvalues of the normalized adjacency matrix λ_i, satisfy the property $-1 \leq \lambda_i \leq 1$.
- For a connected graph, the multiplicity of $\lambda_i = 1$ is unity.
- $\lambda_i > -1$ for non-bipartite graphs and $\lambda_i = -1$ with unit multiplicity for connected bipartite graphs.

Hence the same proof of Theorem 10 can be used to assert the claim in this Theorem for symmetric normalized matrices.

Let us now consider the case of the random-walk normalized adjacency matrix. Recall that the output of the two-channel filter bank $\boldsymbol{y}$ can be written as follows,

$$\boldsymbol{y} = \frac{1}{2}\left(\mathbf{I}_N + \mathbf{K}\mathbf{A}_n^{\mathsf{R}W}\right)\boldsymbol{x}, \tag{8.10}$$

where $\mathbf{K}$ is a diagonal matrix with $K(i,i) = 1$ if node i has the LP output after sampling, -1 otherwise. We want to show that $(\mathbf{I}_N + \mathbf{K}\mathbf{A}_n^{\mathsf{R}W})$ is invertible for the perfect-reconstruction property to hold. We have the following,

$$\begin{aligned}
\mathbf{I}_N + \mathbf{K}\mathbf{A}_n^{\mathsf{R}W} &= \mathbf{I}_N + \mathbf{K}\mathbf{D}^{-1}\mathbf{A}, && (8.11)\\
&= \mathbf{I}_N + \mathbf{K}\mathbf{D}^{-1/2}\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{D}^{1/2}, && (8.12)\\
&= \mathbf{I}_N + \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{D}^{1/2}, && (8.13)\\
&= \mathbf{D}^{-1/2}\left(\mathbf{I}_N + \mathbf{K}\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)\mathbf{D}^{1/2}, && (8.14)\\
&= \mathbf{D}^{-1/2}\left(\mathbf{I} + \mathbf{K}\mathbf{A}_n^{\mathsf{S}}\right)\mathbf{D}^{1/2}, && (8.15)
\end{aligned}$$

where the the third equality follows from the fact that $\mathbf{K}$ and $\mathbf{D}$ are diagonal matrices. Thus $(\mathbf{I}_N + \mathbf{K}\mathbf{A}_n^{\mathsf{R}W})$ is invertible since we showed that $(\mathbf{I}_N + \mathbf{K}\mathbf{A}_n^{\mathsf{S}})$ is invertible.

■

Remark: The Spline-like filters have the following properties: perfect-reconstruction, critical-sampling and graph-invariance. Note that these filters are not diagonalizable by the eigenvectors of the Laplacian matrix $\mathbf{L}$ as in the case of circulant graphs. However they are diagonalized by the eigenvectors of the corresponding normalized Laplacian matrices (i.e. $\mathbf{L}_n^{\mathsf{R}W}$ and $\mathbf{L}_n^{\mathsf{S}}$). The eigenvectors of the normalized Laplacian matrix too have properties similar to those of the Laplacian matrix. Hence, to get a sense for the filter response properties, one can plot the response obtained by diagonalization with the eigenvectors of the normalized Laplacian matrix. Further, note that the Spline-like filters obtained as a function of the random walk normalized Laplacian matrix are not symmetric in nature since the random-walk normalized adjacency matrix is not symmetric and therefore the eigenvectors are not orthonormal.

Fig 8.4 shows the response of the LP and HP filters for the *Tapir* mesh graph (see Fig. 8.5). The nature of the filters renders them to be linear as a function of the eigenvalues of the normalized Laplacian matrix.

Fig 8.5 shows an example of a mesh graph of a Tapir with 1024 nodes [46]. The figure illustrates spline wavelets at three different scales on the Tapir graph, localized on different regions. The basis functions are obtained by combining the filters at different stages in the decomposition. As the scale increases, the spread of the basis functions increases in the graph domain while remaining increasingly localized in the frequency domain.

Fig 8.6 shows a multiscale decomposition of a Tapir mesh graph with 1024 nodes. The graph signal is binary with ones on one fourth of the nodes and zeros otherwise. The RCM node ordering is used before carrying out the circulant decomposition. Downsampling and reconnection is carried out using the methods discussed in Chapter 7.

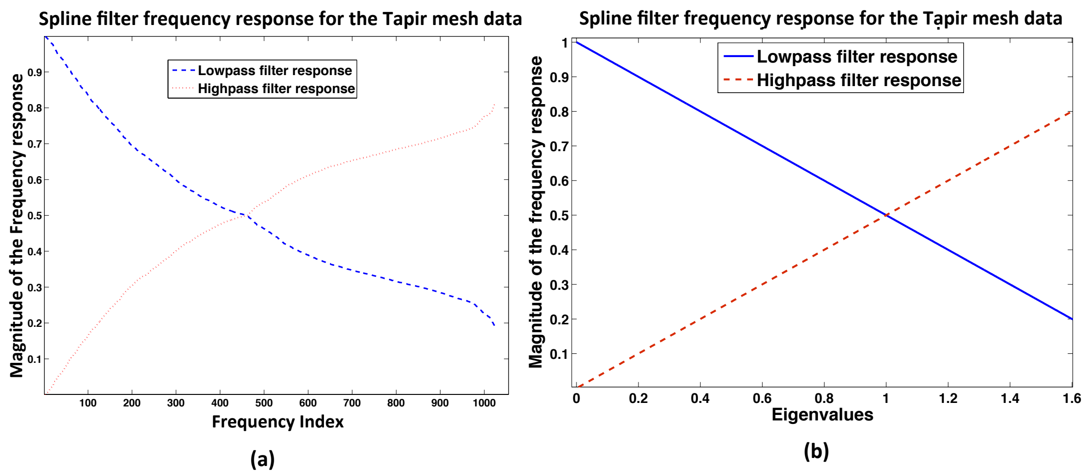


Figure 8.4: Response of the simple Spline-like filters shown for the *Tapir* mesh graph. Note that the response is obtained by diagonalization with the eigenvectors of the normalized Laplacian matrix. Hence the eigenvalues in the x-axis too are the eigenvalues of the normalized Laplacian matrix. The response is linear as a function of the eigenvalues given the structure of the filters.

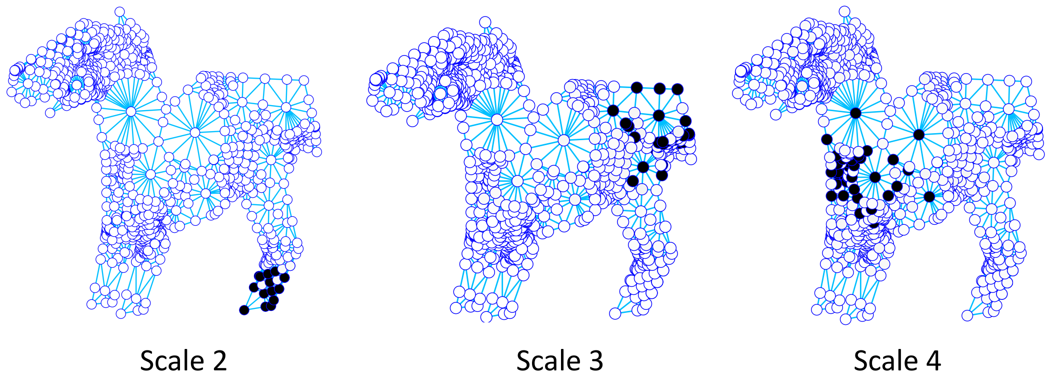
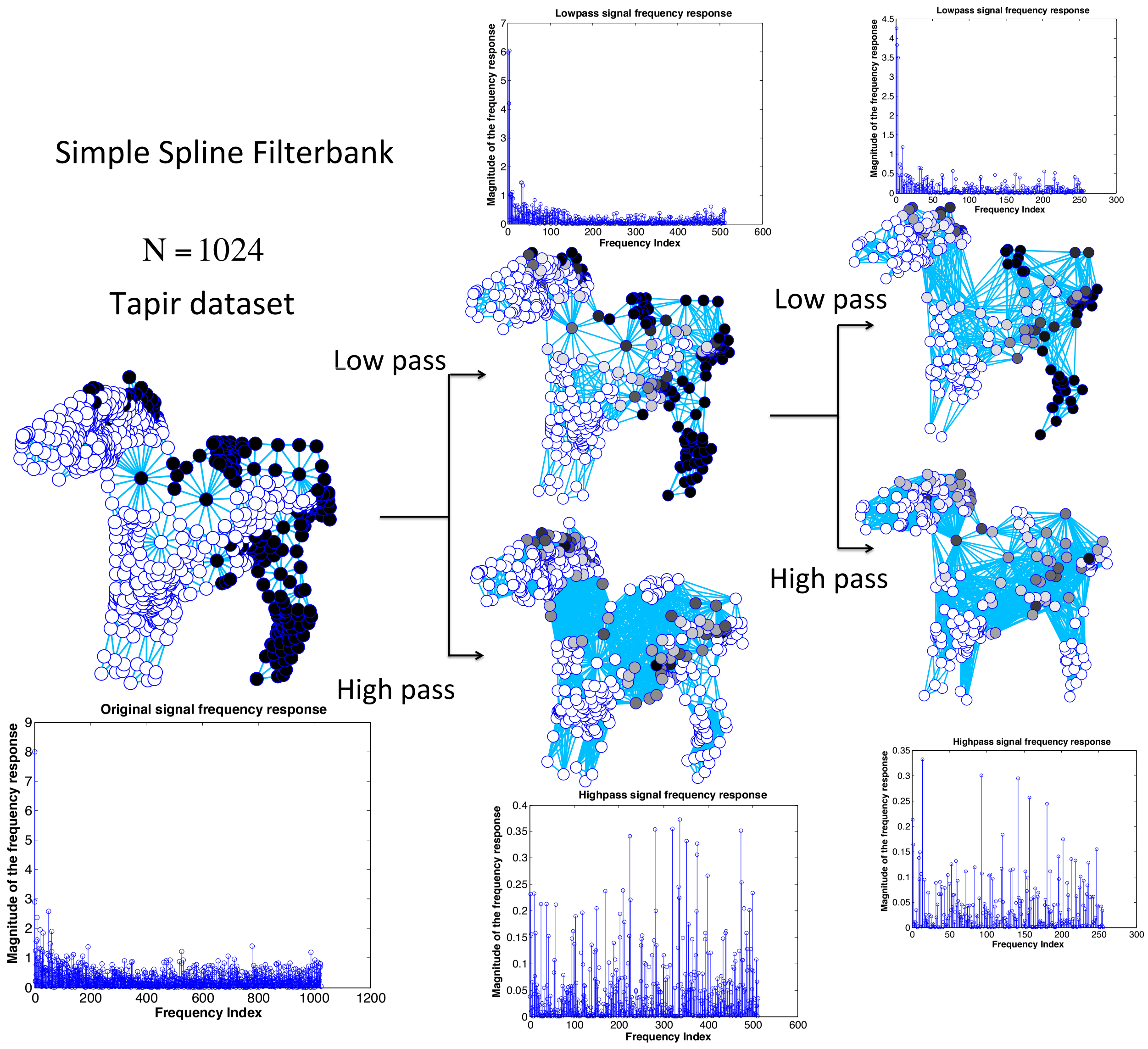


Figure 8.5: Spline wavelets at three different scales illustrated on the Tapir dataset. These are localized at different regions and the spread in the graph domain changes with the scale.



Simple Spline Filterbank
N = 1024
Tapir dataset
Low pass
High pass
Low pass
High pass
Original signal frequency response
Lowpass signal frequency response
Highpass signal frequency response
Lowpass signal frequency response
Highpass signal frequency response
Magnitude of the frequency response
Frequency Index

Figure 8.6: Multiscale decomposition of the Tapir mesh graph using simple Spline-like wavelets. The graph signal is binary with one fourth of the nodes having the value one and the rest are zero. One can observe that the HP output captures the transition of the signal value from unity to zero while the LP is qualitatively representative of the original signal.

8.3.2 Generalized weighted Spline-like filter banks

As before, the simple Spline-like filters have the disadvantage that the filter coefficients are fixed. In order to have flexibility in filter design, we shall introduce weights in the filter matrices that can be optimized over. Depending on the form of the filters, we have two versions of the generalized Spline-like filters—Generalized Weighted Spline-like filter banks- I (GWS-I) and Generalized Weighted Spline-like filter banks- II (GWS-II).

Generalized Weighted Spline-like filter banks- I (GWS-I)

Define the matrix $\widetilde{\mathbf{A}}_n$ as $\widetilde{A}_n(i,j) = W(i,j)A(i,j)$, where $W(i,j) \in [0,1]$ and $\sum_{j=0}^{N-1} W(i,j) = 1$. The LP and HP filters are now defined as follows,

$$\widetilde{\mathbf{H}}_{\mathsf{LP}} = \frac{1}{2}\left(\mathbf{I}_N + \widetilde{\mathbf{A}}_n\right), \tag{8.16}$$

$$\widetilde{\mathbf{H}}_{\mathsf{HP}} = \frac{1}{2}\left(\mathbf{I}_N - \widetilde{\mathbf{A}}_n\right). \tag{8.17}$$

Theorem 20. *The generalized spline filters defined above form a critically-sampled perfect reconstruction LSI filterbank for any downsampling pattern, as long as at least one of the nodes retains a low-pass output and the weights are symmetric—that is, $W(i,j) = W(j,i)$.*

Proof: Note that $\widetilde{\mathbf{A}}_n$ can be treated as the adjacency matrix of a weighted graph. The normalized adjacency matrix $\widetilde{\mathbf{A}}_n$ has the same properties as that of the normalized adjacency matrix of an unweighted graph [48] (Proposition 2.8). The eigenvalues are therefore bounded between -1 and 1. The minimum eigenvalue is -1 only for a bipartite graph. Thus the same proof of Theorem 19 can be used to prove the invertibility of these filters.

■

Remark: The only difference between the generalized Spline-like filters defined for circulant graphs in Section 5.4.2 and the ones above, is that we only allow for symmetric weights here. Note that we do not have any restriction on the number of components to be retained in the LP and HP branches as in Theorem 12. If suppose we restrict the weights to be symmetric in the generalized filters for circulant graphs as well then we would not have any restriction on the number of components to be retained in the two branches of the filter bank.

Note that the GWS-I filters are not diagonalizable in general by the eigenvectors of the Laplacian matrix (normalized or un-normalized). In the circulant graph case, since the filter banks were circulant, the diagonalizing matrices were the DFT matrices irrespective of the weights. The main disadvantage this poses is that it is unclear on how to optimize the weights since there is no notion of a global frequency response for these classes of filters. One option is to obtain the LSI decomposition of these filters as discussed in Section 8.1. The weights can then be optimized to obtain desirable frequency responses on the component circulant graphs. This can be computationally intensive as the size of the graphs increase. In the next section, we shall discuss a different version of the filter banks which allow for easier weight optimization at the loss of some flexibility in the design.

Generalized Weighted Spline-like filter banks- II (GWS-II)

From our previous discussion, it seems like the generalized filter bank problem design is more tractable if only the filters could be diagonalized by the eigenvectors of the Laplacian matrix. This can be achieved by restricting the form of the filters to be a polynomial in the adjacency matrix. Consider the following polynomial form of the filters for the LP and HP filter banks,

$$\mathbf{H}_{\mathsf{LP}} = \frac{1}{2}\left(\mathbf{I}_N + \sum_{\ell=1}^{J} w_\ell (\mathbf{A}_n^{\mathsf{S}})^\ell\right), \tag{8.18}$$

$$\mathbf{H}_{\mathsf{HP}} = \frac{1}{2}\left(\mathbf{I}_N - \sum_{\ell=1}^{J} w_\ell (\mathbf{A}_n^{\mathsf{S}})^\ell\right), \tag{8.19}$$

where $[w_1, \cdots, w_J]$ are the weights to be optimized for a desired filter response. One can similarly define filters using the random-walk normalized adjacency matrix. Let the eigen decomposition of $\mathbf{A}_n^{\mathsf{S}}$ be given as follows,

$$\mathbf{A}_n^{\mathsf{S}} = \mathbf{U}_{\mathsf{S}} \mathbf{\Gamma}_{\mathsf{S}} \mathbf{U}_{\mathsf{S}}^{\mathsf{H}}. \tag{8.20}$$

Note that the eigenvectors of the normalized adjacency matrix and the normalized Laplacian matrix coincide. The filters can now be expressed as follows,

$$\begin{aligned}\mathbf{H}_{\mathsf{LP}} &= \frac{1}{2}\left(\mathbf{I}_N + \sum_{\ell=1}^{J} w_\ell \mathbf{U}_{\mathsf{S}} \mathbf{\Gamma}_{\mathsf{S}}^{\ell} \mathbf{U}_{\mathsf{S}}^{\mathsf{H}}\right), &(8.21)\\ &= \frac{1}{2}\left(\mathbf{U}_{\mathsf{S}}\left(\mathbf{I}_N + \sum_{\ell=1}^{J} w_\ell \mathbf{\Gamma}_{\mathsf{S}}^{\ell}\right)\mathbf{U}_{\mathsf{S}}^{\mathsf{H}}\right). &(8.22)\end{aligned}$$

The eigenvalues of $\mathbf{H}_{\mathsf{LP}}$ are thus given by $\left(1 + \sum_{\ell=1}^{J} w_\ell \gamma_{\mathsf{S}}^{\ell}\right)$. Similarly we have that,

$$\mathbf{H}_{\mathsf{HP}} = \frac{1}{2}\left(\mathbf{U}_{\mathsf{S}}\left(\mathbf{I}_N - \sum_{\ell=1}^{J} w_\ell \mathbf{\Gamma}_{\mathsf{S}}^{\ell}\right)\mathbf{U}_{\mathsf{S}}^{\mathsf{H}}\right). \tag{8.23}$$

Theorem 21. *The generalized Spline-like filters defined in Eqn. 8.22 and Eqn. 8.23 form a critically-*

sampled perfect reconstruction LSI filter bank for any downsampling pattern as long as the weights satisfy one of the following properties,

$$w_\ell \geq 0 \quad \textit{and} \quad \sum_{\ell=1}^{J} w_\ell = 1, \tag{8.24}$$

$$\textit{or} \tag{8.25}$$

$$\left(\sum_{\ell=1}^{J} w_\ell \gamma_{\mathsf{S},i}^{\ell}\right)^2 > 1 \quad , \quad \forall i = 0, \cdots, N-1, \tag{8.26}$$

where $\{\gamma_{\mathsf{S},i}\}_{i=0}^{N-1}$*, are the eigenvalues of the normalized adjacency matrix,* $\mathbf{A}_n^{\mathsf{S}}$.

Proof: The proof steps are similar to that of Theorem 10. The output of the two-channel filter bank $\boldsymbol{y}$ can be written as follows,

$$\boldsymbol{y} = \frac{1}{2}\left(\mathbf{I}_N + \mathbf{K}\mathbf{B}_n^{\mathsf{S}}\right)\boldsymbol{x}, \tag{8.27}$$

where $\mathbf{K}$ is a diagonal matrix with $K(i,i) = 1$ if node i has the LP output after sampling, -1 otherwise and $\mathbf{B}_n^{\mathsf{S}} = \sum_{\ell=1}^{J} w_\ell (\mathbf{A}_n^{\mathsf{S}})^\ell$. We need to show that $\left(\mathbf{I}_N + \mathbf{K}\mathbf{B}_n^{\mathsf{S}}\right)$ is invertible. From Eqn 8.20 we have that,

$$\mathbf{B}_n^{\mathsf{S}} = \mathbf{U}_{\mathsf{S}}\left(\sum_{\ell=1}^{J} w_\ell \boldsymbol{\Gamma}_{\mathsf{S}}^{\ell}\right)\mathbf{U}_{\mathsf{S}}^{\mathsf{H}}. \tag{8.28}$$

Suppose $\boldsymbol{z} = \mathbf{U}_{\mathsf{S}}\boldsymbol{r}$ lies in the null space of $\left(\mathbf{I}_N + \mathbf{K}\mathbf{B}_n^{\mathsf{S}}\right)$. We then have the following,

$$\left(\mathbf{I}_N + \mathbf{K}\mathbf{B}_n^{\mathsf{S}}\right)\boldsymbol{z} = 0, \tag{8.29}$$

$$\left(\mathbf{I}_N + \mathbf{K}\mathbf{U}_{\mathsf{S}}\left(\sum_{\ell=1}^{J} w_\ell \boldsymbol{\Gamma}_{\mathsf{S}}^{\ell}\right)\mathbf{U}_{\mathsf{S}}^{\mathsf{H}}\right)\mathbf{U}_{\mathsf{S}}\boldsymbol{r} = 0, \tag{8.30}$$

$$\mathbf{U}_{\mathsf{S}}\boldsymbol{r} = -\mathbf{K}\mathbf{U}_{\mathsf{S}}\left(\sum_{\ell=1}^{J} w_\ell \boldsymbol{\Gamma}_{\mathsf{S}}^{\ell}\right)\boldsymbol{r}, \tag{8.31}$$

Taking the norm on both the sides of the equation we get that,

$$\sum_{i=0}^{N-1}\left(1-\left(\sum_{\ell=1}^{J} w_\ell \gamma_{\mathsf{S},i}^{\ell}\right)^2\right) r(i)^2 \;=\; 0, \tag{8.32}$$

$$\sum_{i=0}^{N-1}\left(1-\psi_i^2\right) r(i)^2 \;=\; 0, \tag{8.33}$$

where $\psi_i = \sum_{\ell=1}^{J} w_\ell \gamma_{\mathsf{S},i}^{\ell}$ are the eigenvalues of $\mathbf{B}_n^{\mathsf{S}}$ and $\gamma_{\mathsf{S},i}$ is the i^{th} eigenvalue of $\mathbf{A}_n^{\mathsf{S}}$. Suppose that $\left(\sum_{\ell=1}^{J} w_\ell \gamma_{\mathsf{S},i}^{\ell}\right)^2 > 1$, then clearly $\boldsymbol{r} = \mathbf{0}$ and the proof follows. Else if the weights satisfy the other condition, i.e., $w_\ell \geq 0$ and $\sum_{\ell=1}^{J} w_\ell = 1$, then we can show that the same proof of Theorem 10 carries over by showing that the eigenvalues of $\mathbf{B}_n^{\mathsf{S}}$ satisfy the following required properties.

- The eigenvalues ψ_i are bounded between -1 and 1. This follows from the fact that $-1 \leq \gamma_{\mathsf{S},i} \leq 1$ since $w_\ell > 0$ and $\sum_{\ell=1}^{J} w_\ell = 1$.

- The multiplicity of $\psi_i = 1$ is unity for connected graphs. This follows by substituting the value of $\gamma_{\mathsf{S},i} = 1$ whose multiplicity is unity in the case of connected graphs.

- For bipartite graphs the multiplicity of $\psi_i = -1$ is at most unity. We know that the multiplicity of $\gamma_{\mathsf{S},i} = -1$ is unity for bipartite graphs. Thus we have,

$$\sum_{\ell=1}^{J} w_\ell (-1)^\ell = \sum_{\ell \text{ even}} w_\ell - \sum_{\ell \text{ odd}} w_\ell \leq 1. \tag{8.34}$$

We also know that the eigenvectors of $\mathbf{B}_n^{\mathsf{S}}$ and $\mathbf{A}_n^{\mathsf{S}}$ are the same. Hence the proof of Theorem 10 goes through in this case and the filter banks satisfy the perfect reconstruction property.

∎

Remark: Note that for the second condition on the weights to hold (Eqn 8.26), the adjacency matrix of the original graph should not have any zero eigenvalues. Such graphs are known as non-singular graphs [40]. The filters $\mathbf{H}_{\mathsf{LP}}$ and $\mathbf{H}_{\mathsf{HP}}$ are diagonalizable by the eigenvectors of the graph

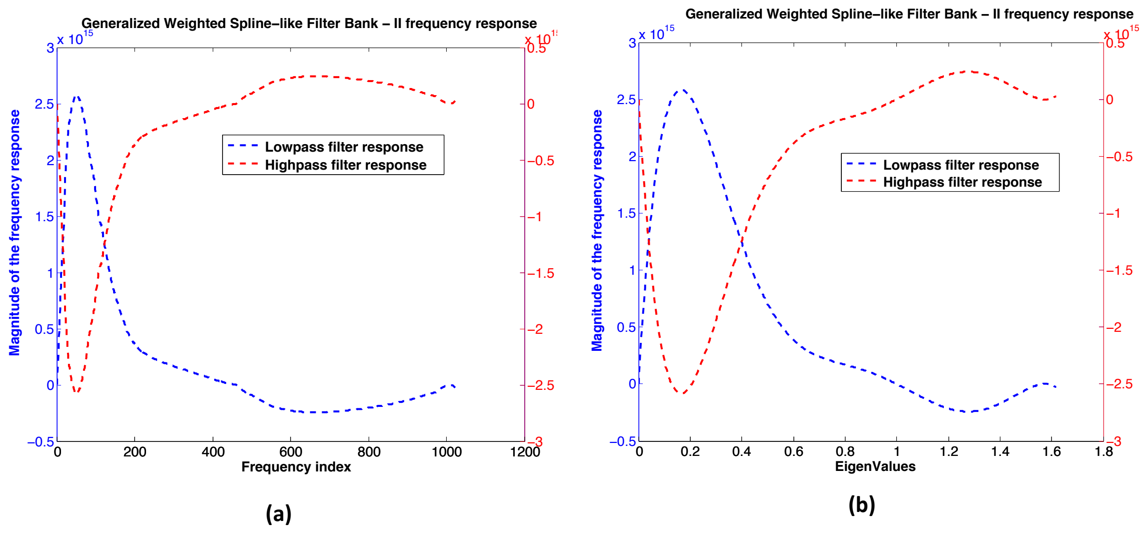


Figure 8.7: Example LP and HP GWS-II responses for the *Tapir* dataset plotted as a function of the frequency index (Fig (a)) and the eigenvalues (Fig (b)). The number of hops is taken to be 10 and the filter coefficients are optimized to approximate an ideal LP response. In this example, the constraints on the weights in the optimization forced the filter response at the zero-frequency to be zero. Additional constraints could be imposed to obtain desirable responses.

Laplacian matrix and their responses are given as follows,

$$H^{\mathsf{G}}_{\mathsf{LP}}(k) = \frac{1}{2}\left(1 + \sum_{\ell=1}^{J} w_\ell \gamma^{\ell}_{\mathsf{S},N-1-k}\right), \tag{8.35}$$

$$H^{\mathsf{G}}_{\mathsf{HP}}(k) = \frac{1}{2}\left(1 - \sum_{\ell=1}^{J} w_\ell \gamma^{\ell}_{\mathsf{S},N-1-k}\right). \tag{8.36}$$

A least-squares formulation or a similar optimization technique could be used to determine the weights in order to approximate a desired frequency response. Fig 8.7 shows an example of the LP and HP responses obtained after suitably optimizing the weights to approximate a desired ideal LP response. The number of hops in the filter is taken to be 10. One can observe that the response has a sharper cutoff in the transition band as compared to the simple Spline-like filters (Fig 8.4). However, the response at the lower frequencies, especially nearer to the zero-frequency is not good. Constraints could possibly be introduced in the optimization problem to have a better response depending on the application of interest.

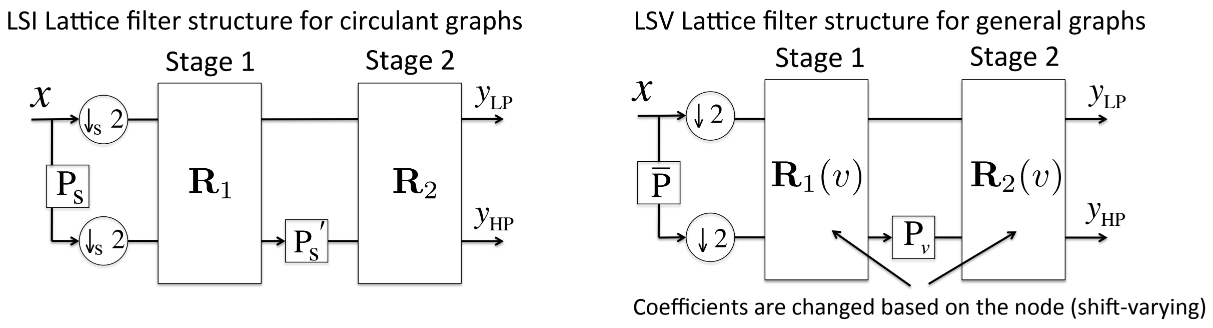


Figure 8.8: Lattice structure for filters on general graphs. The structure is similar to that of circulant graphs. The coefficients change for each node in the graph which can be optimized for every circulant graph in the decomposition. The shift $\bar{\mathbf{P}}$ is introduced to represent the fact that we keep the complementary set of nodes in the second branch after downsampling by two in the first branch. $\mathbf{P}_v$ is the shift on the corresponding circulant graph.

8.3.3 Properties of the Spline-like filter banks

- The filters satisfy the critical-sampling and perfect-reconstruction properties.
- They are localized in the graph domain given their explicit representation in terms of the adjacency matrix. Notice however that the reconstruction filter is not necessarily localized.
- The simple spline-like filters and GWS-II are diagonalizable by the eigenvectors of the normalized Laplacian matrix. GWS-I in general does not satisfy this property.
- The spline-like and GWS-II filters are graph-independent. However, GWS-I in general requires optimization of the coefficients for every node depending on the circulant decomposition which is dependent on the graph.

8.4 Shift-varying Lattice filter bank structures

The Lattice filter structures defined for circulant graphs in Section 5.5 can be extended to general graphs by varying the coefficients of the filter for different shifts. This is analogous to the design of lattice filters in the time domain to obtain time-varying frequency responses [49]. As long as each of the stages are invertible, we get perfect reconstruction by carefully applying the inverse filters at the right time instants. The coefficients of the filters are suitably optimized to get the desired frequency response at the corresponding time instants. Analogously, we design the filter coefficients to get the desired response for each of the component circulant graphs at the specified shift.

Fig. 8.8 shows the design of shift-varying lattice filters for circulant and general graphs. For LSI filters on circulant graphs, the coefficients of the different stages in the lattice($\mathbf{R}_1, \mathbf{R}_2$ here) do not change with the different signal shifts. For LSV filters, the coefficients are a function of the node being shifted to, i.e. $\mathbf{R}(v)$. The shifts in each of these filters depends on the circulant graph associated with that node. As long as each of these filters are invertible, the signal can be reconstructed. At the synthesis stage, care should be taken to apply the inverting filters in proper order. The shift $\bar{\mathbf{P}}$ is introduced to represent the fact that we keep the complementary set of nodes in the second branch after downsampling by two in the first branch.

The lattice structure we saw earlier guarantees perfect reconstruction. The question is how to design such filters. Note that, as in the case of the GWS-I, the filters here are not necessarily diagonalizable by the eigenvectors of the Laplacian matrix. Hence optimizing the coefficients is not easy. In the time domain, LTV filters are used to obtain different time-frequency tilings at different instants. Similarly, we might be interested in obtaining different spectral responses in different sections of the graph. Let us assume that locally the graph has the same circulant structure and this slowly varies in different sections of the graph. For each of these local circulant graphs, filter coefficients could be designed to give a desired frequency response in that part of the graph. Transitioning to a different part of the graph (with a different circulant structure), the coefficients can be changed to get the desired frequency response in that section of the graph.

There is a lot of work in the literature for designing time-varying lattice filters; one of the key issues is the design of boundary filters that smoothen the transition from one frequency response to another. These issues are present for general graphs as well. If the graph has no structure and arbitrarily varies at each shift, then it is hard to get a good overall filter response. We are hoping that graphs in real-world applications will have some sort of a "smooth" transition in local structures so that we can design graph-frequency tilings for different sections of the graph. More detailed analysis in this direction are future research directions.

The shift-varying Lattice filters are critically-sampled perfect-reconstruction filters by construction and are localized in both the analysis and synthesis stages. However they do not satisfy the diagonalizability property in general and are not graph-independent.

8.5 Example multiscale decompositions

In this section, we consider some more examples of general graph datasets including a real-world temperature dataset. In these examples, we use the Reverse-Cuthill-Mckee algorithm (section 6.4.1) to obtain an ordering of the nodes and then apply the multi scale decomposition techniques using the circulant decomposition of the resulting Laplacian matrix. The circulant decomposition is useful for two operations: First, the sampling set of the nodes for HP and LP is defined based on the individual circulant graphs in the decomposition. Second, the lattice filters

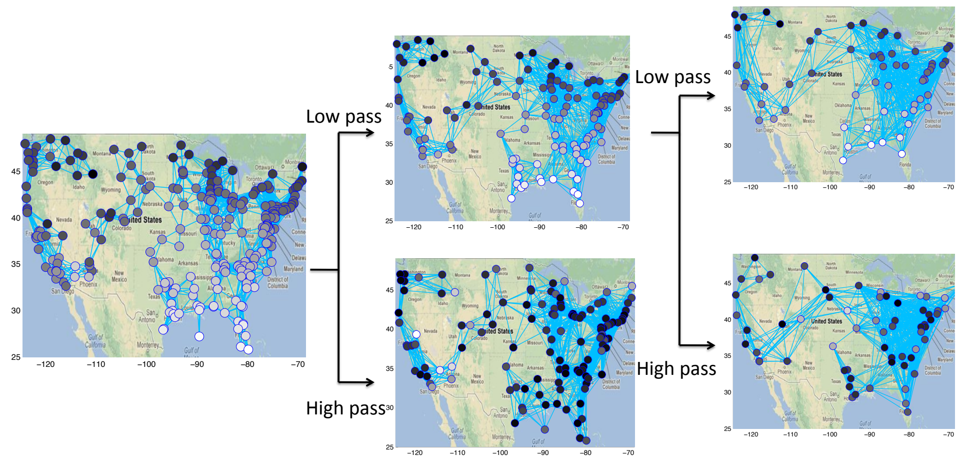


Figure 8.9: Multiscale decomposition of a US temperature dataset graph circulant graph using Spline-like filters and the *Kron*-reconnection scheme.

are obtained by optimizing the coefficients of the filter for each of the circulant graphs. Note that the simple Spline-like filter definition does not depend on the circulant decomposition except for determining the sampling set.

8.5.1 Weather dataset

In this example for a general graph, we consider a weather dataset. We extracted average temperature measurements at various locations across the United States during March 2012 from the Federal Climate Complex's weather dataset [2]. However, there is no predefined underlying graph structure for this dataset. We generated a Euclidean-distance-based underlying graph with nodes at each weather station (347 nodes) and edges between any two stations closer than a threshold distance. This graph structure captures the temperature correlation across physically close locations. In our framework, we assumed that the underlying graph is given and do not consider generating the graph from the data.

Fig. 8.9 shows the dataset plotted on a map of the US with its multiscale decomposition. The temperature signal is plotted on the graph by intensity—lighter coloration indicates higher temperature—and the signal undergoes two rounds of filtering and sampling. A simple one-hop Spline-like filter is used for the decomposition with a *Kron*-reduction scheme for the reconnection of the downsampled graph. The simple Spline-like filters are based on the random-walk-normalized adjacency matrix. The sampling strategy seems to provide a visually good approximation of the original graph. Further, the LP and HP filters naturally capture signal variations.

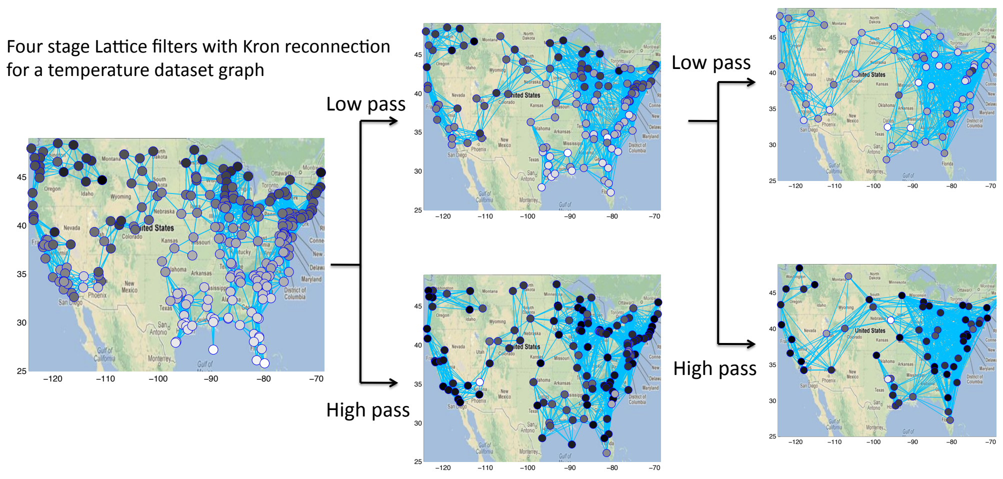

Four stage Lattice filters with Kron reconnection
for a temperature dataset graph
Low pass
High pass
Low pass
High pass

Figure 8.10: Multiscale decomposition of a US temperature dataset graph using 4-stage Lattice filters and the *Kron*-reconnection scheme.

Fig. 8.10 shows a multiscale decomposition of the temperature dataset using 4-stage lattice filters. The first stage is a visually reasonable representation of the original graph data. However, the filter design for the second stage does not give a good output. This is possibly due to not optimizing the shifts in the different stages of the Lattice filter.

8.5.2 Watts-Strogatz small world network graphs

In this section, we look at synthetic graphs generated using the Watts-Strogatz model [23] for small world networks. The model takes as input the average degree of each node and a parameter β. First, a circulant graph with the set $S = \{1, 2, ..., d_{avg}/2\}$ is generated. For each node in the graph, every edge incident to the node is randomly rewired with a probability β. Here we consider a graph generated according to this model with 64 nodes, mean degree 4 and $\beta = 0.1$.

Fig. 8.11 shows a multiscale decomposition of a synthetic Watts-Strogatz graph using Spline-like filters and the *Kron*-reduction scheme. Half of the nodes in this graph have the signal value unity and the other half have zero. Lighter coloration indicates a larger value. Fig. 8.12 shows a reconnection scheme with Spline-like filters and the reconnection scheme now replaced by the circulant reconnection strategy.

Clearly the output of the filters depends on the reconnection scheme being used. A good reconnec-

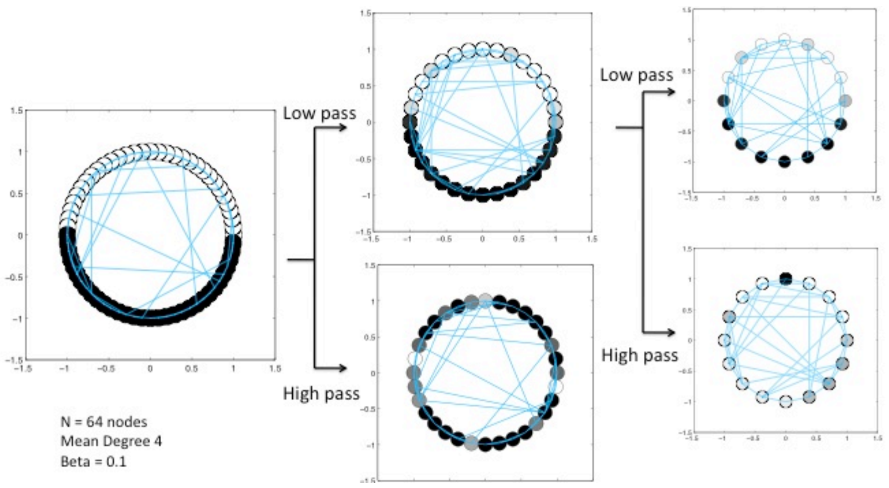


Figure 8.11: Multiscale decomposition of a synthetic Watts-Strogatz graph with $N = 64$, mean degree 4 and $\beta = 0.1$, using one-hop Spline-like filters and the *Kron*-reconnection scheme.

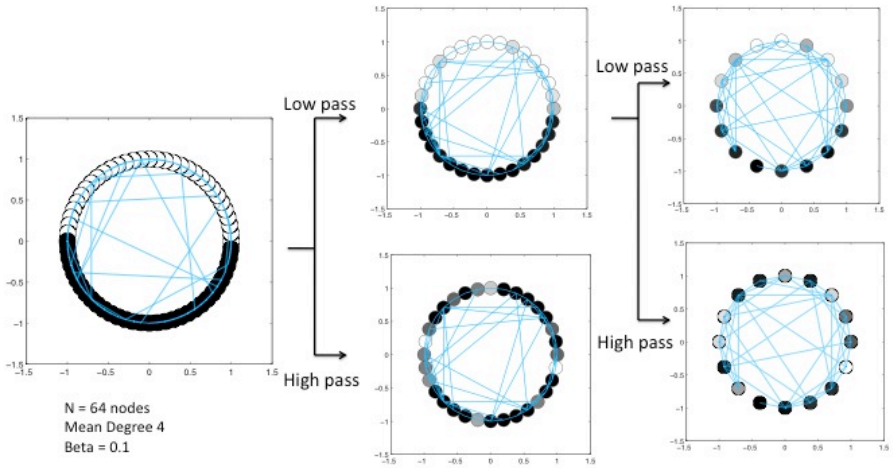


Figure 8.12: Multiscale decomposition of a synthetic Watts-Strogatz graph with $N = 64$, mean degree 4 and $\beta = 0.1$, using one-hop Spline-like filters and the circulant reconnection scheme.

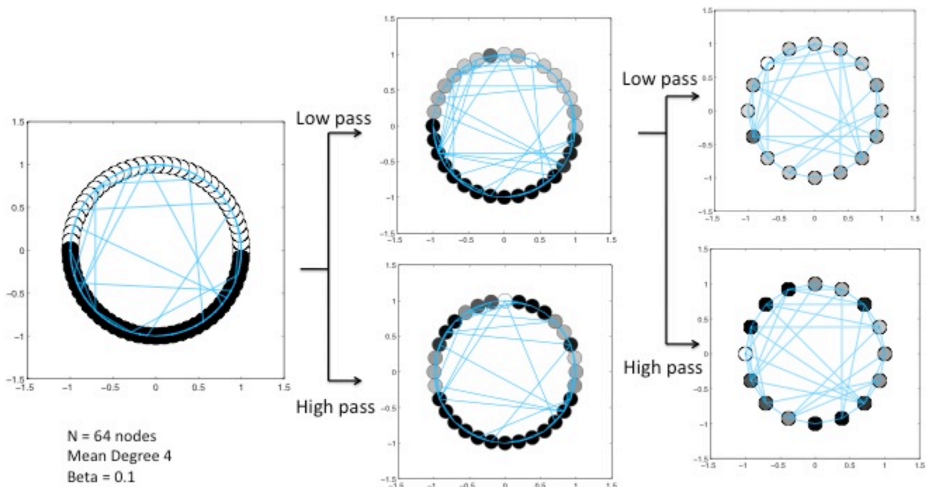


Figure 8.13: Multiscale decomposition of a synthetic Watts-Strogatz graph with $N = 64$, mean degree 4 and $\beta = 0.1$, using 4-stage Lattice filters and the *Kron*-reconnection scheme.

tion scheme should qualitatively retain the structure of the graph. For example, if the graph signal is sparse in the GFT domain of the original graph, then we might also expect it to be sparse in the GFT basis defined by the downsampled graph. A deeper study of the reconnection strategies and their effects on the GFT basis is warranted.

Fig. 8.13 shows the decomposition with the filters now replaced by Lattice filters. The first stage of the decomposition seems to be relatively better than the second stage. The low-pass output in the second stage is not as good a representation of the signal as that obtained with Spline-like filters. The original graph is closer to a circulant graph than the downsampled graph in the first stage. Thus the Lattice filters in the first stage are better designed than those in the second stage since, we have not optimized our Lattice filters for different shifts.

8.6 Chapter highlights

- Linear Shift-Varying filters on general graphs can be decomposed into a bank of shift-invariant filters on the individual circulant graphs in the circulant decomposition of the given graph.
- Multiresolution filter bank structures such as the Laplacian pyramid, Spline-like filter banks and Lattice structures can be extended to general graphs retaining some of the properties like perfect-reconstruction and critical-sampling.
- The main drawback of the filter banks compared to their circulant analogues is that some of these filters (GWS-I and Lattice structures) are not necessarily diagonalizable by the eigenvectors of the Laplacian matrix which poses difficulties in the filter design while optimizing for the filter coefficients.
- The simple Spline-like filters and GWS-II are amenable to tractable design. The filter coefficients can be optimized to approximate desired responses. These are graph-independent designs.

Part IV

Applications

Chapter 9

Wavelet Regularized Graph Semi-Supervised Learning

9.1 Machine Learning—a brief introduction

In this era of Big Data, there is a surge of research interest in developing efficient and accurate algorithms for machine learning (ML) tasks like classification and prediction. In the most abstract form, a classical ML algorithm aims to estimate the labels $\{y_i\}$ of a given set of data $\{\boldsymbol{a}_i\}$. If the labels are discrete, then the problem is known as classification; for instance, if the task is guessing the genre of a novel, then the labels $\{y_i\}$ would represent different genres, like mystery and romance, while the data points $\{\boldsymbol{a}_i\}$ would represent different novels (concretely $\boldsymbol{a}_i$ is a feature vector extracted for novel i) in the dataset. In the continuous case, this problem is called regression. Depending on the prior information on the data, we have different classes of problems. Semi-supervised learning (SSL) is a class of ML problems wherein we are given limited labeled data, $\{\boldsymbol{a}_i, y_i\}$ and a potentially large set of unlabeled data, $\{\boldsymbol{a}_i\}$. The goal is to predict the label of new incoming data $\boldsymbol{a}$. Numerous algorithms such as support vector machines, regularized least squares, etc. have been proposed, and the reader is referred to [50] for a comprehensive overview of the existing research work.

Meanwhile, since long before the onset of modern ML, the signal processing community has been tackling the problems of estimation and prediction as applied to classical domains such as speech and images. Many ML algorithms (e.g. expectation-maximization [51]) have their roots in signal processing. Numerous ML algorithms have been proposed to tackle graph-structured data, and it is not surprising to expect some overlap between ML and the methods presented here. A signal processing interpretation of ML algorithms can help foster new methods and algorithms, while providing insight as to when certain algorithms work better than others.

9.1.1 Semi-Supervised Learning

Many algorithms have been proposed for SSL. We present some of the techniques here and refer the reader to existing literature [50] for a more comprehensive overview of the field. The naive approach to SSL would be to discard the unlabeled data and determine a classification function based only on the labeled dataset; this is known as supervised learning. However, when the amount of labeled data is very limited, this approach can lead to significant accuracy losses. One of the earliest proposed approaches for SSL is self-training or bootstrapping. This technique starts by estimating a classifier using only the labeled dataset. The unlabeled dataset is then classified using the designed classifier. The classified data points with higher confidence are taken as part of the labeled dataset and the process is iterated. Other popular approaches include transductive support vector machines [52], co-training [53], expectation-maximization [51].

An important class of SSL algorithms known as graph-based SSL (GSSL) has received much attention in the recent ML literature [50]. These algorithms base predictions on underlying graph structures, which emerge naturally in some applications, and are data-generated in others. Many of the GSSL algorithms can be interpreted from a graph signal processing viewpoint. In particular, many of these algorithms use *regularizer* terms to impose constraints of label smoothness over the graph; these terms can be viewed as instances of LP and HP filters defined in the GFT domain [54].

Further, many of the existing application domains such as social networks, present graph datasets spanning millions of nodes and it is imperative to have algorithms that achieve substantial order reduction. Multiresolution analytical tools such as wavelets present an important approach towards this end. In particular, critically sampled wavelet designs are necessary to keep the dimensionality under control. In the context of GSSL, wavelet based regularizers can be used for datasets that have a better representation in the wavelet basis than a Fourier basis. The focus of this chapter is the development of a GSSL algorithm using a regularizer based on the critically-sampled wavelets we have discussed in Chapter 8. We explore datasets for which wavelet-based regularizers work better than Fourier-based regularizers and compare the performance of our algorithm with some standard GSSL algorithms on synthetic and real-world datasets. The datasets are chosen to exemplify the performance gains of graph wavelets over Fourier and vice-versa.

9.2 Graph Semi-Supervised Learning

GSSL algorithms are a class of ML algorithms that start with an underlying graph on the data points. Each datapoint (whether labeled or not) corresponds to a node of this graph. Weighted edges reflect the degree of similarity between the connected data points. The assumption is that the labels vary smoothly over the graph. The structure of the underlying graph depends on the application. In some applications, the structure is pre-determined. For example, in the case of labeling blog data [55], one could use the underlying blogroll network—two blogs are connected if they refer to one another either in the blogroll or via hyperlinks embedded in the blog itself. In many other cases, the underlying graph is built using a suitable distance measure on the features of the data.

Different distance measures are used in different applications. For example, gaussian kernels [56] assign a weight $e^{-\beta||\boldsymbol{a}_i-\boldsymbol{a}_j||^2}$ to the edge between nodes i and j. Cosine distance based on tf-idf vectors are typically used in document classification [57]. Other graph constructions include k-nearest neighbor graphs, exponential weighted graphs etc.

There is no clear consensus on how to obtain the graph, and it is largely application-dependent. Most existing algorithms build a graph using a suitable distance measure and then design the classification algorithm using the defined graph. We do not address the graph design question and assume that an underlying graph is given or we build it using one of the distance measures suitably chosen depending on the application of interest.

9.2.1 Graph Semi-Supervised Learning Algorithms

A GSSL algorithm can be formulated as a solution to an optimization problem. The goal is to find an optimal classification (label-prediction) function that minimizes a suitably chosen penalty function. Thus, if $\{\boldsymbol{a}_i, y_i\} \in (\mathcal{A}, \mathcal{Y})$ are the data points and corresponding labels, then classifier f is a function that maps the data points to the labels, i.e.,

$$f : \mathcal{A} \rightarrow \mathcal{Y}, \tag{9.1}$$

$$f(\boldsymbol{a}_i) = y_i. \tag{9.2}$$

We will use $f(i)$ to denote $f(\boldsymbol{a}_i)$. For algorithmic purposes, $\boldsymbol{f}$ can also be thought of as a matrix with rows indexed by the data points and columns indexed by the labels. The $(i,j)^{\text{th}}$ entry is 1 if the i^{th} datapoint is labeled j and is 0 otherwise.

The objective function in the optimization contains two terms. The first term is a loss function that penalizes deviation of the classification function's prediction from observed node labels. Typically, the loss function is defined by a standard classifier. For example, a least-squares cost function is used in the case of Regularized Least-Squares (RLS) algorithms; hinge loss is used for Support Vector Machine (SVM) classifiers [50]; and so on. The second term is a regularizer that penalizes non-smoothness of the function over the underlying graph. Different regularizers give rise to different GSSL algorithms. In the following section, we review some of the GSSL algorithms proposed in the literature.

Graph Mincut based learning

One of the earliest GSSL algorithms is based on *Graph Mincut learning*, proposed by Blum et al. [58]. The algorithm searches for a label-prediction function $\boldsymbol{f}$ that minimizes the objective function,

$$\underbrace{\infty \sum_{i \in \Xi} (f(i) - y_i)^2}_{\textit{loss function}} + \underbrace{\gamma \overbrace{\sum_{i,j} w_{ij} (f(i) - f(j))^2}^{= \boldsymbol{f}^\mathsf{T} \mathbf{L} \boldsymbol{f}}}_{\textit{regularizer}}, \tag{9.3}$$

where w_{ij} are the weights on the edges of the underlying graph, γ is the regularizer tradeoff parameter and Ξ is the labeled dataset. The loss function with the ∞ pre-factor forces the predicted labels to exactly match the observed nodes and penalizes deviation of labels on adjacent nodes. One could use a softer penalty by removing the ∞ pre-factor.

Local Global Consistency (LGC)

The objective function here is given as follows [59]:

$$\sum_{i \in \Xi} (f(i) - y_i)^2 + \gamma \boldsymbol{f}^\mathsf{T} \mathbf{L}_n^\mathsf{S} \boldsymbol{f}, \tag{9.4}$$

where $\mathbf{L}_n^\mathsf{S}$ is the symmetric normalized Laplacian matrix.

Tikhonov Regularization

Here the regularization is based on different powers of the Laplacian matrix [60]. The objective function is given as follows:

$$\frac{1}{|\Xi|} \sum_{i \in \Xi} (f(i) - y_i)^2 + \gamma \boldsymbol{f}^\mathsf{T} \mathbf{L}^p \boldsymbol{f}, \tag{9.5}$$

Manifold Regularization using Laplacian

This approach was proposed by Belkin et al. [61] and can be viewed as a generalized form of the above approaches. It employs two regularization terms:

$$\frac{1}{|\Xi|} \sum_{i \in \Xi} V(\boldsymbol{a}_i, y_i, f(i)) + \gamma_A ||\boldsymbol{f}||_K^2 + \gamma_I ||\boldsymbol{f}||_I^2, \tag{9.6}$$

where V is an arbitrary loss function, K is a 'base kernel', e.g. linear or radial basis function and I is the regularization term imposed by the manifold such as the Laplacian.

The regularizer terms, as in Graph Mincut based learning, can alternatively be written as $\boldsymbol{f}^{\mathsf{T}}\mathbf{L}\boldsymbol{f}$ where $\mathbf{L}$ is the Laplacian of the underlying graph. This can be expanded as follows:

$$\boldsymbol{f}^{\mathsf{T}}\mathbf{L}\boldsymbol{f} = \boldsymbol{f}^{\mathsf{T}}\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathsf{T}}\boldsymbol{f} = \sum_{k=1}^{N} \lambda_k F^{\mathsf{G}}(k)^2, \tag{9.7}$$

which is essentially the energy in the frequency domain of the signal $\boldsymbol{f}$ after suitably filtering it. More generally, we can use a regularizer of the form $\sum_{k=1}^{N} g(\lambda_k) F^{\mathsf{G}}(k)^2$, where $g(\cdot)$ is a suitably-defined HP/LP filter depending on the polarity of the regularizer term in the objective function. For Graph Mincut learning, the filter is HP with $g(\lambda) = \lambda$. The Local Global Consistency (LGC) algorithm uses a normalized Laplacian regularizer [59]. The Tikhonov regularizer [60] uses $g(\lambda) = \lambda^p$ for some $p \geq 1$; the Diffusion Kernel classifier [62] uses the filter $g(\lambda) = e^{-\lambda/\beta}$; and the Regularized Gaussian Process Kernel [56] uses the filter $1/(\lambda + \beta)$. A extensive overview of this literature is available in [50].

Based on this observation, using the tools developed for graph signal processing, we can imagine designing filters systematically so they are tuned to different datasets. This requires that we identify and study the filter characteristics that must be optimized for a given application—an interesting research direction in its own right. On the other hand, it is well known in classical signal processing that certain signals are captured better by wavelets than by the Fourier basis. Hence, a natural question is whether a regularizer based on graph wavelets would capture the characteristics of certain graph datasets better. This is the focus of the remainder of this chapter.

9.2.2 Wavelet Regularized GSSL

So far, the algorithms we have discussed impose regularization in the GFT domain. We know in classical signal processing that for certain kinds of signals, wavelets provide a more compact representation. Hence it might be interesting to replace the Fourier basis by a wavelet basis in the regularizer. In that event, we might want the label prediction function to be sparse with respect to a suitably defined wavelet basis on the graph. This could be useful for datasets where the labels do not smoothly transition over the graph, but might have drastic boundaries of change. Intuitively, wavelets are good at capturing these kinds of patchy, localized transitions, while standard Fourier decompositions (for instance) are not.

A Wavelet-Regularized Least-Squares (WLRS) algorithm imposes a sparsity regularizer constraint on the wavelet decomposition of the label-prediction function $\boldsymbol{f}$—the underlying graph signal in this case. If $\widetilde{\boldsymbol{f}}$ is the wavelet basis representation of $\boldsymbol{f}$, then a WRLS algorithm with a L1-penalty to impose sparsity in the wavelet domain has the following optimization formulation,

$$\min_{\boldsymbol{f}} \sum_{i \in \Xi} (f(i) - y_i)^2 + \gamma ||\widetilde{\boldsymbol{f}}||_1. \tag{9.8}$$

The wavelets we use in our experiments will be the simple Spline-like wavelets described in Section 8.3.1 and therefore we shall term the proposed algorithm as spline-like WRLS. The proposed formulation is similar in spirit to the regression-based formulation used in [63]. The main distinction of our work is that the wavelets we use are critically sampled, whereas those in [63] are oversampled representations. Additionally, we also provide simulation results on real world datasets.

9.3 Experimental results

In this section we explore the performance of the proposed spline-like-WRLS GSSL algorithm on some synthetic and real-world datasets. All the graph datasets are unweighted. A four-level wavelet decomposition with equal weights is used. We compare the performance to two standard GSSL algorithms: Local Global Consistency [61] and Diffusion Kernel based classifier [62]. The value of β in the diffusion operator $g(\lambda) = e^{-\lambda/\beta}$ is taken to be 2. The regularizer tradeoff parameter γ is taken to be 0.01, unless specified otherwise.

We also compare the performance to two existing graph wavelet designs proposed in the literature: Diffusion Wavelet Transform (DWT) [42] and Spectral Graph Wavelet Transform (SGWT) [64]. We use the default parameters in the toolboxes provided for each transform. The number of scales (levels) is taken to be five for each of the wavelet designs. We solve the L1-regularized optimization problem using an iterative hard-thresholding algorithm [65], with a sparsity threshold of 20 for all the algorithms.

The key observation in our experiments is that wavelet-regularized algorithms work well when, rather than globally smooth, the dataset is locally smooth with potentially sharp transitions. The Laplacian imposes global smoothness constraints and performs better on datasets where label transitions are gradual. This is often the case when the underlying graph is constructed from data using a similarity metric. However, some graph datasets have an underlying graph provided by the application. In these cases, it is likely that the labels will be similar in local neighborhoods, but they can vary sharply (e.g., in electoral datasets). We expect a wavelet basis to capture these variations better, and this intuition is confirmed in our experiments.

We start with a synthetic dataset designed to illustrate our reasoning explicitly. The dataset is a collection of 747 major cities in the continental United States grouped with respect to the individ-

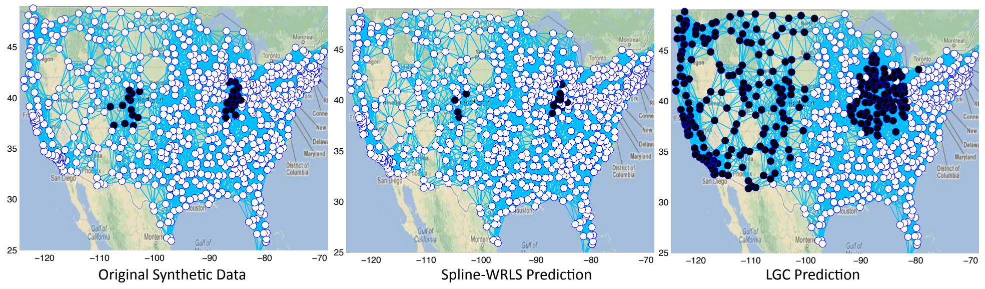


Figure 9.1: Label prediction of the spline-like WRLS algorithm as compared to the LGC semi-supervised classifier in one of the extreme cases. The LGC tends to diffuse to a larger extent than the spline-like WRLS since it only has a local view of the data as compared to the spline-like which has multiscale views of the data.

# Tr.	LGC (%)	Diff. Ker RLS (%)	spline-like WRLS (%)
10	63.52	58.1	**66.32**
30	69.11	58.52	**74.22**
50	72.48	57.75	**78.82**
70	74.30	58.96	**81.16**
90	74.62	58.09	**80.82**
110	75.93	58.63	**81.24**
130	76.35	59.59	**84.96**

Table 9.1: Accuracy in prediction for the Synthetic data set. Each of the rows are obtained after averaging over 100 realizations of randomly chosen training points. One can observe that the spline-like WRLS consistently performs better.

ual states they belong to. The underlying graph is built by connecting cities that are less than a particular distance apart. All the cities in two states - Colorado and Indiana, are assigned a label 0 and the rest of the cities are assigned a label 1. The training data is created by choosing, randomly, an equal number of data points for each label.

Fig. 9.1 shows the original dataset and the predicted dataset for a training set of 32 points. LGC tends to diffuse to a much larger extent than the spline-like WRLS in certain cases like the one shown in the figure. Hence it cannot capture localized changes well. The spline-like basis in turn seems to capture the localized variations better.

Table 9.1 shows the prediction accuracies (percentage of the labels predicted correctly) as a function of the number of training data points for the different algorithms. The spline-like-WRLS

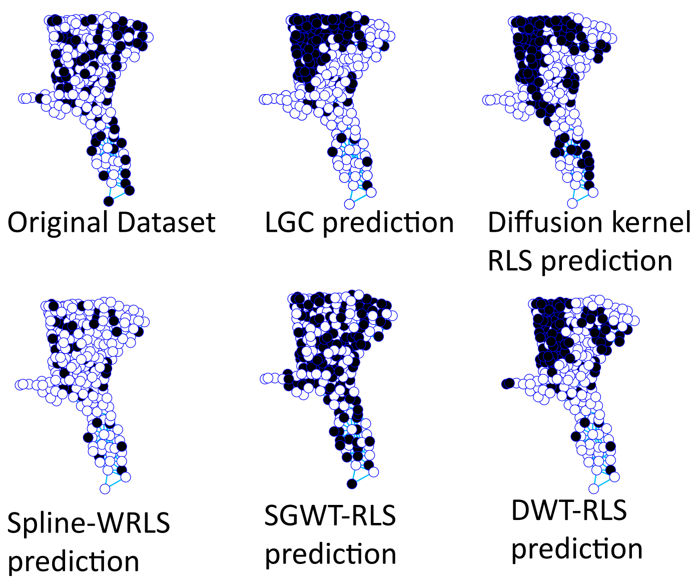


Figure 9.2: Label prediction for the Florida electoral dataset.

# Tr.	LGC (%)	Diff. Ker RLS (%)	spline-like WRLS (%)	SGWT RLS (%)	DWT RLS (%)
10	52.52	52.60	**55.01**	52.73	51.70
30	56.52	57.12	**61.01**	56.79	55.15
50	59.60	60.38	**64.83**	60.29	57.65
70	64.64	64.31	**70.17**	64.66	63.27
90	67.48	68.41	**77.10**	68.54	64.38

Table 9.2: Accuracy in prediction for the Florida electoral data set. Each of the rows is obtained after averaging over 100 realizations of randomly chosen training points. The spline-like RLS consistently performs better.

seems to perform consistently better than the other algorithms. We did not apply the DWT and SGWT-RLS to this dataset since the run-time is prohibitively slow.

We also carried out experiments with a real-world electoral dataset from the counties in Florida, Georgia, and South Carolina. The graph is built by joining two counties if they share a common border. A set of 272 data points is available. Table 9.2 shows the prediction accuracies for this dataset. Fig. 9.2 shows the label prediction for this dataset. As before, the Laplacian-based approaches tend to impose a global smoothness on the labels. The spline-like wavelets seem to capture the localized variations. The performances of DWT and SGWT are similar to those of the Laplacian-based approaches for this dataset.

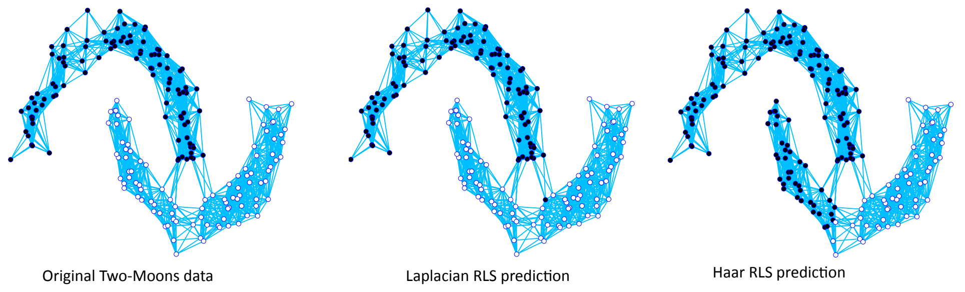


Figure 9.3: Label prediction for the Two-Moons synthetic dataset with 10 training samples, that exemplifies the inaccuracy of the wavelet basis in capturing the global smoothness. The wavelet regularizer imposes local smoothness with sharp boundary changes as we observe in the dataset.

# Tr	Lap RLS (%)	spline-like WRLS (%)	SGWT RLS (%)	DWT RLS (%)
10	**98.45**	60.63	95.88	53.32
20	**99.86**	61.60	97.52	54.32
30	**99.98**	63.09	97.84	57.72
40	**100.00**	67.31	98.06	59.76
50	**100.00**	68.05	98.13	62.06

Table 9.3: Accuracy in prediction for the Two-Moons dataset. Each of the rows is obtained after averaging over 100 realizations of randomly chosen training points. The spline-like wavelet does not fare well in this case. The SGWT however seems to be better, but is outperformed by the Laplacian regularizer.

Let us now look at a dataset that is globally smooth to begin with and compare the performance of the algorithms. The 'Two-Moons' dataset [54] is a 200 node standard synthetic dataset that is globally smooth as observed in Fig. 9.3. The Laplacian regularizer captures the global smoothness of the graph and is able to have a good prediction accuracy for minimal training set as well. Table 9.3, shows the error performance for the different algorithms. Clearly spline-like WRLS and DWT-RLS perform worse than Laplacian RLS. SGWT-RLS performs very closely to Laplacian RLS.

We also compare the algorithms on a blog dataset—AGBlog [66]. This is a two-class classification problem where the blogs are labeled either *liberal* or *conservative*. For faster execution, we chose only a subset of the nodes in the blog. Table 9.4 shows the performance of the various algorithms. The diffusion kernel-based RLS performs the best. The wavelet-based approaches do not provide any advantage over the Laplacian methods in this case, possibly due to the global smoothness of the data.

# Tr.	LGC	Diff. Ker RLS	spline-like RLS	SGWT RLS	DWT RLS
10	72.19	**87.73**	79.73	82.00	78.89
25	78.50	**90.27**	85.16	84.02	85.18
40	87.32	**91.29**	87.74	86.25	87.89
55	89.89	**92.28**	89.49	87.91	89.58
70	92.36	**93.40**	91.33	89.56	91.08

Table 9.4: Accuracy in prediction for the AGBlog dataset with 160 randomly-chosen connected nodes from the original dataset. Each of the rows is obtained after averaging over 100 realizations of randomly-chosen training points. The Diffusion Kernel RLS performs the best in this case.

Our experiments are not comprehensive, but the main aim of this work was to understand datasets for which a wavelet based regularizer works better than a Laplacian based regularizer. The experimental results suggest that the regularizer to be used largely depends on the characteristics of the dataset. In particular, wavelet regularized algorithms work better for datasets that are locally smooth with sharp boundary transitions.

However one should treat the results with some caution. Unfortunately there are many parameters to be optimized for each of the algorithms (e.g. regularizer tradeoff parameter, number of wavelet scales to use, sparsity assumption, etc.) and we do not have a systematic way of optimizing them at this stage. Further we also observed that the coefficients with respect to those wavelets whose support is contained on unlabeled nodes is always zero as noted in [63]. This requires us to go to higher scales with a larger spread or have a larger training set. However we know that wavelets also capture consistent data characteristics at different scales that can be used to alleviate this issue. The authors in [63] propose a regularizer that attempts to impose consistency across different scales. This needs further exploration.

Our datasets are also limited in size mainly due to the computational inefficiency of the algorithms. For large datasets, computing the different wavelet transforms, in particular the DWT and SGWT, is practically restrictive at this stage. We have chosen real world datasets that allow us to execute the different algorithms in a reasonable time frame. Developing computationally efficient wavelet transforms is an important topic which needs to be addressed.

9.4 Chapter Highlights

- Machine Learning algorithms typically consider the problem of predicting labels for datasets given features for each data point and some training data.
- Graph Semi-Supervised Learning (GSSL) algorithms are a class of machine learning algorithms that impose a graph structure on the data (or inherently imposed by the application). They can be formulated as an optimization problem whose objective function has two parts—a loss function that imposes consistency of predicted labels on the training set and a regularizer that imposes smoothness of the label prediction function on the graph.
- GSSL algorithms have an interpretation in the graph signal processing domain. In particular, the regularizers used in different GSSL algorithms can be viewed as different filters designed in the GFT domain.
- The Fourier based regularizers are now replaced with a wavelet based regularizer. Sparsity is imposed in the wavelet domain representation of the label prediction function.
- Experiments seem to suggest that wavelet-based regularizers are better suited for datasets that are locally smooth as opposed to Fourier based regularizers that work for globally smooth datasets.
- Wavelet based regularizers inherently have the issue that coefficients of wavelets whose support is contained on unlabeled nodes is always zero. Regularizers that capture the dependencies across multiple scales need to be designed to alleviate this issue.

Part V

Related Work and Open Research Problems

Chapter 10

Related Work

In this chapter, we outline the literature on graph signal processing that have either emerged under the same topic or in related areas dealing with similar problem formulations. The foundation for much of the research in this area stems from the interesting spectral properties of the Laplacian, which have been well studied in the literature [41, 67, 68]. Earlier work in computer graphics [4–6] used mesh processing to analyze 3-D object structures and exploited Laplacian matrix properties to define graph-dependent filters for smoothing. Courant's nodal domain theorems [16] bring out the frequency interpretation of the eigenvectors of the Laplacian and are one of the motivations for defining the notion of a GFT. The formal definition and analysis of the GFT appear in more recent works [64, 69, 70] that are motivated by the deluge of graph-structured data seen in modern day applications.

Most of the recent work in this direction has focused on defining basic operations on graph signals (e.g., shifting, sampling, and filtering) and understanding the properties of the GFT in relation to these operations. As we have mentioned in Chapter 3, a major challenge in defining fundamental operations like signal shifting is a general graph's irregular structure; typically, the graph looks different from different nodes. Hence, it is unclear how to "shift" or "sample" a signal on the graph. Hammond et al. [64] resort to defining these operations in the spectral domain, motivated by the relation between these operations in time domain and the corresponding Fourier domain. For example, convolution of two signals on the graph is defined as the point-wise multiplication of the corresponding GFT's. Similarly, shifting is defined as a convolution with a delta signal and so on. Sandryhaila and Moura [8] on the other hand define these operations using the adjacency matrix. In particular, shifting is defined as multiplying the input signal with the adjacency matrix and filtering operations are defined as polynomials in the adjacency matrix. Shifting being a basic operation, this definition affects other basic operations like filtering. Further, [8] uses a slightly different definition of the GFT by considering the Jordan normalized form of the adjacency matrix rather than the Laplacian.

Many of these works have also focused on defining and understanding the effect of sampling signals on graphs and in some cases, defining the underlying graph after sampling. Hammond et al. [64] define sampling based on the eigenvector corresponding to the largest eigenvalue. The underlying downsampled graph is defined based on the *Kron*-reduction strategy. Narang et al. [69] focus on bipartite graphs and show that sampling on a bipartite graph by retaining the set of nodes in one group of the bipartite graph, creates an aliasing in the GFT domain similar to that in regular time domain. They exploit this property to design critically sampled perfect reconstruction filter-banks on bipartite graphs. There have also been fundamental results by Issac Pesenson and Meyer Pesenson [71] and Jorgensen et al. [72] on sampling theorems for signals on arbitrary graphs. In particular, they characterize the bandwidth of a signal that can be supported on a graph given a subset of nodes that are removed. However these results are quite conservative and we find that for circulant graphs, the bandwidth provided by our theorems for the defined sampling patterns are much larger than that given by their results.

Multiscale representation of graph signals and wavelet design for graph signals has received significant attention in the recent literature on graph signal processing. Agaskar and Lu [26] have derived uncertainty principles for general graphs that describe the tradeoff between the signal support in the graph domain and the spectral domain. However, these results are not sufficiently tight for all graphs. Wang and Ramchandran [73], define localized filters for sensor networks that allow for a multiscale representation as well as perfect reconstruction. Crovella and Kolaczyk [74] define wavelet-like functions that are scaled and dilated versions of a single generating function. Spectral graph wavelets proposed by Hammond et al. [64], have a similar flavor where the functions are defined in the spectral domain. There is related work in the computer graphics community on multi resolution signal processing for meshes [75–77]. These largely deal with triangulated meshes which have additional structure since they have been defined based on an underlying smooth surface with an euclidean embedding and the structure is used to define multi resolution operations. However, optimality properties in relation to the GFT basis have not been studied in depth.

Diffusion wavelets developed by Coifman and Maggioni [42, 78] consist of locally supported functions on graphs that are obtained by suitably modifying powers of a diffusion operator like the adjacency matrix. All these methods provide an oversampled multi-scale representation of the graph signal. The result of Narang et al. [79], for bipartite graphs can be extended to general graphs [31] by having a bipartite decomposition of the original graph which provides a critically sampled multi-scale representation. However, they do not have a clear way to extend their analysis to multiple scales, since the underlying graph after sampling is not defined. They resolve this by using heuristics to reconnect the graph. There are also other works using lifting-based constructions [80, 81] and tree-based constructions [82] for multiscale analysis. A summary of some of the literature in this area can be found in [9, 38].

There has also been quite some interest in applying these graph signal processing techniques to real world applications. The earliest work by Taubin et al. [4] focussed on using Laplacian-based methods for mesh signal processing. The work by Coifman and Maggioni [64] used diffusions wavelets for text classification and was subsequently explored by Wang and Mahadevan [83] for document classification. Crovella and Kolaczyk [74] used wavelets for network traffic analysis. Narang et al. [80], used lifting-based constructions and bipartite constructions for image and video processing. Aliaksei and Moura [8], used their theory to develop filters for prediction over cellular networks.

The mathematical biology community has related work in dealing with "fitness landscapes" that visualize the relationship between genotypes and reproductive successes [84]. Fourier functions based on the Laplacian have been defined for these landscapes and there has been quite some work analyzing the properties of these basis and design of computationally efficient algorithms [85–87]. These are mostly in the context of optimization wherein the problem of interest is to optimize over these landscapes which are highly non-convex. A Fourier-theoretic analysis of these landscapes provides some insights in understanding the behavior of heuristic algorithms.

There is also quite a lot of work on Fourier analysis for regular domains like groups. In particular, there is some literature on Cayley groups [25] that are closely related to Cayley graphs of which circulant graphs are a subset of. However, the group structure here is related to the node connectivity and not the signals defined on the nodes as such (see Section 3.2.2 for a more detailed discussion). A more detailed exploration in this direction with a group theoretic approach to our problem might provide some insights. There is also some related work on Fourier and wavelet analysis for regular manifolds like spheres [88, 89].

Recent progress notwithstanding, much work remains in the quest for a unified and comprehensive signal processing theory for graphs. Our approach has been motivated by LTI signal processing, which has had considerable impact in classical signal processing. We view general graphs as LSV systems analogous to LTV systems in the classical theory.

A corpus of research literature deals with LTV systems through the lens of LTI systems [10–14]. Accordingly, we first study *circulant* graphs—a class of graphs amenable to LSI operations. Existing work [90] that discusses the shift-invariant nature of circulant graphs does not explore the properties in substantive depth. We have shown that even certain fundamental shift-varying operations, such as sampling, are natural on these graphs. We define these shifting and sampling formally and derive the properties of the GFT in relation to these operations. We then decompose general graphs into a bank of circulant graphs analogous to decomposing a LTV filter as a bank of LTI filters. Perfect reconstruction filter banks leading to multiscale representations are designed using the developed concepts. The following papers contain parts of work that have appeared in this thesis [91–94].

Chapter 11

Open research problems

The area of graph signal processing is relatively new and there are many open research problems. Most of the research work in the recent past, including this thesis, has taken the approach of developing signal processing tools for graphs by drawing analogies with classical signal processing. Applications are now being explored to see where these tools fit. On the other hand, a lot of research work in the early past have been motivated by applications such as in computer graphics [5], sensor networks [73], network analysis [74] and mathematical biology [84]. A lot of the tools in these areas have been independently developed and have not been explored in-depth. The current research work in this area, including the work presented here, has focused on developing a unified theory for the underlying problem facing all these applications. Given this perspective, it is now important to get back to understanding challenges posed by different applications—old as well as new, that can guide the development of tools required to solve these specific problems and further provide us with a metric to evaluate these tools.

Many of the recent applications such as social networks, pose challenges of scale dealing with millions of nodes. Complexity reduction is imperative for these applications wherein multi resolution analysis can be a powerful tool. Intuitively, we need to obtain summaries of large graph datasets at multiple scales so that, depending on the application, queries could be evaluated on the summarized dataset. We would expect that each scale be "representative" of the original dataset—both the graph and the signal associated with it. Given that the queries would be evaluated on the summary graph at a lower scale, it is necessary that the graph signal at the lower scale be optimized to provide a good approximation with respect to the metrics that are relevant to the query of interest.

So far the work on multi resolution analysis in this literature, has focussed on metrics borrowed from classical signal processing. For example, emphasis on perfect-reconstruction might not be required. Mean-squared error might not be the right evaluation metric to be optimized for in these applications. Thus it is important to take a re-look at the multiscale filter designs from an application perspective. This will also help evaluate algorithms arising from this new perspective by comparing them with existing machine-learning algorithms for some of these tasks.

There are many applications where prediction is important i.e. data available in one part of the graph would need to be extrapolated to the rest of the graph. The learning application discussed in Chapter 9 can be viewed in this category. The natural question to ask here is, "Could one design prediction filters analogous to least-mean-squared (LMS) filters and Wiener filters as defined in classical signal processing?" To enable this, it is important to define random variables on graphs (a related notion is that of graphical models) and develop a statistical signal processing theory for random signals on graphs. The random models developed should reflect real-world applications.

There are various other obvious generalizations that can be considered such as vector-valued signals on each node, time-varying signals associated with each node and so on. Further, the graph structure itself could vary over time. These problems can be pretty challenging to address and newer analysis methods are warranted. However, there is still quite a lot of open research problems even in the case of scalar signals defined over graphs which we discuss below.

The basic notion of a GFT has been defined with respect to the eigenvectors of the Laplacian matrix of a graph. The eigenvalues are used to order the eigenvectors from low-frequency to high-frequency. Many existing works in the literature treat the eigenvalues themselves as frequencies while we do not make this assumption. It is important to understand the physical significance of the eigenvalues in this context. From the double-differential operator analogy of the Laplacian matrix, the eigenvalues seem to be more related to the squares of the frequencies. Further, for a ring graph, the eigenvalues are a non-linear function of the classical discrete Fourier frequencies. The treatment of the eigenvalues as frequencies would also impact filter design. Secondly, when we consider operations like sampling, we would like to know how the spectrum of the downsampled signal on the smaller graph relates to that on the bigger graph which would also help evaluate different reconnection strategies. This might be related to the behavior of the eigenvalues on the original and downsampled graphs. Hence an in-depth study of eigenvalues is warranted in the context of graph signal processing.

For many graphs, we have repeated eigenvalues, in which case the eigenvectors themselves are not unique. This is true in the case of circulant graphs itself. We saw that a particular choice of eigenvectors simplified analysis while deriving many of the properties. Thus a question of interest is, "How does one define a unique basis set for the GFT?". Further most of the theory has been developed for undirected graphs. It is important to extend the concepts to directed graphs. Weighted graphs also need to be studied in depth.

In the case of circulant graphs, it would be interesting to develop a group-theoretic analysis given the connection between Cayley graphs and Cayley groups. We have discussed the basic principles underlying filter design for circulant graphs. Interesting topics include extending specific classes of filters from classical signal processing. One should however keep the applications in mind while designing these filters. We have discussed analogues of first-order spline filter banks for graphs.

One could now think of generalizing higher order spline filter designs that offer more flexibility in terms of their response.

Our treatment of general graphs is not as extensive as that of circulant graphs. The primary approach has been to decompose general graphs into a bank of circulant graphs. However, the main issue with the decomposition is that it is not invariant to the node ordering. Obtaining invariant decompositions in general can be a hard problem. There are some algorithms for approximate cycle decompositions of graphs. It is unclear whether one should consider decomposition of graphs or deal with the original graph directly. Classical image processing has largely focussed on separable processing which is analogous to decomposing a grid graph into multiple path graphs. We believe in a similar philosophy for general graphs as well since it is difficult to directly work with irregular structures.

Most of the work in the literature has so far considered fundamental operations and filter designs for graphs. There is very little work tackling with complexity issues which has been a primary reason why these techniques have not been applied to very large scale datasets, the norm in modern day applications. Hence it is very important to develop algorithms and filter designs that are computationally efficient and amenable to large scale processing.

Overall this is an exciting and emerging field with many open research problems. There is much to be done both on the theoretical as well as the application front. We envision that in the future, signal processing theory would be taught from the perspective of graphs.

Table of notations

Notation	Meaning
G	graph
V	vertex set of the graph
E	edge set of the graph
N	number of nodes in the graph
$\mathbf{A}$	graph adjacency matrix
$\mathbf{L}$	graph Laplacian matrix
$\mathbf{D}$	diagonal matrix of node degrees
$\mathbf{A}^{\mathsf{S}}$	symmetric normalized adjacency matrix
$\mathbf{A}^{\mathsf{RW}}$	random-walk normalized adjacency matrix
$\mathbf{L}^{\mathsf{S}}$	symmetric normalized Laplacian matrix
$\mathbf{L}^{\mathsf{RW}}$	random-walk normalized Laplacian matrix
$\boldsymbol{x}, \boldsymbol{y}$	signals on graphs
Λ	diagonal matrix containing the non-negative real eigenvalues of $\mathbf{L}$
λ_k	k^{TH} eigenvalue of the Laplacian matrix $\mathbf{L}$
$\mathbf{U}$	matrix of orthonormal eigenvectors of the Laplacian matrix
$\mathbf{U}_{\mathsf{S}}$	matrix of eigenvectors of the symmetric normalized Laplacian matrix
$\mathbf{U}_{\mathsf{RW}}$	matrix of eigenvectors of the random-walk normalized Laplacian matrix
$\boldsymbol{u}_k$	eigenvector corresponding to the k^{TH} eigenvalue, λ_k
$\mathbf{X}^{\mathsf{G}}$	graph Fourier transform of the graph signal $\boldsymbol{x}$
$\mathbf{X}^{\mathsf{F}}$	discrete Fourier transform of the graph signal $\boldsymbol{x}$
GFT	Graph Fourier Transform
DFT	Discrete Fourier Transform
S	generating set of the circulant graph
s_k	k^{th} element of the generating set S
$()_N$	$\mod N$
$\mathcal{G}$	group
$\mathbf{F}$	DFT matrix
$\mathbf{P}$	shift matrix defined with respect to the element $s = 1 \in S$
$\mathbf{P}_s$	shift matrix defined with respect to the element $s \in S$
$\boldsymbol{h}$	graph filter coefficients

$\mathbf{H}$	filter matrix corresponding to $\boldsymbol{h}$
$\mathbf{H}^{\mathsf{G}}$	GFT of filter $\boldsymbol{h}$
LTI	Linear Time Invariant
LTV	Linear Time Varying
LSI	Linear Shift Invariant
LSV	Linear Shift Varying
$\mathbf{0}_N$	N-length vector of all-zeros
$\mathbf{1}_N$	N-length vector of all-ones
$\mathbf{I}_N$	identity matrix of size N
$\downarrow_s m$	downsample by a factor of m with respect to the element $s \in S$
$\sigma(\cdot)$	permutation mapping from the DFT to GFT for circulant graphs
$\boldsymbol{a} \circ \boldsymbol{b}$	Hadamard product or element-wise for product between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
$\mathcal{I}_A()$	indicator matrix of $\mathbf{A}$ with zero-diagonal entries
LP	low-pass
HP	high-pass
$\mathbf{H}_{\mathsf{LP}}$	analysis LP filter of a two-channel filter bank
$\mathbf{H}_{\mathsf{HP}}$	analysis HP filter of a two-channel filter bank
$\mathbf{H}_{\mathsf{INV}}$	synthesis filter of a two-channel filter bank
$\mathbf{Q}_k, \widehat{\mathbf{Q}}_k, \widetilde{\mathbf{Q}}_k$	diagonal matrices in the circulant decompositions
diag()	diagonal matrix of the elements inside the argument. If the argument is a matrix, then this represents a diagonal matrix with the elements being the diagonal entries of the input matrix

Table 11.1: Table of notations

Bibliography

[1] B Krishnamurthy, P Gill, and M Arlitt. A few chirps about twitter. In *Proceedings of the first workshop on Online social networks*, pages 19–24. ACM, 2008.

[2] Gsod : Global surface summary of day. *http://www.ncdc.noaa.gov/cgi-bin/res40.pl*.

[3] Å J Holmgren. Using graph models to analyze the vulnerability of electric power networks. *Risk analysis*, 26(4):955–969, 2006.

[4] G Taubin, T Zhang, and G Golub. Optimal surface smoothing as filter design. *Computer VisionECCV'96*, pages 283–292, 1996.

[5] G Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358. ACM, 1995.

[6] H Zhang, O Van Kaick, and R Dyer. Spectral mesh processing. In *Computer graphics forum*, volume 29, pages 1865–1894. Wiley Online Library, 2010.

[7] K P Murphy. *Machine learning: a probabilistic perspective*. The MIT Press, 2012.

[8] A Sandryhaila and J Moura. Discrete signal processing on graphs. 2012.

[9] D I Shuman, S K Narang, P Frossard, A Ortega, and P Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *Signal Processing Magazine, IEEE*, 30(3):83–98, 2013.

[10] G Lohar and A G Wacker. Matrix theory approach to the canonical representation of a class of linear discrete-time time-variant systems. *Circuits and Systems, IEEE Transactions on*, 37(2):303–306, 1990.

[11] G Lohar, D P Mukherjee, and D Dutta Majumder. On a decomposition of 2-d circular convolution. *Pattern recognition letters*, 13(10):701–706, 1992.

[12] C Herley and M Vetterli. Orthogonal time-varying filter banks and wavelet packets. *Signal Processing, IEEE Transactions on*, 42(10):2650–2663, 1994.

[13] R Meyer and C S Burrus. A unified analysis of multirate and periodically time-varying digital filters. *Circuits and Systems, IEEE Transactions on*, 22(3):162–168, 1975.

[14] S M Phoong and P P Vaidyanathan. A polyphase approach to time-varying filter banks. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 3, pages 1554–1557. IEEE, 1996.

[15] Béla Bollobás. *Modern graph theory*, volume 184. Springer, 1998.

[16] E B Davies, G M L Gladwell, J Leydold, and P F Stadler. Discrete nodal domain theorems. *Linear Algebra and its Applications*, 336(1):51–60, 2001.

[17] A V Oppenheim. *Discrete-Time Signal Processing, 2/E.* Pearson Education India, 2006.

[18] C D Godsil and G Royle. *Algebraic graph theory*, volume 8. Springer New York, 2001.

[19] M E Muzychuk and G Tinhofer. Recognizing circulant graphs in polynomial time: An application of association schemes. *Journal of Combinatorics*, 8(1):26–26, 2001.

[20] F Boesch and R Tindell. Circulants and their connectivities. *Journal of Graph Theory*, 8(4):487–499, 1984.

[21] S Toida. A note on ádám's conjecture. *Journal of Combinatorial Theory, Series B*, 23(2):239–246, 1977.

[22] G A Jones and J M Jones. *Elementary number theory*. Springer, 1998.

[23] D J Watts and S H Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442, 1998.

[24] N Jacobson. *Lectures in abstract algebra*, volume 1. van Nostrand New York, 1951.

[25] A Terras. *Fourier analysis on finite groups and applications*, volume 43. Cambridge University Press, 1999.

[26] A Agaskar and Y M Lu. An uncertainty principle for functions defined on graphs. In *SPIE Optical Engineering+ Applications*, pages 81380T–81380T. International Society for Optics and Photonics, 2011.

[27] J Kovacevic, V K Goyal, and M Vetterli. Fourier and wavelet signal processing. 2013.

[28] J Leskovec and C Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2006.

[29] F Dorfler and F Bullo. Kron reduction of graphs with applications to electrical networks. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 60(1):150–163, 2013.

[30] B Ayazifar. *Graph spectra and modal dynamics of oscillatory networks*. PhD thesis, MIT, 2002.

[31] S K Narang and A Ortega. Perfect reconstruction two-channel wavelet filter banks for graph structured data. *Signal Processing, IEEE Transactions on*, 60(6):2786–2799, 2012.

[32] P Stevenhagen and H W Lenstra. Chebotarëv and his density theorem. *The Mathematical Intelligencer*, 18(2):26–37, 1996.

[33] B Alexeev, J Cahill, and D G Mixon. Full spark frames. *Journal of Fourier Analysis and Applications*, 18(6):1167–1194, 2012.

[34] M Mishali and Y C Eldar. Blind multiband signal reconstruction: Compressed sensing for analog signals. *Signal Processing, IEEE Transactions on*, 57(3):993–1009, 2009.

[35] N Biggs. *Algebraic graph theory*. Cambridge University Press, 1993.

[36] P Lancaster and M Tismenetsky. *Theory of matrices*, volume 2. Academic press New York, 1969.

[37] P P Vaidyanathan. Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial. *Proceedings of the IEEE*, 78(1):56–93, 1990.

[38] D I Shuman, M Javad Faraji, and P Vandergheynst. A framework for multiscale transforms on graphs. *arXiv preprint arXiv:1308.4942*, 2013.

[39] G Strang. *Wavelets and filter banks*. Wellesley Cambridge Press, 1996.

[40] G H Golub and Van Loan C F. *Matrix computations*, volume 3. JHU Press, 2012.

[41] F R K Chung. Lectures on spectral graph theory. *CBMS Lectures, Fresno*, 1996.

[42] R R Coifman and M Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

[43] D Dor and M Tarsi. Graph decomposition is np-complete: A complete proof of holyer's conjecture. *SIAM Journal on Computing*, 26(4):1166–1187, 1997.

[44] E Cuthill and J McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172. ACM, 1969.

[45] J Shi and J Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[46] D Gleich. gaimc: Graph algorithms in matlab code. In *Matlab Toolbox*. Matlab, 2009.

[47] E J Candès and M B Wakin. An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):21–30, 2008.

[48] B Mohar. *Some applications of Laplace eigenvalues of graphs*. Springer, 1997.

[49] J L Arrowood Jr and M J T Smith. Exact reconstruction analysis/synthesis filter banks with time-varying filters. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 3, pages 233–236. IEEE, 1993.

[50] X Zhu. Semi-supervised learning literature survey. *Computer Sciences TR*, 1530, 2008.

[51] R A Redner and H F Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM review*, 26(2):195–239, 1984.

[52] T Joachims. *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers, 2002.

[53] Zhi-Hua Zhou and M Li. Semi-supervised regression with co-training. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 1, pages 2–2, 2005.

[54] X Zhu. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, School of Computer Science, 2005.

[55] J Leskovec, M McGlohon, C Faloutsos, N Glance, and M Hurst. Cascading behavior in large blog graphs. *arXiv preprint arXiv:0704.2803*, 2007.

[56] X Zhu, Z Ghahramani, and J Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 20, page 912, 2003.

[57] L D Baker and A K McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103. ACM, 1998.

[58] A Blum and S Chawla. Learning from labeled and unlabeled data using graph mincuts. 2001.

[59] D Zhou, O Bousquet, T N Lal, J Weston, and B Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(753760):284, 2004.

[60] M Belkin, I Matveeva, and P Niyogi. Regularization and semi-supervised learning on large graphs. In *Learning theory*, pages 624–638. Springer, 2004.

[61] M Belkin, P Niyogi, and V Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434, 2006.

[62] R I Kondor and J Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, 2002.

[63] D I Shuman, M J Faraji, and P Vandergheynst. Semi-supervised learning with spectral graph wavelets. *SAMPTA 2011.*

[64] D K Hammond, P Vandergheynst, and R Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[65] T Blumensath. Accelerated iterative hard thresholding. *Signal Processing*, 2012.

[66] F Lin and W W Cohen. Semi-supervised classification of network data using very few labels. In *ASONAM 2010*, pages 192–199. IEEE.

[67] R Merris. Laplacian matrices of graphs: a survey. *Linear algebra and its applications*, 197:143–176, 1994.

[68] B Mohar. The laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2:871–898, 1991.

[69] S K Narang and A Ortega. Downsampling graphs using spectral theory. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 4208–4211. IEEE, 2011.

[70] X Zhu and M Rabbat. Approximating signals supported on graphs. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 3921–3924. IEEE, 2012.

[71] I Z Pesenson and M Z Pesenson. Sampling, filtering and sparse approximations on combinatorial graphs. *Journal of Fourier Analysis and Applications*, 16(6):921–942, 2010.

[72] P E T Jorgensen. A sampling theory for infinite weighted graphs. *Opuscula Mathematica*, 31(2):209–236, 2011.

[73] W Wang and K Ramchandran. Random distributed multiresolution representations with significance querying. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 102–108. ACM, 2006.

[74] M Crovella and E Kolaczyk. Graph wavelets for spatial traffic analysis. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1848–1857. IEEE, 2003.

[75] I Guskov, W Sweldens, and P Schröder. Multiresolution signal processing for meshes. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 325–334. ACM Press/Addison-Wesley Publishing Co., 1999.

[76] P S Heckbert and M Garland. Survey of polygonal surface simplification algorithms. Technical report, DTIC Document, 1997.

[77] M Jansen, R Baraniuk, and S Lavu. Multiscale approximation of piecewise smooth two-dimensional functions using normal triangulated meshes. *Applied and Computational Harmonic Analysis*, 19(1):92–130, 2005.

[78] M Maggioni, J C Bremer Jr, R R Coifman, and A D Szlam. Biorthogonal diffusion wavelets for multiscale representations on manifolds and graphs. In *Optics & Photonics 2005*, pages 59141M–59141M. International Society for Optics and Photonics, 2005.

[79] S K Narang and A Ortega. Local two-channel critically sampled filter-banks on graphs. In *Proc. IEEE International Conference on Image Processing (ICIP)*, pages 333–336, 2010.

[80] S K Narang and A Ortega. Lifting based wavelet transforms on graphs. In *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*, pages 441–444. Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference, International Organizing Committee, 2009.

[81] M Jansen, G P Nason, and B W Silverman. Multiscale methods for data on graphs and irregular multidimensional situations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(1):97–125, 2009.

[82] M Gavish, B Nadler, and R R Coifman. Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning. In *Proc. International Conference on Machine Learning, Haifa, Israel*, 2010.

[83] C Wang and S Mahadevan. Multiscale analysis of document corpora based on diffusion models. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1592–1597, 2009.

[84] N Beerenwinkel, L Pachter, and B Sturmfels. Epistasis and shapes of fitness landscapes. *arXiv preprint q-bio/0603034*, 2006.

[85] E D Weinberger. Fourier and taylor series on fitness landscapes. *Biological Cybernetics*, 65(5):321–330, 1991.

[86] D Rockmore, P Kostelec, W Hordijk, and P F Stadler. Fast fourier transform for fitness landscapes. *Applied and Computational Harmonic Analysis*, 12(1):57–76, 2002.

[87] M D Vose and A H Wright. The simple genetic algorithm and the walsh transform: Part i, theory. *Evolutionary Computation*, 6(3):253–273, 1998.

[88] J R Driscoll and D M Healy. Computing fourier transforms and convolutions on the 2-sphere. *Advances in applied mathematics*, 15(2):202–250, 1994.

[89] J-P Antoine and P Vandergheynst. Wavelets on the n-sphere and related manifolds. *Journal of mathematical physics*, 39:3987, 1998.

[90] L J Grady and J R Polimeni. *Discrete Calculus: Applied Analysis on Graphs for Computational Science*. Springer, 2010.

[91] V N Ekambaram, G C Fanti, B Ayazifar, and K Ramchandran. Critically-sampled perfect-reconstruction spline-wavelet filterbanks for graph signals. In *IEEE GlobalSip 2013, Austin, Tx*.

[92] V N Ekambaram, G C Fanti, B Ayazifar, and K Ramchandran. Multiresolution graph signal processing via circulant structures. In *IEEE DSP/SPE Worskhop 2013, Napa Valley, California*. IEEE, 2013.

[93] V N Ekambaram, G C Fanti, B Ayazifar, and K Ramchandran. Wavelet regularized graph semi-supervised learning. In *IEEE GlobalSip 2013, Austin, Tx*.

[94] V N Ekambaram, G C Fanti, B Ayazifar, and K Ramchandran. Circulant structures and graph signal processing,. In *Proc. Int. Conf. Image Process., Melbourne, Australia*, 2013.