Multiscale Transforms for Signals on Graphs: Methods and Applications

By

JEFFREY L. IRION

B.S. (University of California, San Diego) 2009

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Naoki Saito (Chair)

James Bremer

Albert Fannjiang

Committee in Charge

2015

Contents

Jeffrey L. Irion
December 2015
Applied Mathematics

Multiscale Transforms for Signals on Graphs: Methods and Applications

Abstract

Advances in data recording, data storage, and computing power have made possible both the collection and analysis of signals on a new domain: graphs. Here, a signal's structure is no longer confined to the equispaced, regularly connected domains of classical signal processing. Such freedom allows for much richer classes of signals to be considered and analyzed, but this increased versatility does not come without challenges. Nearly all of the theory and tools developed for classical signals cannot be generalized easily, if at all, to signals on graphs. Current methods must change and evolve, and new methods must be developed.

In this dissertation we present two multiscale transforms for signals on graphs that we have developed: the Hierarchical Graph Laplacian Eigen Transform (HGLET) and the Generalized Haar-Walsh Transform (GHWT), which can accurately be viewed as generalizations of the block DCT and Haar-Walsh wavelet packet transform, respectively. These transforms yield overcomplete dictionaries of basis vectors (and the corresponding expansion coefficients of an input signal) from which we can choose an orthonormal basis (and the corresponding nonredundant expansion coefficients) that is suited to the task at hand. For this purpose, we generalize the best basis search algorithm to the setting of our graph transforms. We prove some theoretical results for approximation and present experimental results in which we compare our transforms to previously developed transforms. Building upon these approximation results, we perform experiments in which we denoise signals on graphs using our transforms.

To further demonstrate the effectiveness and versatility of our transforms, we apply them to problems dealing with classical signals. First, we use the HGLET to simultaneously segment, denoise, and compress one-dimensional signals. We do so using an iterative algorithm in which we repeatedly partition the graph, analyze the signal, and find a best basis using the minimum

description length (MDL) principle as our cost functional. For our second application, we apply the GHWT to the problem of matrix data analysis. The advantage of the GHWT is that it can take into account the interrelationships between the rows and columns of the matrix, thereby enabling better analysis and characterization of the data. We present results for a sparse term-document matrix and a dense scrambled image matrix, and in both cases the tensor GHWT best basis reveals information about the data.

Acknowledgments

First and foremost, I'd like to thank my parents. Without their support, encouragement, and advice, this work would not have been possible. They've always been there for me and they've always been on my side, and I'm very lucky to have them as my parents.

I'd like to thank my grandmother "Yiyia" for allowing me to stay at her home while I finished writing my dissertation. I appreciate all the food she made for me (especially beef!) so that I could have more time to work, and she even did most of my grocery shopping for a couple months. I only hope that one day I can replace Doc Martin as her favorite doctor! And while my grandfather "Papou" passed away while I was still in undergrad, I remember the great value that he placed on education and I know he'd be proud of me.

I owe a great deal of thanks to my adviser, Professor Naoki Saito. He has been highly involved with my research, and he has invested a lot of time and energy in me. He has provided me with financial support, funding for travel, letters of recommendation, ideas and suggestions for my research, and general guidance and direction with my graduate studies. I'm very grateful to have had him as my adviser, and I very much appreciate all that he's done for me.

This research was partially supported by Dr. Saito's ONR grant N00014-12-1-0177 and NSF grand DMS-1418779, and was conducted with Government support under contract FA9550-11-C-0028 and awarded by the Department of Defense, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a.

Along with Professor Saito, I'd like to thank Professor James Bremer and Professor Albert Fannjiang for serving on my dissertation committee. I'm especially grateful to them for promptly reviewing my dissertation so that I can graduate in December 2015.

I'd like to thank the developers of several software packages for MATLAB. First, I'd like to thank the developers of WaveLab [**26**] for not only providing their software, but also for being leaders in the reproducible research movement. I'd like to thank the developers of SymmLab (which accompanies [**55**]), as their framework was very helpful in designing my own toolbox. Finally, I'd like to thank the developers of `export_fig` (`http://www.mathworks.com/matlabcentral/fileexchange/23629-export-fig`), which was used to output all of the MATLAB figures contained in this dissertation.

Thanks to Martin Prikryl for providing WinSCP (`http://winscp.net/`) free of charge. Being able to synchronize my work on multiple computers has made my life so much easier. Without a doubt, WinSCP has been one of the top three pieces of software that I have used for my graduate work (the other two being MATLAB and LaTeX).

I'm grateful to those researchers whose data I analyzed: Julie Coombs and her collaborators for the dendritic tree data; David Gleich for the Minnesota road network; the city of Toronto for the Toronto road network data; the developers of WaveLab for the "Msignal" and "Piece-Regular" signals; David Donoho, Iain Johnstone, and Warren Sarle for the clean and noisy "Blocks" signals; and Jeffrey Solka and his collaborators [**54**], Mauro Maggioni, and Matan Gavish for the Science News dataset.

Preliminary versions of parts of this dissertation (specifically, portions of Chapters 1, 3, 4, 5, 6, and 9) were published as conference proceeding papers [**37**, **38**, **39**].

"There's something to be said about purposely chasing something difficult."

Then there's a lot to be said about this Ph.D., because it's the hardest thing I've ever done! I'd like to thank all of the friends and family who have encouraged me along the way. And I'd like to thank all of those who have motivated and inspired me to believe in myself and to persevere. To have a dream and go after it with everything I've got. To work hard every day. And to keep moving forward, no matter what.

CHAPTER 1

Introduction

In recent years, the advent of new sensors, measurement technologies, and social network infrastructure has provided huge opportunities to visualize complicated interconnected network structures, record data of interest at various locations in such networks, analyze such data, and make inferences and diagnostics. We can easily observe such network-based problems in truly diverse fields: biology and medicine (e.g., voltages at dendrites connected to neurons, blood flow rates in a network of blood vessels); computer science (e.g., Internet traffic, email correspondences among user accounts); electrical engineering (e.g., sensor networks); hydrology and geology (e.g., river flow measurements in a ramified river network); and civil engineering (e.g., traffic flow on a road network), to name just a few. Consequently, there is an explosion of interest and demand to analyze data sampled on such irregular grids, graphs, and networks, as evidenced by many recent special issues of journals.

What about mathematical and computational tools for analyzing such datasets? Traditional harmonic analysis tools such as Fourier and wavelet transforms have been the 'crown jewels' for analyzing regularly-sampled data. They have found widespread use in a variety of applications, some of the most common being data compression, image analysis, and statistical signal processing. However, these conventional harmonic analysis tools originally developed for functions on simple Euclidean domains (e.g., a rectangle) or signals sampled on regular lattices cannot directly handle datasets recorded on general graphs and networks. Hence, the community of applied and computational harmonic analysts has recognized the importance of transferring these tools to the graph setting, resulting in many efforts to extend classical wavelets to the ever-expanding realm of data on graphs [**5**, **10**, **11**, **32**, **34**, **41**, **43**, **49**, **51**, **52**, **62**, **63**, **74**, **76**, **77**, **78**, **86**].

A fundamental difficulty in extending wavelets to the graph setting is that we lack a true notion of frequency. Indeed, much of classical signal processing relies on our ability to view a signal through two complementary lenses: time and frequency. Without a notion of frequency, it is quite nontrivial

to develop and apply the Littlewood-Paley theory (i.e., the dyadic partitioning of the frequency domain), which is the theoretical foundation of classical wavelets. Therefore, a common strategy has been to develop wavelet-*like* transforms rather than trying to directly transfer classical wavelets to the graph setting. As our contributions, we present two novel multiscale transforms for signals on graphs, along with several best basis search algorithms. We explore the theoretical properties of the transforms and the best basis algorithms, and we demonstrate their merit in approximation and denoising experiments. Moreover, we showcase the versatility of our transforms by applying them to problems involving classical 1-D and 2-D signals.

The organization of this dissertation is as follows. In Chapter 2 we review pertinent background material, including wavelets and wavelet packets (§2.1), graph theory (§2.2), and numerous transforms for signals on graphs (§2.3). In Chapter 3 we discuss recursive graph partitioning, which is a common strategy used by researchers to develop graph transforms and a precursor to our constructions. We present our Hierarchical Graph Laplacian Eigen Transform (HGLET) and Generalized Haar-Walsh Transform (GHWT) in Chapters 4 and 5, respectively. These transforms yield overcomplete dictionaries of orthonormal bases from which we can select a particular orthonormal basis tailored for a task at hand. For that purpose, we generalize the classical best basis algorithm to our graph transforms in Chapter 6. In Chapter 7 we present theoretical and experimental results for approximation achieved using the best basis algorithm in conjunction with our transforms. Building off of their success for approximation, in Chapter 8 we present some results for denoising signals on graphs. Having demonstrated the effectiveness of our transforms for analyzing signals on graphs, we then apply them to classical problems involving 1-D and 2-D signals. In Chapter 9 we present a method for simultaneously segmenting, denoising, and compressing 1-D signals using the HGLET variations and the best basis algorithm with the minimum description length criterion (MDL, see §9.2). In Chapter 10 we apply the GHWT to the task of matrix analysis and demonstrate the ability of the best basis algorithm to accurately capture the structure of the matrix data. We conclude with a summary of our transforms and results.

Throughout this dissertation, we make an effort to be clear and consistent with our notation. Between wavelets, graph theory, previously developed graph transforms, and our own HGLET and

GHWT, there certainly is a lot of notation! Here is what we typically use in this dissertation and its meaning.

Notation	Usual Meaning
N	the number of nodes in a graph or the length of a vector
M	the number of edges in a graph
j	scale/resolution index
k	location/subgraph index
l	an index for vectors/functions, often corresponding (at least somewhat) to frequency
n	an index $n \in [1, N]$; often used for vector entries (e.g., $\boldsymbol{f}(n)$)
i	a generic index variable
i	the imaginary number $\mathrm{i} = \sqrt{-1}$
$V = V(G)$	the set of vertices of a graph
$V_k^j = V(G_k^j)$	the set of vertices of a subgraph
V_j	a space in a classical multiresolution approximation (see §2.1, page 15)
$\mathcal{V}_j$	a space in the multiresolution approximation of Sharon and Shkolnisky [**74**] (see §2.3.2, page 46)
$\phi_{j,k}(t)$	a scaling function
$\psi_{j,k}(t)$	a wavelet function
$\boldsymbol{\phi}_l$	a Laplacian eigenvector
$\boldsymbol{\phi}_{k,l}^j$	an HGLET basis vector
$\boldsymbol{\psi}_{k,l}^j$	a GHWT basis vector

TABLE 1.1. The notation that we strive to keep consistent throughout this dissertation.

To provide some examples:

- For wavelets, $\psi_{j,k}(t)$ is a continuous wavelet function at scale j and location k. Similarly, $w_{j,k}^l(t)$ is a continuous wavelet packet function at scale j and location k.
- For a graph $G = G(V, E)$, we set $N := |V(G)|$, and thus a signal on the graph is a vector $\boldsymbol{f} \in \mathbb{R}^N$. The Laplacian eigenvectors are $\{\boldsymbol{\phi}_l\}_{l \in [0, N-1]}$.
- For our transforms, we use G_k^j to denote the kth subgraph of G on level j. $\boldsymbol{\phi}_{k,l}^j$ and $\boldsymbol{\psi}_{k,l}^j$ are HGLET and GHWT basis vectors, respectively, which correspond to subgraph G_k^j.

Accompanying this dissertation is the Multiscale Transforms for Signals on Graphs (MTSG) toolbox for MATLAB, available from `https://github.com/JeffLIrion/MTSG_Toolbox`. The toolbox includes scripts (see `DissertationFigures.m`) for generating many of the figures and tables contained herein, which we list below.

1.1. List of Reproducible Figures and Tables

Figures

Tables

CHAPTER 2

Background Material

2.1. Wavelets and Wavelet Packets

The Fourier transform is the classical tool in harmonic analysis. It maps a function $f(t)$ on the time domain to a function $\hat{f}(\omega)$ on the frequency domain; i.e., $\mathcal{F} : f \to \hat{f}$. The Fourier transform of a function $f(t) \in L^1(\mathbb{R})$ is given by [**80**, Ch. 5][1]

$$\mathcal{F}f(\omega) := \hat{f}(\omega) := \int_{-\infty}^{\infty} f(t)\mathrm{e}^{-2\pi\mathrm{i}\omega t}\,\mathrm{d}t.$$

If $\hat{f}(\omega)$ also belongs to $L^1(\mathbb{R})$, then we can recover $f(t)$ via the inverse Fourier transform [**80**, Ch. 5]:

$$f(t) = \int_{-\infty}^{\infty} \hat{f}(\omega)\mathrm{e}^{2\pi\mathrm{i}\omega t}\,\mathrm{d}\omega.$$

However, while the Fourier transform works well for analyzing functions which are global and smooth, it does not work well for functions that are localized in time [**42**, §3.1]. A number of mathematical tools have been developed to circumvent this shortcoming, including wavelets. Indeed, the wavelet transform has proven very useful in harmonic analysis due to its ability to handle signals that are localized in both time and frequency. A *wavelet*, also known as a mother wavelet, is a function $\psi \in L^2(\mathbb{R})$ that is centered at $t = 0$ with $\|\psi\|_2 = 1$ and

$$\int_{-\infty}^{\infty} \psi(t)\,\mathrm{d}t = \hat{\psi}(0) = 0 \tag{2.1}$$

[**45**, §4.3].

[1] By using a density argument, we are also able to define the Fourier transform for all functions $f \in L^2(\mathbb{R})$ [**45**, §2.2.2].

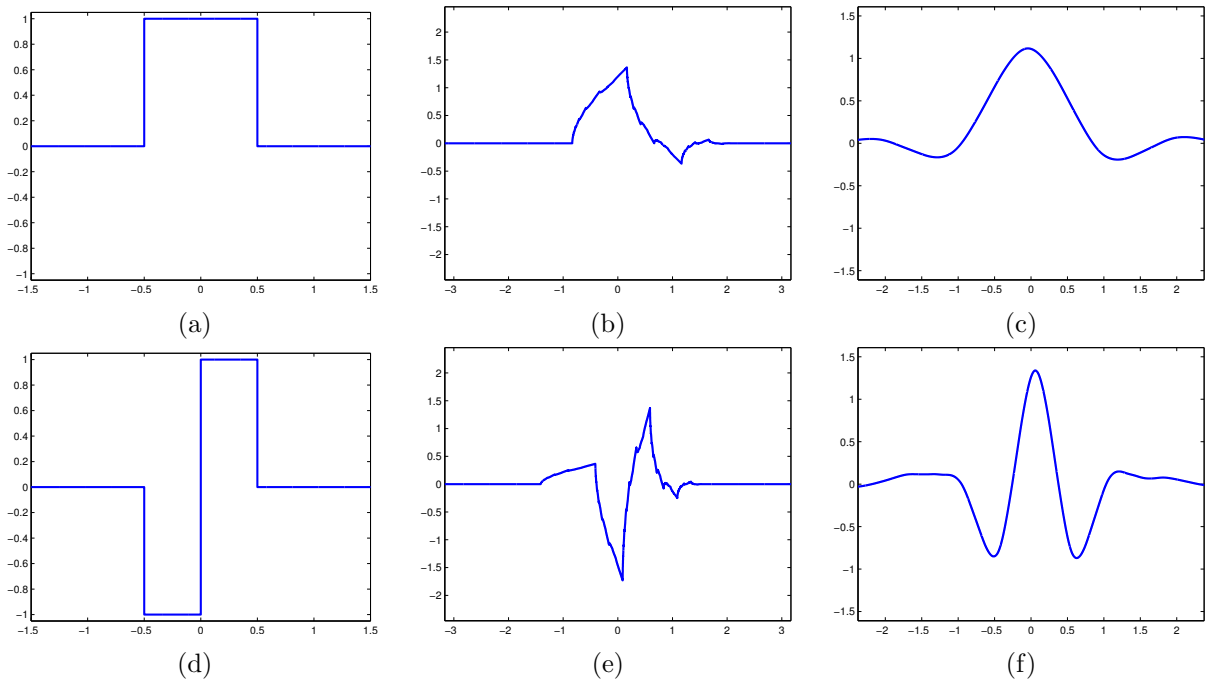


FIGURE 2.1. Three different scaling functions: Haar (a), Daubechies-4 (b), and Coiflet-4 (c). The corresponding wavelet functions: Haar (d), Daubechies-4 (e), and Coiflet-4 (f).

The wavelet function is accompanied by a *scaling function* ϕ, also known as a father wavelet, which satisfies $\|\phi\|_2 = 1$ [**45**, §4.3]; examples of scaling and wavelet functions can be seen in Figure 2.1. For all times/locations $u \in \mathbb{R}$ and scales $s \in \mathbb{R}^+$ (i.e., $s > 0$), we generate a family of translated and dilated versions of the wavelet function, $\{T_u D_s \psi\}_{u\in\mathbb{R},\ s\in\mathbb{R}^+}$, and scaling function, $\{T_u D_s \phi\}_{u\in\mathbb{R},\ s\in\mathbb{R}^+}$. However, to reduce redundancy in the forthcoming reconstruction, we typically restrict to discrete scales $s = 2^j$ and locations $u = 2^j k$, where $j, k \in \mathbb{Z}$, and we define [**42**, Ch. 7]:

$$\phi_{j,k}(t) := T_{2^j k} D_{2^j} \phi(t) = \frac{1}{\sqrt{2^j}} \phi\left(\frac{t - 2^j k}{2^j}\right) \tag{2.2a}$$

$$\psi_{j,k}(t) := T_{2^j k} D_{2^j} \psi(t) = \frac{1}{\sqrt{2^j}} \psi\left(\frac{t - 2^j k}{2^j}\right). \tag{2.2b}$$

The continuous wavelet transform of a function $f \in L^2(\mathbb{R})$ at scale $s = 2^j$ and time $u = 2^j k$ is [**45**, §4.3]

$$Wf(j,k) := \langle f, \psi_{j,k} \rangle = \int_{-\infty}^{\infty} f(t) \overline{\frac{1}{\sqrt{2^j}} \psi\left(\frac{t - 2^j k}{2^j}\right)}\, \mathrm{d}t. \tag{2.3}$$

Suppose that the set $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ constitutes a *frame*, meaning that there exist $B \geq A > 0$ such that

$$A\|f\|_2^2 \leq \sum_{j,k} |\langle f, \psi_{j,k}\rangle|^2 \leq B\|f\|_2^2 \qquad \text{for all } f \in L^2(\mathbb{R}).$$

(If $A = B$ then we say that it is a *tight frame*.) Then we can reconstruct f in the L^2 sense as [**42**, §7.2]

$$f(t) = \sum_{k=-\infty}^{\infty} \langle f, \phi_{j_{\max},k}\rangle\, \tilde{\phi}_{j_{\max},k}(t) + \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{j_{\max}} \langle f, \psi_{j,k}\rangle\, \tilde{\psi}_{j,k}(t). \tag{2.4}$$

In the first term the scaling function recovers information at the coarse scales, $j_{\max} \leq j < \infty$, while in the second term the wavelet function recovers information about the signal at the finer scales, $-\infty < j \leq j_{\max}$. The $\tilde{\psi}_{j,k}$ and $\tilde{\phi}_{j,k}$ are known as the *dual wavelet* and *dual scaling functions*, respectively, and they satisfy the biorthogonality relations [**45**, §7.8]

$$\begin{aligned}
\left\langle \psi_{j,k}, \tilde{\psi}_{j',k'} \right\rangle &= \delta(j-j')\delta(k-k') && \text{for every } j,j',k,k' \in \mathbb{Z} \\
\left\langle \phi_{j,k}, \tilde{\phi}_{j,k'} \right\rangle &= \delta(k-k') && \text{for every } j,k,k' \in \mathbb{Z} \\
\left\langle \psi_{j,k}, \tilde{\phi}_{j,k'} \right\rangle &= 0 && \text{for every } j,k,k' \in \mathbb{Z} \\
\left\langle \tilde{\psi}_{j,k}, \phi_{j,k'} \right\rangle &= 0 && \text{for every } j,k,k' \in \mathbb{Z},
\end{aligned} \tag{2.5}$$

where δ is the Kronecker delta. Henceforth, we restrict to the case of *orthogonal wavelets*, which means that both the dual wavelet and dual scaling functions are equal to their original counterparts.

Two common considerations in the design of particular wavelets are their support and their number of vanishing moments; that is, an orthogonal wavelet $\psi(t)$ has p vanishing moments if

$$\int_{-\infty}^{\infty} t^m \psi(t)\, \mathrm{d}t = 0 \quad \text{for all } 0 \leq m < p.$$

By the Fix-Strang condition, an orthogonal wavelet ψ has p vanishing moments if and only if every polynomial of degree $0 \leq m < p$ can be expressed as a linear combination of scaling functions $\{\phi(t-k)\}_{k\in\mathbb{Z}}$ [**82**]. This makes wavelets well-suited for representing signals which are piecewise polynomial, or nearly so. Using the wavelets in Figure 2.1 as examples, the Haar wavelet has compact support of length 1 and has 1 vanishing moment, which means that it is orthogonal to constant functions. A Daubechies-N wavelet (N even) has compact support of length $N-1$ and

has $N/2$ vanishing moments, and the Daubechies-2 wavelet is simply the Haar wavelet. A Coiflet-N wavelet has $2N$ vanishing moments and compact support of length $6N-1$ [**45**, §7.2].

So far we have defined the wavelet transform as the integral (2.3) for continuous-time functions, but we now look to move towards a more efficient implementation and ultimately to discrete-time signals. We begin by defining the spaces

$$V_j := \overline{\text{span}(\{\phi_{j,k}\}_{k\in\mathbb{Z}})}$$
$$W_j := \overline{\text{span}(\{\psi_{j,k}\}_{k\in\mathbb{Z}})}.$$

For orthogonal wavelets, the following properties are satisfied [**45**, §7.1], [**19**, §5.1]:

$$W_j \perp W_{j'} \quad \text{for all } j \neq j' \tag{2.6a}$$

$$V_j \perp W_j \tag{2.6b}$$

$$f(t) \in V_j \Leftrightarrow f\left(\frac{t}{2}\right) \in V_{j+1} \tag{2.6c}$$

$$V_j \supset V_{j+1} \tag{2.6d}$$

$$\{0\} = \bigcap_{j=-\infty}^{\infty} V_j \tag{2.6e}$$

$$L^2(\mathbb{R}) = \overline{\bigcup_{j=-\infty}^{\infty} V_j} \tag{2.6f}$$

$$V_j = V_{j+1} \oplus W_{j+1}. \tag{2.6g}$$

Properties (2.6a) and (2.6b) follow from the biorthogonality relations (2.5), and properties (2.6c)-(2.6f) mean that the spaces $\{V_j\}_{j\in\mathbb{Z}}$ form a *multiresolution approximation*. Property (2.6g) enables us to write wavelet and scaling functions as a linear combination of the scaling functions at a finer scale:

$$\phi_{j,k}(t) = \sum_n \underbrace{h(n-2k)}_{=\langle \phi_{j,k}, \phi_{j-1,n}\rangle} \phi_{j-1,n}(t) \tag{2.7a}$$

$$\psi_{j,k}(t) = \sum_n \underbrace{g(n-2k)}_{=\langle \psi_{j,k}, \phi_{j-1,n}\rangle} \phi_{j-1,n}(t), \tag{2.7b}$$

where h and g are sequences of real numbers known as filters. These filters allow us to perform the wavelet transform without having to construct the wavelets on each level and compute the inner products. Specifically, by inserting the refinement relations (2.7a) and (2.7b) into (2.4) and replacing the dual functions (since we have assumed that the wavelets are orthogonal), we can reconstruct the signal as

$$f(t) = \sum_{k=-\infty}^{\infty} c_{j_{\max}}(k)\phi_{j_{\max},k}(t) + \sum_{k=-\infty}^{\infty}\sum_{j=-\infty}^{j_{\max}} d_j(k)\psi_{j,k}(t), \tag{2.8}$$

where

$$c_j(k) := \langle f, \phi_{j,k}\rangle = \sum_n h(n-2k)c_{j-1}(n) \tag{2.9a}$$

$$d_j(k) := \langle f, \psi_{j,k}\rangle = \sum_n g(n-2k)c_{j-1}(n). \tag{2.9b}$$

The c_j's and d_j's are known as scaling coefficients and wavelet coefficients, respectively.

Now we consider the case of a discrete signal $\boldsymbol{f} \in \mathbb{R}^N$, where $N = 2^{n_0}$ for some $n_0 \in \mathbb{Z}$. The refinement relations (2.7a) and (2.7b) still hold, except that the summations are now finite[2]. For simplicity of notation, we take $j = 0$ to be the finest level and $j = j_{\max}$ to be the coarsest, where $0 < j_{\max} \leq n_0$. We begin by setting the scaling coefficients on the finest level to be the values of the discrete signal, i.e., $c_0(k) := \boldsymbol{f}(k)$ for $k = 1, \ldots, N$. As with the continuous case, we use the refinement relations to generate the scaling and wavelet coefficients at the subsequent coarser levels. Keeping in mind that there are only a finite number of scales j and locations k, we generalize the reconstruction formula (2.8) to the discrete case as

$$\boldsymbol{f}(n) = \sum_{k=0}^{2^{n_0-j_{\max}}-1} \underbrace{c_{j_{\max}}(k)}_{:=\langle \boldsymbol{f}, \phi_{j_{\max},k}\rangle} \phi_{j_{\max},k}(n) + \sum_{j=1}^{j_{\max}}\sum_{k=0}^{2^{n_0-j}-1} \underbrace{d_j(k)}_{:=\langle \boldsymbol{f}, \psi_{j,k}\rangle} \psi_{j,k}(n), \tag{2.10}$$

with c_j and d_j computed as before in (2.9a) and (2.9b), with considerations made at the boundaries. Making use of filters and refinement relations, we can implement the discrete wavelet transform in $O(N)$ operations, which is fewer than the $O(N \log N)$ operations required by the Fast Fourier Transform [**15**].

[2]Considerations must be made at the boundary; see, e.g., [**45**, §7.5] for details.

At each level $j < j_{\max}$, the scaling coefficients are processed using Eqs. (2.9a) and (2.9b) to yield the scaling and wavelet coefficients on the next level; we illustrate this structure in the tree in Figure 2.2. Furthermore, from one scale to the next we dilate the scaling and wavelet functions by a factor of 2, as seen in (2.2a) and (2.2b). The result is that the frequency resolution of the functions are doubled, while their time resolution is cut in half. Figure 2.2 illustrates the relationship between the structure of the wavelet transform and the time-frequency resolution of the wavelet basis.

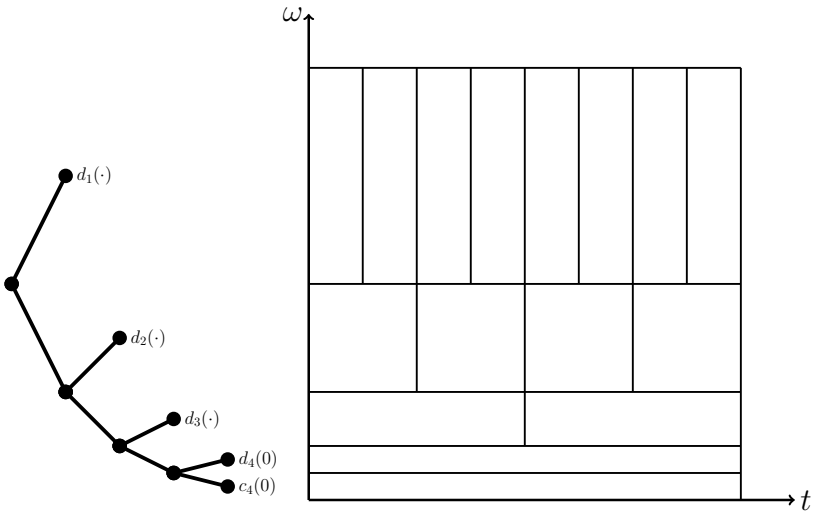


FIGURE 2.2. The tree on the left illustrates the structure of the wavelet transform for a discrete signal of length 16. The transform starts on the left with level $j = 0$, and at each level the low- and high-pass filters h and g, respectively, are applied to the scaling coefficients to yield scaling (lower child node) and wavelet (upper child node) coefficients on the next level. The figure on the right shows the time-frequency resolutions that correspond to the wavelet (and scaling) coefficients.

A major advantage of the wavelet transform is that it is able to capture features of the signal with varying localizations in time and frequency. That is, the time and frequency resolutions of the basis functions are not fixed, as is the case with the short-time Fourier transform [**45**, §4.2], [**19**, Ch. 1]. Indeed, from the logarithmic tiling of the time-frequency plane in Figure 2.2, we see that the wavelet transform can capture low-frequency global features, high-frequency local features, and everything in between. However, a drawback of wavelets is the inverse relationship between the frequency and the localization in time of the basis functions. As such, wavelets provide a poor representation for signals with high-frequency global components. This is due to the fact that

only the scaling coefficients are processed with the low- and high-pass filters, which results in the one-sided wavelet tree seen in Figure 2.2.

This shortcoming led to the development of *wavelet packets* by Coifman, Meyer, and Wickerhauser [**13**]. Here, we apply the low- and high-pass filters to the high-frequency wavelet coefficients in the same manner that we do with the low-frequency scaling coefficients. As in our discussion of wavelets, for the sake of simplicity we restrict our discussion to the case of orthogonal wavelet packets. Following the notation in [**61**, Ch. 9], we begin by setting $w^0_{j,k}(t) := \phi_{j,k}(t)$ and $w^1_{j,k}(t) := \psi_{j,k}(t)$. We generate wavelet packet functions via

$$w^{2l}_{j,k}(t) := \sum_n h(n-2k) w^l_{j-1,n}(t) \tag{2.11a}$$

$$w^{2l+1}_{j,k}(t) := \sum_n g(n-2k) w^l_{j-1,n}(t) \tag{2.11b}$$

and wavelet packet coefficients via

$$d^{2l}_j(k) := \left\langle f, w^l_{j,k} \right\rangle = \sum_n h(n-2k) d^l_{j-1}(n) \tag{2.12a}$$

$$d^{2l+1}_j(k) := \left\langle f, w^l_{j,k} \right\rangle = \sum_n g(n-2k) d^l_{j-1}(n). \tag{2.12b}$$

Note that the scaling and wavelet coefficients are a subset of the wavelet packet coefficients: $c_j(k) = d^0_j(k)$ and $d_j(k) = d^1_j(k)$. Figure 2.3 shows some examples of wavelet packet functions. In the simplest sense, these are more oscillatory versions of the wavelet functions seen in Figure 2.1.

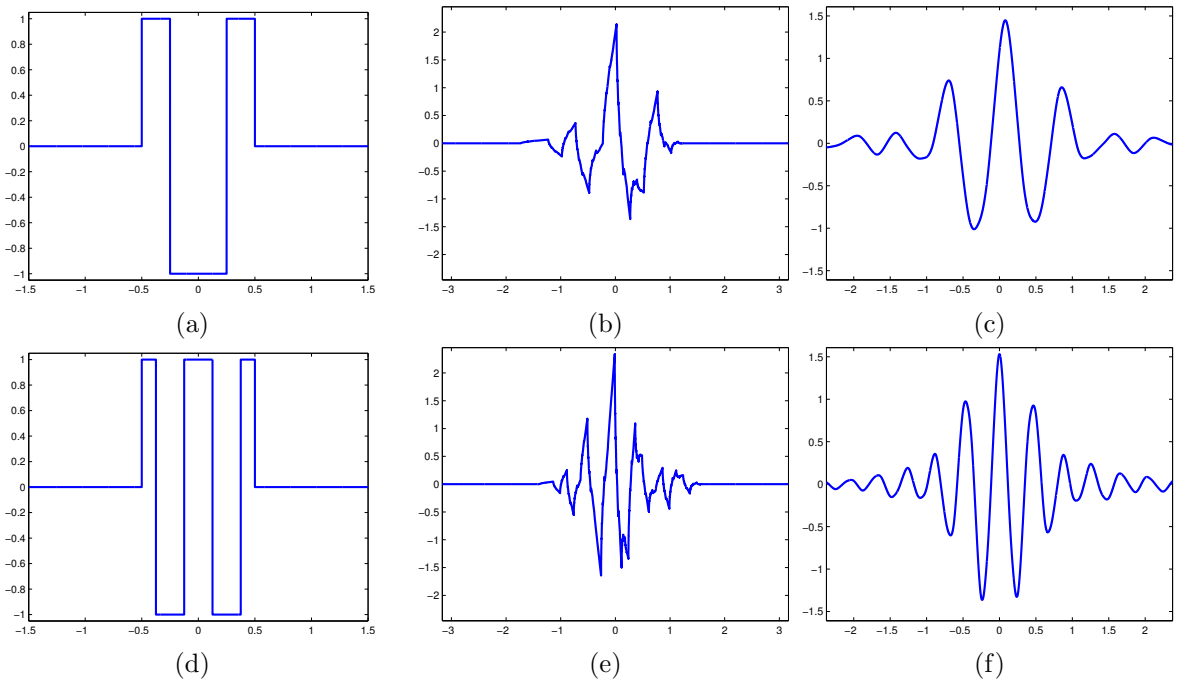


FIGURE 2.3. Three different wavelet packets $w_{0,0}^3$ for: Haar-Walsh (a), Daubechies-4 (b), and Coiflet-4 (c). Wavelet packets $w_{0,0}^6$ for: Haar-Walsh (d), Daubechies-4 (e), and Coiflet-4 (f).

As with wavelets, for a signal $\boldsymbol{f} \in \mathbb{R}^N$, where $N = 2^{n_0}$, we perform the wavelet packet transform using refinement relations with the filters h and g. Thus, Eqs. 2.12a and 2.12b still hold, with considerations being made at the boundaries. For a signal of length $N = 2^{n_0}$, the output of the wavelet packet transform is an $N \times (n_0 + 1)$ matrix of expansion coefficients. Whereas the wavelet transform generates N coefficients and its computational cost is $O(N)$, the computational cost of the wavelet packet transform is $O(N \log N)$ operations, since we generate N coefficients on levels $j = 1, 2, \ldots, n_0 = \log_2 N$ (the coefficients on level $j = 0$ are simply the original values of the discrete signal). In the case of $N = 8$, Table 2.1 displays the wavelet packet coefficients, Figure 2.4 displays the Haar-Walsh wavelet packet functions, and the tree in Figure 2.5a illustrates the structure of the full wavelet packet transform.

$j=0$	$d_0^0(0)$	$d_0^0(1)$	$d_0^0(2)$	$d_0^0(3)$	$d_0^0(4)$	$d_0^0(5)$	$d_0^0(6)$	$d_0^0(7)$
$j=1$	$d_1^0(0)$	$d_1^0(1)$	$d_1^0(2)$	$d_1^0(3)$	$d_1^1(0)$	$d_1^1(1)$	$d_1^1(2)$	$d_1^1(3)$
$j=2$	$d_2^0(0)$	$d_2^0(1)$	$d_2^1(0)$	$d_2^1(1)$	$d_2^2(0)$	$d_2^2(1)$	$d_2^3(0)$	$d_2^3(1)$
$j=3$	$d_3^0(0)$	$d_3^1(0)$	$d_3^2(0)$	$d_3^3(0)$	$d_3^4(0)$	$d_3^5(0)$	$d_3^6(0)$	$d_3^7(0)$

TABLE 2.1. The table of wavelet packet coefficients for a signal of length 8. Scaling coefficients ($l=0$) are in black, wavelet coefficients ($l=1$) are in red, and wavelet packet coefficients ($l \geq 2$) are in blue.

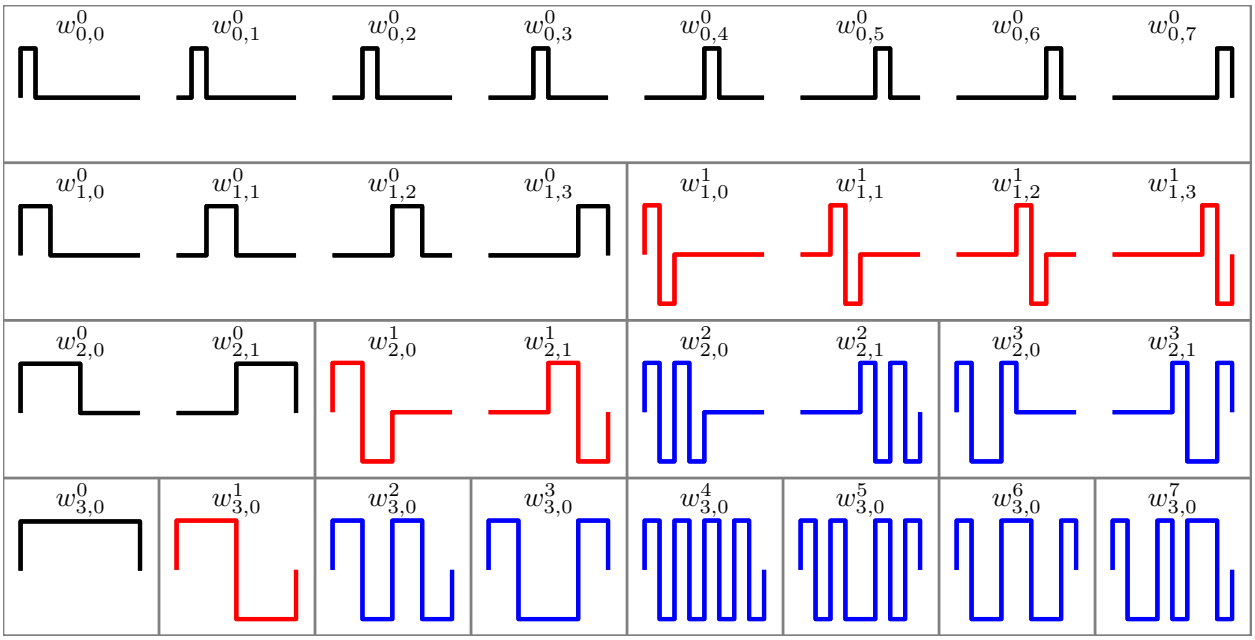


FIGURE 2.4. The Haar wavelet packet functions for a signal of length 8. As in Table 2.1, scaling functions ($l=0$) are in black, wavelet functions ($l=1$) are in red, and wavelet packet functions ($l \geq 2$) are in blue. The functions on the bottom level of this chart are the so-called Walsh functions.

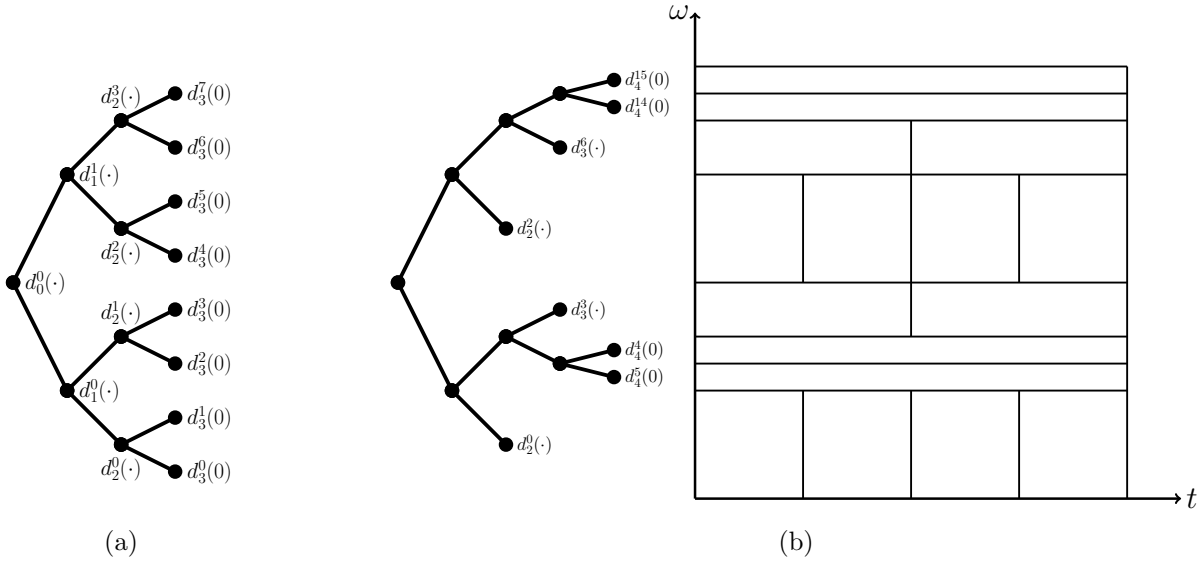


FIGURE 2.5. (a) The structure of the full wavelet packet transform for a discrete signal of length 8. (b) The wavelet packet tree and corresponding tiling of the time-frequency plane for a particular wavelet packet basis for a discrete signal of length 16.

Before we continue our general discussion of wavelet packets, we will take this opportunity to say a few words about the Haar-Walsh wavelet packets. These functions are piecewise-constant, as can be seen in Figure 2.4 for the case of $N = 8$. In particular, the functions on level $j = \log_2 N$ (i.e., the bottom level in the figure) are known as the *Walsh functions*, and they only assume the values $\{-\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}\}$. These Walsh functions correspond to the rescaled columns of the $N \times N$ *Hadamard matrix* H_N, which is an $N \times N$ matrix that assumes only the values $\{-1, 1\}$ and satisfies $H_N H_N^{\mathsf{T}} = H_N^{\mathsf{T}} H_N = N I_N$, where I_N is the identity matrix [**1**, Ch. 1]. For $N = 8$, the *natural-ordered Hadamard matrix* H_8 and the *Paley-ordered Hadamard matrix* H_8^{p} are

$$H_8 := \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}, \qquad H_8^{\mathrm{p}} := \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

Note that these matrices differ only in the order of their columns, and also that the order of the columns in H_8^{p} corresponds to the Walsh functions in Figure 2.4. The *Walsh-Hadamard transform* of a signal $\boldsymbol{f} \in \mathbb{R}^N$ is given by $\frac{1}{\sqrt{N}} H_N \boldsymbol{f}$, and this can be computed in $N \log_2 N$ addition/subtraction operations [**1**, §2.2].

Returning to our general wavelet packet discussion, observe that the wavelet tree in Figure 2.2 is one-sided, while the wavelet packet tree in Figure 2.5a is full and balanced, containing numerous combinations of functions which comprise an orthonormal basis. Thus, the wavelet packet transform is flexible, allowing us to choose from this overcomplete tree a particular orthonormal basis whose tiling of the time-frequency plane is well-suited to the signal(s) being considered. And we certainly have plenty of choices: for a signal of length $N = 2^{j_{\max}} > 2$, we have more than $2^{N/2} = 2^{2^{j_{\max}-1}}$ choosable orthonormal bases [**45**, §8.1]. Indeed, the orthonormal basis illustrated in Figure 2.5b is one of the 677 possible orthonormal wavelet packet bases for a signal of length 16; this is the same as the number of binary trees of height $\leq \log_2(N) + 1 = 5$ [**30**, **31**]. (A *binary tree* is a rooted tree where each vertex has at most two children [**89**, §2.3].) The number B_n of binary trees of height $\leq n$ is given by the recurrence relation

$$B_n = \begin{cases} 1 & n = 1 \\ B_{n-1}^2 + 1 & n \geq 2. \end{cases}$$

Given the overwhelming number of choosable bases, the obvious question is: how should a basis be chosen? To this end, Coifman and Wickerhauser developed the *best basis algorithm* [**14**], which searches through the overcomplete dictionary[3] of expansion coefficients and chooses the basis that minimizes a user-specified cost functional $\mathcal{M}$. They require that this cost functional be *additive*, meaning that $\mathcal{M}(0) = 0$ and $\mathcal{M}(\{x_i\}) = \sum_i \mathcal{M}(x_i)$. The input to their algorithm is a full table of wavelet packet coefficients for a signal of dyadic length; an example for a signal of length 8 is shown in Table 2.1. We search among this overcomplete set of expansion coefficients, comparing costs of parent blocks and their children/descendent blocks. The output is the particular set of orthonormal coefficients $\{b_i\}_{i=0}^{N-1}$ that minimizes the cost functional (see Proposition 2.1.1); accordingly, the basis to which these coefficients correspond is called the *best basis*. Their algorithm proceeds as follows.

[3] A *dictionary* is a set of elementary functions, or *atoms*, used to analyze a signal [**60**]; we also use the term dictionary to refer to the collection of corresponding expansion coefficients.

Algorithm 1 Best Basis Algorithm [**14**]

Initialize the level $j = j_{\max}$ basis to be the best basis. That is, initialize $\{b_i\}_{i=0}^{N-1} := \{d_{j_{\max}}^l(0)\}_{l=0}^{N-1}$.
for $j = j_{\max} - 1, \ldots, 0$ **do**
  for $l = 0, \ldots, 2^j - 1$ **do**
    if $\mathcal{M}\left(\{b_i\}_{i=l2^{j_{\max}-j}}^{(l+1)2^{j_{\max}-j}-1}\right) \geq \mathcal{M}\left(\{d_j^l(k)\}_{k=0}^{2^{j_{\max}-j}-1}\right)$ **then**
      Set $\{b_i\}_{i=l2^{j_{\max}-j}}^{(l+1)2^{j_{\max}-j}-1} := \{d_j^k(l)\}_{k=0}^{2^{j_{\max}-j}-1}$.
    end if
  end for
end for

The algorithm can be described simply in terms of Table 2.1: start at the bottom and proceed upwards, comparing blocks of coefficients to those blocks of the current best basis that lie beneath and revising the best basis as needed. As justification for their best basis algorithm, Coifman and Wickerhauser offer the following result.

PROPOSITION 2.1.1. *[**14**] Given a signal $\boldsymbol{f}$ of dyadic length, an additive cost functional $\mathcal{M}$, and an overcomplete set of wavelet packet expansion coefficients $\{d_j^l(k)\}_{j,k,l}$, the set $\{b_i\}_{i=0}^{N-1}$ returned by Algorithm 1 is the set of orthonormal expansion coefficients from the wavelet packet dictionary that minimizes $\mathcal{M}$.*

Saito and Coifman extended this algorithm to the setting of classification, developing the *local discriminant basis algorithm* for classification and regression [**66**,**67**], which was later refined in [**68**]. The search proceeds in the same manner, but rather than minimizing an additive cost functional $\mathcal{M}$ they maximize an additive discriminant measure $\mathcal{D}$ over the wavelet packet coefficients for a collection of signals. The resulting local discriminant basis coefficients, or a subset thereof, are then used as features for a classification algorithm, such as Linear Discriminant Analysis (LDA) or Classification and Regression Tree (CART). They have used this same search technique to find the least statistically-dependent basis [**65**].

It is worth mentioning that other basis selection algorithms have been proposed, most notably the matching pursuits [**46**] and basis pursuit [**6**] algorithms. In addition, a number of search

algorithms pertaining specifically to Haar-Walsh wavelet packets have been developed [**2**, **35**, **87**]. We mention this for the sake of completeness, but we do not address them further as they are not amenable to the multiscale transforms that we have developed.

2.2. Graph Theory

Having discussed classical wavelets and wavelet packets, we now turn our attention to a different topic altogether: graphs. In this section we cover some fundamental graph theory, and in doing so introduce the graph notation that will be used throughout this dissertation.

Let $G = (V, E)$ be an undirected connected graph. $V = V(G) = \{v_1, v_2, \ldots, v_N\}$ denotes the set of vertices (or nodes) of the graph, where $N := |V(G)|$. For simplicity, we typically associate each vertex with its index and write i in place of v_i. $E = E(G) = \{e_1, e_2, \ldots, e_M\}$ is the set of edges, where each e_k connects two vertices i and j, and $M := |E(G)|$. In this dissertation we consider only finite graphs (i.e., $M, N < \infty$). Moreover, we restrict to the case of simple graphs; that is, graphs without loops (an edge connecting a vertex to itself) and multiple edges (more than one edge connecting a pair of vertices i and j). We use $\boldsymbol{f} \in \mathbb{R}^N$ to denote a signal on G, and we define $\mathbf{1} := (1, \ldots, 1)^\mathsf{T} \in \mathbb{R}^N$. For a subset of vertices $X \subseteq V(G)$, we define $\mathbf{1}_X \in \mathbb{R}^N$ to be the vector that is one at all positions corresponding to nodes in X and zero elsewhere. We also define $\boldsymbol{f}|_X \in \mathbb{R}^{|X|}$ to be the restriction of $\boldsymbol{f}$ to the vertices in X.

We now discuss several matrices associated with graphs. The information in both V and E is captured by the *edge weight matrix* $W(G) \in \mathbb{R}^{N \times N}$, where $W_{ij} \geq 0$ is the edge weight between nodes i and j. In an unweighted graph, this is restricted to be either 0 or 1, depending on whether nodes i and j are connected, and we may refer to $W(G)$ as an *adjacency matrix*. In a weighted graph, W_{ij} indicates the affinity between i and j. In either case, since G is undirected, $W(G)$ is a symmetric matrix. We then define the *degree matrix* $D(G)$ as the diagonal matrix with entries $d_i = \sum_j W_{ij}$. With this in place, we are now able to define the *(unnormalized) Laplacian* matrix, *random-walk normalized Laplacian* matrix, and *symmetric normalized Laplacian* matrix, respectively, as

$$
\begin{aligned}
L(G) &:= D(G) - W(G) \\
L_{\mathrm{rw}}(G) &:= D(G)^{-1} L(G) \\
L_{\mathrm{sym}}(G) &:= D(G)^{-1/2} L(G) D(G)^{-1/2}.
\end{aligned}
$$

We use $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{N-1}$ to denote the sorted Laplacian eigenvalues and $\boldsymbol{\phi}_0, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{N-1}$ to denote their corresponding eigenvectors, where the specific Laplacian matrix to which they refer will be clear from either context or superscripts.

These matrices have been studied extensively, and we now highlight three key properties (further information can be found in [**7**,**88**]). First, for all three matrices the smallest eigenvalue is zero and for a connected graph all the other eigenvalues are strictly positive. Furthermore, for both L and L_{rw} the eigenvector associated to eigenvalue zero is the normalized constant vector: $\boldsymbol{\phi}_0 = \mathbf{1}/\sqrt{N}$ and $\boldsymbol{\phi}_0^{\text{rw}} = \mathbf{1}/\sqrt{\sum_{i=1}^{N} d_i}$. Second, both L and L_{sym} are symmetric matrices and therefore their eigenvectors form orthonormal bases for $\mathbb{R}^N$. Indeed, their associated quadratic forms,

$$\boldsymbol{f}^{\mathsf{T}} L \boldsymbol{f} = \frac{1}{2} \sum_{i,j} W_{ij} \Big(\boldsymbol{f}(i) - \boldsymbol{f}(j) \Big)^2$$

$$\boldsymbol{f}^{\mathsf{T}} L_{\text{sym}} \boldsymbol{f} = \frac{1}{2} \sum_{i,j} W_{ij} \left(\frac{\boldsymbol{f}(i)}{\sqrt{d_i}} - \frac{\boldsymbol{f}(j)}{\sqrt{d_j}} \right)^2,$$

allow the first property to be easily observed. Third, L_{rw} and L_{sym} have the same eigenvalues, and their eigenvectors are related in the following way:

$$\boldsymbol{\phi}_l^{\text{rw}} = D(G)^{-1/2} \boldsymbol{\phi}_l^{\text{sym}} \qquad l = 0, 1, \ldots, N-1. \tag{2.13}$$

From this, it is easily seen that the eigenvectors of L_{rw} are orthonormal with respect to the weighted inner product $\langle , \rangle_{D(G)}$; that is, $(\boldsymbol{\phi}_{l_1}^{\text{rw}})^* D(G) \boldsymbol{\phi}_{l_2}^{\text{rw}} = \delta_{l_1, l_2}$. This also explains why the constant vectors $\boldsymbol{\phi}_0$ and $\boldsymbol{\phi}_0^{\text{rw}}$ are normalized by different constants. We will later use (2.13) to expand a signal $\boldsymbol{f} \in \mathbb{R}^N$ in terms of the eigenvectors of L_{rw} using only matrix multiplication by $D(G)^{-1/2}$ and the eigenvectors of L_{sym}; i.e., without solving the linear system $L_{\text{rw}} \boldsymbol{c} = \boldsymbol{f}$. It is also worth mentioning that $\lambda_{N-1}^{\text{rw}} = \lambda_{N-1}^{\text{sym}} \leq 2$.

In addition to serving as bases for signals on a graph, Laplacian eigenvectors can also be used for graph partitioning. For a connected graph G, Fiedler showed that an eigenvector corresponding to the first nonzero eigenvalue of the unnormalized Laplacian (i.e., $\boldsymbol{\phi}_1$) partitions the vertices into

two sets,

$$V_1 = \left\{ i \;\middle|\; \boldsymbol{\phi}_1(i) \geq 0 \right\}$$
$$V_2 = V \setminus V_1,$$

such that the subgraphs induced on V_1 and V_2 by G are both connected graphs [**29**]. Thus, the *Fiedler vector*, as it has come to be known, provides a simple means of bipartitioning. This result also holds when using $\boldsymbol{\phi}_1^{\text{rw}}$ (which is equivalent to using $\boldsymbol{\phi}_1^{\text{sym}}$, since (2.13) reveals that the eigenvector entries will have the same signs). Justification of this approach comes from the fact that it yields an approximate minimizer of the bipartitioning criterion called the *RatioCut* (or the *Normalized Cut*) when L (or L_{rw}, respectively) is used [**75**, **88**]. This result can be seen as a corollary of the Discrete Nodal Domain Theorem [**4**, **20**], and by utilizing more of the Laplacian eigenvectors we can partition the graph into more subgraphs.

We now cover two examples which motivate the use of Laplacian eigenvectors for analyzing signals on graphs. The first such example is P_N, the unweighted path graph of length N, which is illustrated in Figure 2.6. Its unnormalized Laplacian is

$$\underbrace{\begin{bmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 1 \end{bmatrix}}_{L(G)} = \underbrace{\begin{bmatrix} 1 & & & & & \\ & 2 & & & & \\ & & \ddots & & & \\ & & & 2 & & \\ & & & & 1 \end{bmatrix}}_{D(G)} - \underbrace{\begin{bmatrix} 0 & 1 & & & & \\ 1 & 0 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & 0 & 1 & \\ & & & 1 & 0 \end{bmatrix}}_{W(G)}. \tag{2.14}$$

The Laplacian eigenvalues and eigenvectors are given by[4]

$$\left.\begin{aligned} \lambda_l &= 4\sin^2\left(\frac{\pi l}{2N}\right) \\ \boldsymbol{\phi}_l(n) &= \cos\left(\frac{\pi l\left(n-\frac{1}{2}\right)}{N}\right) \end{aligned}\right\} \quad \text{where } l = 0, \ldots, N-1 \quad \text{and } n = 1, \ldots, N. \tag{2.15}$$

[4]To minimize notation, we do not normalize the eigenvectors in (2.15), (2.17), and (2.19).

FIGURE 2.6. A path graph P_n provides a simple yet important example.

As noted in [**50**, **72**], these eigenvectors are exactly the DCT-II basis vectors. Meanwhile, the symmetric normalized Laplacian is given by

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2} = \begin{bmatrix} 1 & -1/\sqrt{2} & & & & & \\ -1/\sqrt{2} & 1 & -1/2 & & & & \\ & -1/2 & \ddots & \ddots & & & \\ & & \ddots & \ddots & -1/2 & & \\ & & & -1/2 & 1 & -1/\sqrt{2} & \\ & & & & -1/\sqrt{2} & 1 & \end{bmatrix}. \tag{2.16}$$

Its eigenvalues and eigenvectors are [**7**, **90**]

$$\left.\begin{aligned} \lambda_l &= 2\sin^2\left(\frac{\pi l}{2(N-1)}\right) \\ \boldsymbol{\phi}_l(n) &= \cos\left(\frac{\pi l(n-1)}{N}\right) \end{aligned}\right\} \quad \text{where } l = 0, \ldots, N-1 \quad \text{and } n = 1, \ldots, N. \tag{2.17}$$

As we stated in [**39**], these eigenvectors are the DCT-I basis vectors. And of course, these eigenvectors correspond to those of L_{rw} by (2.13). (In fact, the eigenvectors of L_{rw} are the eigenvectors of the DCT-I second difference matrix before it is rescaled to make it symmetric, thus making its eigenvectors orthogonal; see [**81**, **90**] for more details.) Note that the eigenvalue and eigenvector formulas for L in (2.15) and those for L_{sym} in (2.17) are very similar. This is because L and L_{sym} are both second-difference matrices with Neumann boundary conditions, differing only in how their boundary conditions are discretized (and also by a factor of 2).

The second example to be considered is the unweighted cycle C_N, which is seen in Figure 2.7. Its Laplacian matrix is

$$\underbrace{\begin{bmatrix} 2 & -1 & & & -1 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ -1 & & & -1 & 2 \end{bmatrix}}_{L(G)} = \underbrace{\begin{bmatrix} 2 & & & & \\ & 2 & & & \\ & & \ddots & & \\ & & & 2 & \\ & & & & 2 \end{bmatrix}}_{D(G)} - \underbrace{\begin{bmatrix} 0 & 1 & & & 1 \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ 1 & & & 1 & 0 \end{bmatrix}}_{W(G)}. \tag{2.18}$$

Its eigenvalues and eigenvectors are[5] [**7**]

$$\left.\begin{aligned} \lambda_l &= 4\sin^2\left(\frac{\pi l}{N}\right) \\ \boldsymbol{\phi}_l(n) &= \mathrm{e}^{2\pi \mathrm{i}(n-1)l/N} \end{aligned}\right\} \quad \text{where } l = 0, \ldots, N-1 \quad \text{and } n = 1, \ldots, N. \tag{2.19}$$

These are precisely the basis vectors of the Discrete Fourier Transform (DFT). Indeed, it is easily seen that L in (2.18) is a second difference matrix with periodic boundary conditions, and hence its eigenvectors are the complex exponentials. Furthermore, since the degree matrix is simply $D = 2I$, we have that $L_{\text{sym}} = L_{\text{rw}} = 0.5L$. Therefore, the eigenvectors are the same and their eigenvalues are half those of L. However, a word of caution must be issued regarding the connection between the Laplacian eigenvectors of an unweighted cycle and the DFT basis vectors. As the Laplacian matrices L and L_{sym} are symmetric, they emit a set of real-valued orthonormal eigenvectors; by (2.13), we can obtain a set of real-valued eigenvectors for L_{rw} as well. (All undirected graphs emit a set of real-valued orthonormal eigenvectors for each Laplacian, not just cycles; for L_{rw}, these eigenvectors are orthonormal with respect to the degree-weighted inner product.) For example, we can construct a set of real-valued orthonormal eigenvectors for C_N by taking the *unique* sine and cosine components of the eigenvectors $\boldsymbol{\phi}_l$ in (2.19) and normalizing. Thus, the complex exponentials in (2.19) are merely one choice of eigenvectors. And if our set of eigenvectors does not coincide with the complex exponentials, then generalizations of Fourier theory become problematic, as we will see in §2.3.1.

[5]Note that in (2.19) the eigenvalues are not in nondecreasing order. In fact, for $l \in [1, N-1]$ we have that $\lambda_l = \lambda_{N-l}$.

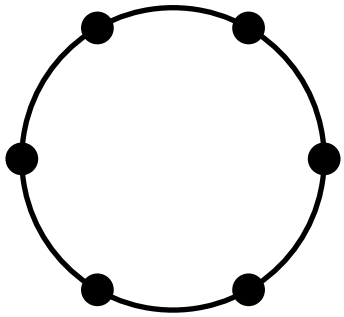


FIGURE 2.7. An unweighted cycle of length 6.

These two examples are important because they serve as a bridge between classical signal processing and signal processing on graphs. The connections we have pointed out between Laplacian eigenvectors and the DCT and DFT help to motivate the use of Laplacian eigenvectors for analyzing signals on graphs. Furthermore, it is desirable that any generalizations of classical concepts and techniques (e.g., the Fourier transform, frequency, dilation, translation, etc.) to the graph setting should agree with their classical counterparts on these simple graphs. We will use these examples later to explain and evaluate various tools and techniques for analyzing signals on graphs.

2.3. A Review of Graph-Based Transforms

We now review previous work that has been done to develop wavelet-like transforms on graphs, and in the process point out problems with some of these approaches. Following in the footsteps of the review by Shuman et al. [**18**], we divide such transforms into two general categories.

2.3.1. Methods based on the Graph Fourier Transform

Before we can present the first category of transforms, some background is necessary. These transforms makes use of the *graph Fourier transform*, which was developed by Hammond et al. [**34**]. Noting that the classical discrete Fourier transform amounts to taking inner products with the complex exponentials, Hammond defines the graph Fourier transform for a signal $\boldsymbol{f} \in \mathbb{R}^N$ on a graph by replacing the complex exponentials with the Laplacian eigenvectors (either those of L or L_{sym})[6]:

$$\widehat{\boldsymbol{f}}(l) := \langle \boldsymbol{f}, \boldsymbol{\phi}_l \rangle \,, \tag{2.20}$$

or in matrix notation,

$$\widehat{\boldsymbol{f}} := \boldsymbol{\Phi}^* \boldsymbol{f}. \tag{2.21}$$

Here, $\boldsymbol{\Phi}$ is the matrix whose columns are the orthonormal eigenvectors of the Laplacian matrix: $\boldsymbol{\Phi} := [\, \boldsymbol{\phi}_0 \mid \boldsymbol{\phi}_1 \mid \cdots \mid \boldsymbol{\phi}_{N-1} \,] \in \mathbb{R}^{N \times N}$. Thus, the graph Fourier transform is simply expanding the signal in terms of the Laplacian eigenvectors[7]. Of course, the inverse graph Fourier transform is simply $\boldsymbol{f} = \boldsymbol{\Phi}\widehat{\boldsymbol{f}}$. We say that the signal $\boldsymbol{f}$ is in the *vertex domain*, whereas $\widehat{\boldsymbol{f}}$ belongs to the *graph Fourier domain*.

The graph Fourier domain provides us with not only a means of analyzing signals, but also a means of generating them. A *kernel* is a function $\hat{g} : \mathbb{R} \to \mathbb{R}$, which we use to define a signal in the graph Fourier domain, $\widehat{\boldsymbol{g}}(l) := \hat{g}(\lambda_l)$, and in turn a signal $\boldsymbol{g} = \boldsymbol{\Phi}\widehat{\boldsymbol{g}}$ in the vertex domain. To

[6]There is a slight abuse of notation in (2.20), since $\boldsymbol{f} = (\boldsymbol{f}(1), \ldots, \boldsymbol{f}(N))$ yet $\widehat{\boldsymbol{f}} = (\widehat{\boldsymbol{f}}(0), \ldots, \widehat{\boldsymbol{f}}(N-1))$. Conveniently, this indexing misdemeanor is avoided when using matrix notation.

[7]Clearly, the graph Fourier transform depends on the choice of eigenvectors, which is certainly not unique: if a Laplacian eigenvalue has multiplicity > 1 then we can choose different sets of eigenvectors which span its eigenspace. Eigenvalue multiplicity notwithstanding, the signs of the eigenvectors are arbitrary. However, it is assumed that once a choice of eigenvectors is made it remains fixed.

clarify our notation, we use regular fonts to denote functions (i.e., kernels) and bold fonts to denote vectors (i.e., signals). Furthermore, signals in the vertex domain are indexed by $n \in [1, N]$ and signals in the graph Fourier domain are indexed by $l \in [0, N-1]$.

A number of classical signal processing techniques have been generalized to the graph setting by appealing to classical relations involving the Fourier transform[8]. For a classical signal $\boldsymbol{f}$, modulation is defined as $M_\omega f(t) = \mathrm{e}^{2\pi \mathrm{i} \omega t} f(t)$. For the graph setting, Shuman et al. define *generalized modulation* by replacing the complex exponential with a Laplacian eigenvector [**76**]:

$$M_l \boldsymbol{f} := \sqrt{N} \boldsymbol{\phi}_l \odot \boldsymbol{f}, \tag{2.22}$$

where $\odot$ denotes elementwise multiplication of the vectors. Thus, generalized modulation is defined for integers $l \in [0, N-1]$. For signals $f, g \in L^1(\mathbb{R})$, the convolution theorem tells us that $\widehat{(f * g)}(\omega) = \hat{f}(\omega)\hat{g}(\omega)$. Exploiting this relationship, Shuman et al. define the *generalized convolution* for signals $\boldsymbol{f}, \boldsymbol{g} \in \mathbb{R}^N$ on a graph as [**34**, **76**]

$$\widehat{(\boldsymbol{f} * \boldsymbol{g})}(l) := \widehat{\boldsymbol{f}}(l)\widehat{\boldsymbol{g}}(l). \tag{2.23}$$

Note that $\boldsymbol{f} * \boldsymbol{g} = \boldsymbol{g} * \boldsymbol{f}$. Typically, $\boldsymbol{g}$ is defined by a kernel $\hat{g}$, which explains why the generalized convolution is often referred to as spectral filtering: convolving a signal $\boldsymbol{f}$ with $\boldsymbol{g}$ attenuates/amplifies its Laplacian eigenvector expansion coefficients by factors $\widehat{\boldsymbol{g}}(l) = \hat{g}(\lambda_l)$, as seen in (2.23). Taking the inverse graph Fourier transform of this, we can express the generalized convolution in the vertex domain as

$$\begin{aligned} (\boldsymbol{f} * \boldsymbol{g})(n) &= \sum_{l=0}^{N-1} \widehat{\boldsymbol{f}}(l)\widehat{\boldsymbol{g}}(l)\boldsymbol{\phi}_l(n) \\ \boldsymbol{f} * \boldsymbol{g} &= \boldsymbol{\Phi}(\widehat{\boldsymbol{f}} \odot \widehat{\boldsymbol{g}}). \end{aligned} \tag{2.24}$$

Given that classical wavelets are translations and dilations of a single mother wavelet, attempts have been made to extend these two operations to the graph setting. Let $\delta(t)$ denote the delta function, let $\delta_x(t) := \delta(t-x)$, and let $\boldsymbol{\delta}_k := \mathbf{1}_{\{k\}} \in \mathbb{R}^N$ (i.e., the Kronecker delta). Recall that translation of a function is equivalent to convolution with the delta function, $T_x f(t) := f(t-x) = f * \delta_x(t)$.

[8]Although the generalized convolution, generalized translation, and generalized dilation are utilized in [**34**], they were not formally defined until [**76**].

The *generalized translation* is analogously defined as [**34**, **76**]

$$\begin{aligned} T_k \boldsymbol{f} &:= \sqrt{N}\, \boldsymbol{f} * \boldsymbol{\delta}_k \\ &= \sqrt{N} \sum_{l=0}^{N-1} \widehat{\boldsymbol{f}}(l) \boldsymbol{\phi}_l^*(k) \boldsymbol{\phi}_l. \end{aligned} \tag{2.25}$$

Generalized translation is defined for nodes $k \in \{1, \ldots, N\}$. Let us define dilation of a function as $\mathcal{D}_s f(t) := \frac{1}{s} f\left(\frac{t}{s}\right)$ (which differs from the L^2-norm preserving translation operator D used in (2.2) for scaling and wavelet functions). Appealing to the fact that $\widehat{(\mathcal{D}_s f)}(\omega) = \hat{f}(s\omega)$, the *generalized dilation* is defined as [**18**, **34**]

$$\begin{aligned} \widehat{(\mathcal{D}_j \boldsymbol{g})}(\lambda) &:= \hat{g}(j\lambda) \\ \mathcal{D}_j \boldsymbol{g} &= \sum_{l=0}^{N-1} \hat{g}(j\lambda_l) \boldsymbol{\phi}_l. \end{aligned} \tag{2.26}$$

Note that the generalized dilation is only defined for a signal $\boldsymbol{g}$ that is generated via a kernel $\hat{g}$.

With these generalized operations in place, we can now discuss several wavelet-like transforms that have been developed for signals on graphs. Hammond et al. proposed the spectral graph wavelet transform (SGWT) [**34**], which uses low- and high-pass kernels $\hat{h}$ and $\hat{g}$, respectively, to define scaling and wavelet functions as

$$\phi_k^{\mathrm{SGWT}} = \frac{1}{\sqrt{N}} T_k \boldsymbol{h} \tag{2.27a}$$

$$\psi_{j,k}^{\mathrm{SGWT}} = \frac{1}{\sqrt{N}} T_k \mathcal{D}_j \boldsymbol{g}. \tag{2.27b}$$

Their transform proceeds by taking inner products of these with the signal $\boldsymbol{f}$ on the graph, which yields an overcomplete set of scaling and wavelet coefficients. However, the design of the scaling kernel $\hat{h}$ is uncoupled from that of the wavelet kernel $\hat{g}$, and so we cannot generate the wavelet functions via refinement relations involving the scaling functions. Furthermore, the SGWT generates an overcomplete wavelet frame, since we translate the scaling and wavelet functions to all nodes $k \in \{1, \ldots, N\}$, and in the case of the wavelets, we do so at multiple scales $j \in \{1, \ldots, J\}$. (We refer the reader to [**34**] for details on selecting J.)

Building upon the graph Fourier generalizations introduced in [**34**], Shuman et al. developed the windowed graph Fourier transform. Recall that the classical windowed Fourier transform entails taking inner products of a signal f with atoms that are modulated translations of a window function g. Shuman et al. generalized this to the graph setting in a straightforward manner, defining the windowed graph Fourier transform atoms as generalized modulations of generalized translations of a window function $\boldsymbol{g} \in \mathbb{R}^N$ [**76**, **78**]. When this window function is generated by a kernel satisfying $\hat{g}(0) \neq 0$, the windowed graph Fourier atoms form a frame for signals on the graph.

Shuman et al. have also developed spectrum-adapted tight frames for signals consisting of generalized translations of M signals, $\{T_k \boldsymbol{g}_m\}_{k \in [1,N],\ m \in [1,M]}$. Each $\boldsymbol{g}_m$ is generated using a kernel $\hat{g}_m$ that is adapted to the spectrum of the graph: $\boldsymbol{g}_m = \boldsymbol{\Phi} \widehat{\boldsymbol{g}}_m = \sum_{l=0}^{N-1} \hat{g}_m(\lambda_l) \boldsymbol{\phi}_l$. In simple terms, "spectrum-adapted" means that each $\boldsymbol{g}_m$ is designed to capture a roughly equal, unique portion of the Laplacian spectrum. After all, if $\hat{g}_m(\lambda_l) \approx 0$ for $l \in [0, N-1]$, then it is of little use for analyzing signals on the graph. On the other hand, we do not want to have two kernels such that $\hat{g}_{m_1}(\lambda_l) \approx \hat{g}_{m_2}(\lambda_l)$ for $l \in [0, N-1]$ because there will be too much overlap in the information that they capture about signals.

Narang and Ortega have developed transforms for signals on graphs by extending wavelet filter-banks to the graph setting [**51**, **52**]. They introduce a notion of downsampling/upsampling in the graph setting which is based on decomposing an arbitrary graph into bipartite subgraphs, thereby affording a notion of "every other node." Their graph-QMF transform [**51**] yields orthogonal wavelets, but the basis vectors are not localized on the graph. On the other hand, their graphBior transform [**52**] yields biorthogonal wavelets with compact support. Both of these transforms are critically sampled (unlike [**34**, **76**, **77**, **78**]), and both yield perfect reconstruction.

Each of these transforms relies heavily upon the graph Fourier transform, which effectively uses the Laplacian eigenvalues and eigenvectors in place of the frequencies and complex exponentials in the classical discrete Fourier transform. While tempting to make this substitution, there are at least two fundamental problems in doing so. First, it is difficult to know the essential support of the Laplacian eigenvectors a priori, which strongly depends on the structure of the graph: sometimes they are completely global, like those of P_N, whereas in other cases they may be quite localized, e.g., on dendritic trees of neurons, as illustrated in Figure 2.8 [**50**, **71**, **72**]. Hence, it is worth

controlling the support of the eigenvectors explicitly. (In fact, this observation has led us to our HGLET construction using recursive graph partitioning, as discussed in Chapter 4.)

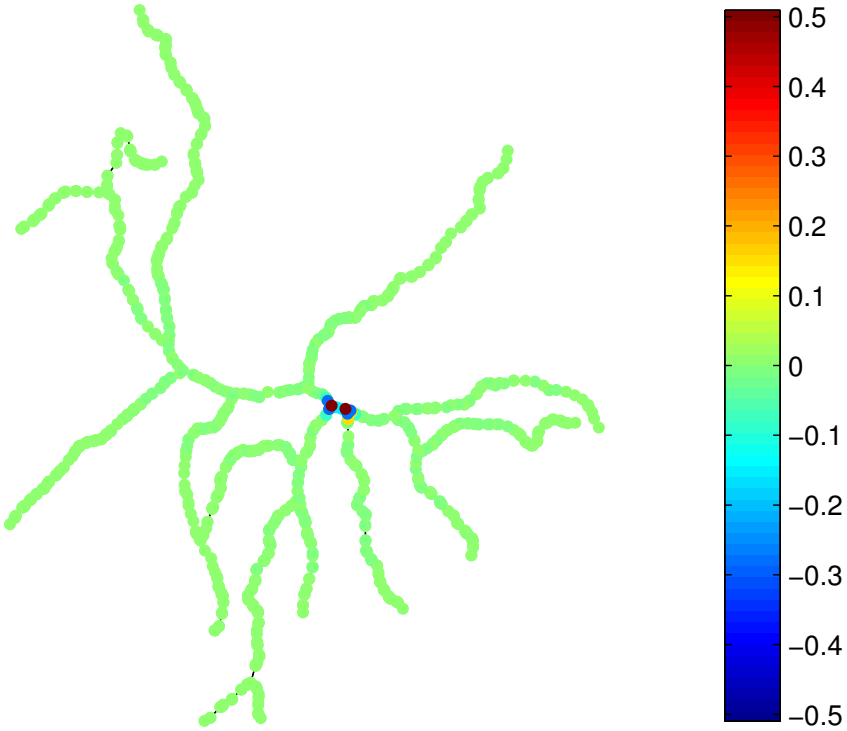


FIGURE 2.8. Unnormalized Laplacian eigenvector ϕ_{1142} on a dendritic tree ($N = 1154$) provides an example of a Laplacian eigenfunction whose support is highly localized. The corresponding eigenvalue is $\lambda_{1142} = 4.3829$. This is a recreation of Figure 5 from [**71**].

The second problem of viewing the Laplacian eigenfunctions as the equivalent of the Fourier basis functions is the intricate relationship between the frequencies and the Laplacian eigenvalues. For very simple 1-D graphs such as P_N and C_N, the eigenvectors are the Fourier basis vectors and the eigenvalues are a nondecreasing function of their corresponding frequencies, as clearly shown in (2.15) and (2.17) in the case of P_N. (In order to observe this relationship for C_N, it is necessary to use the sine and cosine components of (2.19) and sort the eigenvalues in a nondecreasing manner.) Consequently, on P_N and C_N we can develop wavelets using the classical Littlewood-Paley theory [**40**, §2.4] by appropriately partitioning the eigenvalue axis into blocks and combining the corresponding eigenfunctions. However, as soon as the domain becomes even slightly

more complicated, the situation completely changes: we cannot view the eigenvalues as a simple monotonic function of frequency anymore. For example, consider a long but thin strip in $\mathbb{R}^2$, and suppose that the domain is discretized as $P_{N_x} \times P_{N_y}$ $(N_x > N_y)$. Extending (2.15) from a 1-D path graph to this 2-D grid, the eigenpairs for the unnormalized Laplacian are:

$$\left.\begin{aligned} \lambda_l &= 4\left[\sin^2\left(\frac{\pi l_x}{2N_x}\right) + \sin^2\left(\frac{\pi l_y}{2N_y}\right)\right] \\ \boldsymbol{\phi}_l(x,y) &= \cos\left(\frac{\pi l_x\left(x-\frac{1}{2}\right)}{N_x}\right)\cos\left(\frac{\pi l_y\left(y-\frac{1}{2}\right)}{N_y}\right) \end{aligned}\right\} \quad \text{where} \quad \begin{aligned} & l = 0,\ldots,N_xN_y-1 \\ & l_x = 0,\ldots,N_x-1 \\ & l_y = 0,\ldots,N_y-1 \\ & x = 1,\ldots,N_x \\ & y = 1,\ldots,N_y. \end{aligned} \tag{2.28}$$

Let $\{\lambda_l\} = \{\lambda_{(l_x,l_y)}\}$ be ordered in the nondecreasing manner. In this case, the smallest eigenvalue is clearly $\lambda_0 = \lambda_{(0,0)} = 0$, and the corresponding eigenvector is constant. The second smallest eigenvalue λ_1 is $\lambda_{(1,0)} = 4\sin^2(\pi/2N_x)$, since $\pi/2N_x < \pi/2N_y$, and its eigenvector has one oscillation in the x-direction. But, how about λ_2? Even for such a simple situation there are several possibilities for λ_2, depending on N_x and N_y. If $N_x > 2N_y$, then $\lambda_2 = \lambda_{(2,0)} < \lambda_{(0,1)}$. On the other hand, if $N_y < N_x < 2N_y$, then $\lambda_2 = \lambda_{(0,1)} < \lambda_{(2,0)}$. More generally, if $KN_y < N_x < (K+1)N_y$ for some $K \in \mathbb{N}$, then $\lambda_l = \lambda_{(l,0)} = 4\sin^2(\pi l/2N_x)$ for $l = 0,\ldots,K$. Yet the next eigenvalue is $\lambda_{K+1} = \lambda_{(0,1)} = 4\sin^2(\pi/2N_y)$, followed by $\lambda_{K+2} = \lambda_{(K+1,0)} = 4\sin^2(\pi(K+1)/2N_x)$. As one can see from this, the mapping between l and (l_x, l_y) is quite nontrivial. Notice that $\phi_{(l,0)}$ has l oscillations in the x-direction for $0 \le l \le K$, whereas $\phi_{(0,1)}$ has only one oscillation in the y-direction. In other words, all of a sudden the eigenvalue of a completely different type of oscillation sneaks into the eigenvalue sequence, as illustrated in Figure 2.9 for a 101×10 grid. Hence, on a general domain or a general graph, by simply looking at the Laplacian eigenvalue sequence $\{\lambda_l\}_{l=0,1,\ldots}$, it is almost impossible to organize the eigenpairs into physically meaningful dyadic blocks and apply the Littlewood-Paley approach unless the underlying domain is of very simple nature, e.g., P_N or C_N. For complicated domains, the notion of "frequency" is not well-defined anymore, and thus

wavelet construction methods that rely on the Littlewood-Paley theory may lead to unexpected problems on general graphs[9].

(a) $\boldsymbol{\phi}_1$ (b) $\boldsymbol{\phi}_{10}$ (c) $\boldsymbol{\phi}_{11}$

FIGURE 2.9. Unnormalized Laplacian eigenvectors (a) $\boldsymbol{\phi}_1$, (b) $\boldsymbol{\phi}_{10}$, and (c) $\boldsymbol{\phi}_{11}$ on an unweighted 101×10 grid. Eigenvectors $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{10}$ have $1, \ldots, 10$ oscillations in the x-direction, whereas $\boldsymbol{\phi}_{11}$ has 1 oscillation in the y-direction.

Furthermore, the generalized transforms do not agree with their classical counterparts. For example, Figure 2.10 shows dilations of a signal $\boldsymbol{g}$ defined using the kernel $\hat{g}(\lambda) = \mathrm{e}^{-10\lambda}$ on the unweighted cycle C_{64}. (It would have been ideal to use a narrow pulse as the input signal, but this is not readily feasible because the generalized dilation requires that the signal be defined by a kernel $\hat{g} : \mathbb{R} \to \mathbb{R}$.) The differences between the two figures are due to the eigenvectors used: (a) uses the complex exponentials, whereas (b) uses the output of MATLAB's eig function. Although the signal $\boldsymbol{g}$ and its generalized dilations $\mathcal{D}_2\boldsymbol{g}$ and $\mathcal{D}\boldsymbol{g}_4$ in both figures are generated using the same kernel $\hat{g}$, the results differ signficantly. This illustrates that for a kernel $\hat{g}$, both the signal $\boldsymbol{g}$ and its generalized dilations $\mathcal{D}_j\boldsymbol{g}$ depend on the particular choice of eigenvectors.

[9]We want to point out that there have been some efforts to develop the Littlewood-Paley theory on very general and abstract setups such as abstract measure space [**48**] or the so-called spaces of homogeneous type [**21**, Ch. 3].

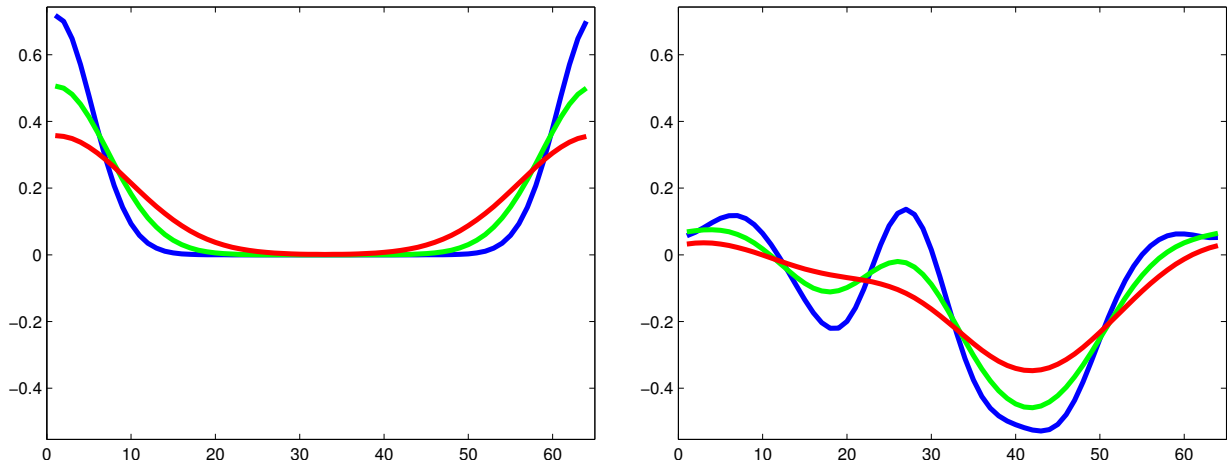
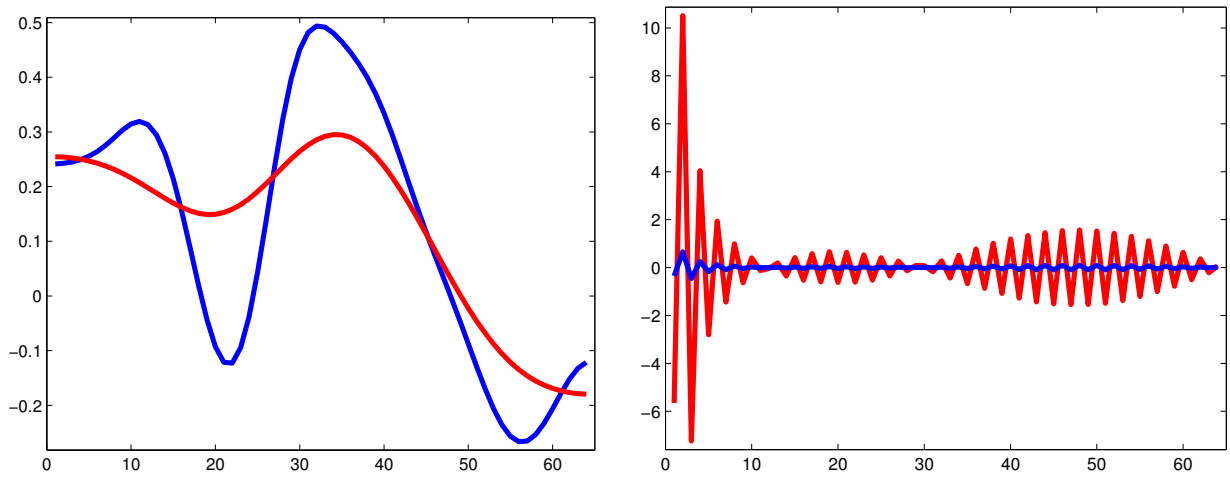


FIGURE 2.10. A signal $\boldsymbol{g}$ (blue) defined by the kernel $\hat{g}(\lambda) = \mathrm{e}^{-10\lambda}$ and the generalized dilations $\mathcal{D}_2\boldsymbol{g}$ (green) and $\mathcal{D}_4\boldsymbol{g}$ (red) on an unweighted cycle of length $N = 64$. The results differ due to the different choices of Laplacian eigenvectors: (a) complex exponentials and (b) the output of MATLAB's eig function.

Moreover, generalized dilation is not really dilating the signal at all, it is simply modifying the contributions of the Laplacian eigenvectors. As seen from (2.26), the contribution of $\boldsymbol{\phi}_l$ to the resulting signal is changed from $\hat{g}(\lambda_l)$ to $\hat{g}(j\lambda_l)$. This explains why in Figure 2.10 we observe a general smoothing of the signal: since $\hat{g}(4\lambda_l) < \hat{g}(2\lambda_l) < \hat{g}(\lambda_l)$ for $l > 0$, the high-frequency components are attenuated as j increases. If instead we chose a kernel such that $\hat{g}(j\lambda_l) > \hat{g}(\lambda_l)$, the dilated signals $\mathcal{D}_j\boldsymbol{g}$ would be more oscillatory than the original signal $\boldsymbol{g}$. Figure 2.11 illustrates exactly this for the unweighted path graph P_{64}. The signal in Figure 2.11a is generated via the same kernel as before, $\hat{g}(\lambda) = \mathrm{e}^{-10\lambda}$, and once again the dilated signal $\mathcal{D}_4\boldsymbol{g}$ is a smoother version of the original. On the other hand, the signal in Figure 2.11b is generated via the kernel $\hat{g}(\lambda) = \left(\frac{\lambda}{8}\right)^2$. In this case we have that $\hat{g}(4\lambda_l) > \hat{g}(\lambda_l)$ for $l > 0$, and accordingly the dilated signal is much more oscillatory than the original.



(a) (b)

FIGURE 2.11. A signal $\boldsymbol{g}$ (blue) and its generalized dilation $\mathcal{D}_4\boldsymbol{g}$ (red) for an unweighted path graph of length $N = 64$. The kernel used in (a) is $\hat{g}(\lambda) = \mathrm{e}^{-10\lambda}$, whereas the kernel in (b) is $\hat{g}(\lambda) = \left(\frac{\lambda}{8}\right)^2$.

As with the generalized dilation, the generalized translation also fails to produce the desired results for signals on simple graphs. In Figure 2.12 we show several generalized translations on an unweighted cycle, which we use because the notion of translation is clearly defined. The generalized translation only produces the correct results when using the complex exponentials as the eigenvectors, as in Figure 2.12a. This is because the complex exponential eigenvectors satisfy $\sqrt{N}\boldsymbol{\phi}_l^*(k)\boldsymbol{\phi}_l = S_N^{k-1}\boldsymbol{\phi}_l$, where S_N is the matrix which circularly shifts the entries in the columns down one position, i.e.,

$$S_N := \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix}.$$

Substituting this into (2.25), we find that generalized translation using the complex exponentials is equivalent to multiplication by a circular shift matrix. However, the generalized translation does not work so nicely when using alternative, real-valued sets of eigenvectors: the sine and cosine components of the complex exponentials in Figure 2.12b, and the output of MATLAB's eig function

in Figures 2.12c and 2.12d. These are, after all, more realistic scenarios, since for a general graph the Laplacian eigenvectors returned by a numerical algorithm will be real-valued and will not possess the unique properties of the complex exponentials. In Figures 2.12b and 2.12c, the generalized translations $T_{128}\boldsymbol{f}$ and $T_{192}\boldsymbol{f}$ are completely different from the original signal $\boldsymbol{f}$. Furthermore, as the figures show, they are not even translations of each other. In Figure 2.12d, we again find that the generalized translations differ from the original signal. However, because here we have defined $\boldsymbol{f}$ via a kernel $\hat{f}(\lambda_l) = \mathrm{e}^{-10\lambda_l}$, the generalized translations $T_{128}\boldsymbol{f}$ and $T_{192}\boldsymbol{f}$ are translations of one another. To explain the behavior of the generalized translation, observe from (2.25) that the nth entry of $T_k\boldsymbol{f}$ is given by $T_k\boldsymbol{f}(n) = \sum_l \widehat{\boldsymbol{f}}(l)\boldsymbol{\phi}_l^*(k)\boldsymbol{\phi}_l(n)$. When using real-valued eigenvectors (as is the case when working with a general graph) and when the signal $\boldsymbol{f}$ is generated by a nonnegative kernel $\hat{f}$, each of the summands for the kth entry is nonnegative and hence $T_k\boldsymbol{f}(k)$ is positive. And by the nodal domain theorem [**4**, **20**], we can infer that the more closely connected node i is to node k, the more likely the summation $T_k\boldsymbol{f}(i) = \sum_l \widehat{\boldsymbol{f}}(l)\boldsymbol{\phi}_l^*(k)\boldsymbol{\phi}_l(i)$ will be positive. Thus, as we have seen the generalized translation really is not a translation operator at all, but rather a kernel localization operator.

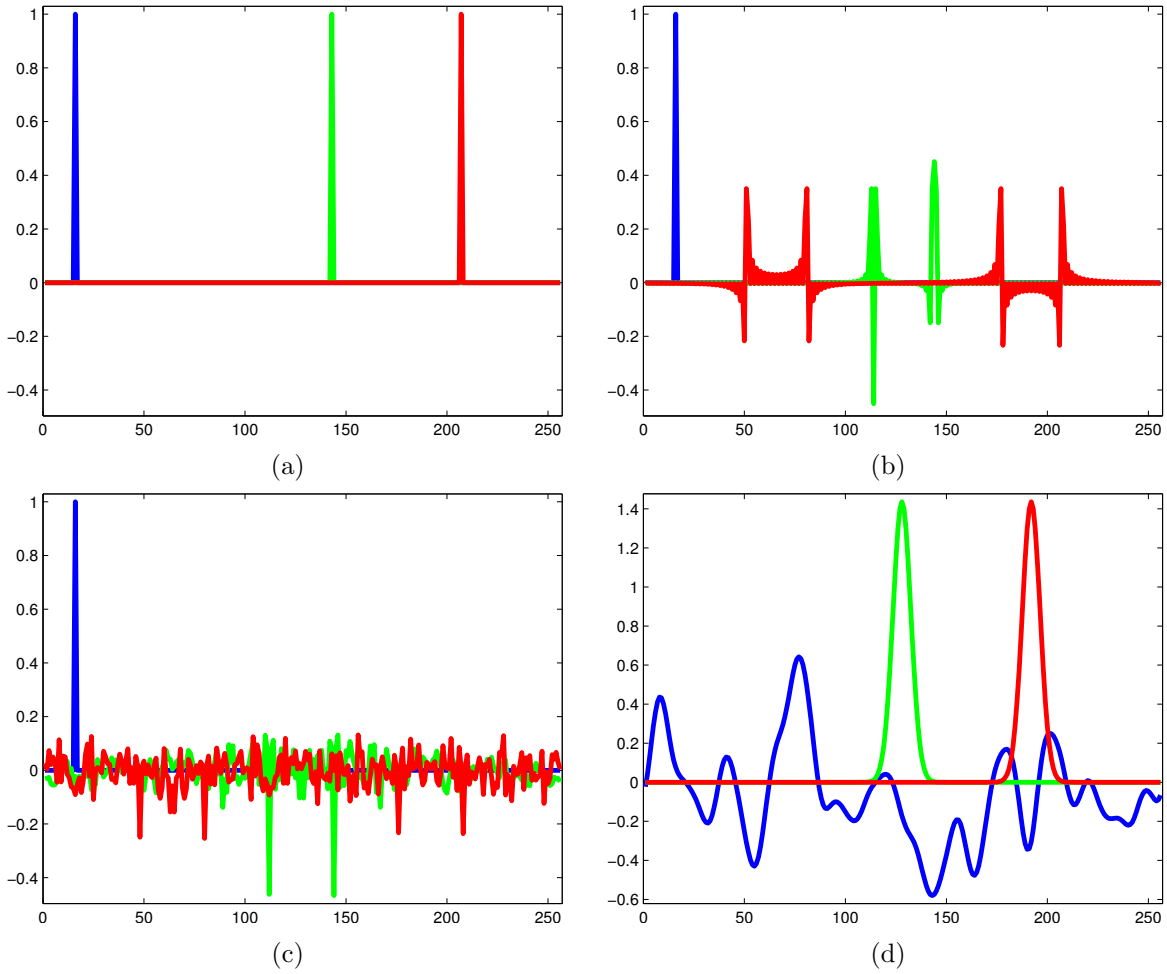


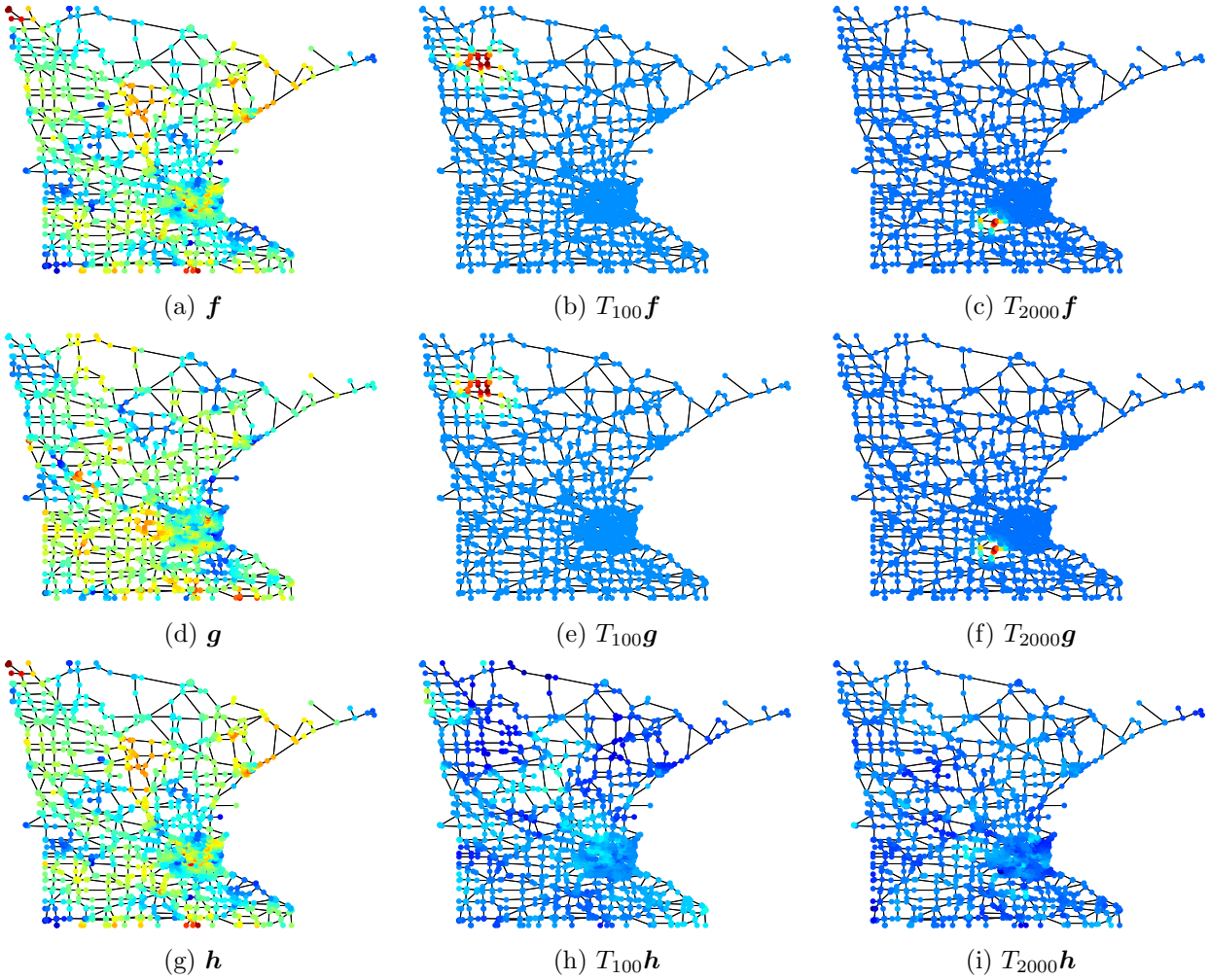
FIGURE 2.12. Examples of generalized translations of a signal $\boldsymbol{f}$ (blue) and its generalized translations $T_{128}\boldsymbol{f}$ (green) and $T_{192}\boldsymbol{f}$ (red) on an unweighted cycle of length $N = 256$. (a) The original signal $\boldsymbol{f} = \boldsymbol{\delta}_{10}$ is a Kronecker delta centered at node 10, and the Laplacian eigenvectors are specified as the complex exponentials. (b) We use the same signal as before, but we specify the eigenvectors as the sine and cosine components of the complex exponentials. (c) Again we use $\boldsymbol{f} = \boldsymbol{\delta}_{10}$, but the eigenvectors are the output of MATLAB's eig function. (d) We use the eigenvectors from eig, but this time our signal $\boldsymbol{f}$ is generated via the kernel $\hat{f}(\lambda_l) = \mathrm{e}^{-10\lambda_l}$.

Examples of generalized translation on a more interesting graph will help to clarify this even more. Therefore, we take this opportunity to introduce the Minnesota road network, which is often used as an example in research publications ([**18**, **34**, **51**, **52**, **62**, **63**, **76**, **77**, **78**], to name a few examples). The $M = 3302$ edges of the graph correspond to roads, and the $N = 2640$ nodes correspond to intersections. The original graph is unweighted and its nodes are labeled from

top-to-bottom. We can convert it to a weighted graph by specifying the edge weights as, say, the inverse Euclidean distances between adjacent nodes. Since 4 pairs of nodes have the same spatial coordinates, we discard 4 nodes when converting it to a weighted graph using inverse Euclidean distances, and thus the resulting graph has $N = 2636$ nodes and $M = 3293$ edges.

Figure 2.13 shows some generalized translation experiments using the Minnesota road network. Replicating the results in [**18**], we use the kernel $\hat{f}(\lambda_l) = \mathrm{e}^{-5\lambda_l}$ and we show the generalized translations $T_{100}\boldsymbol{f}$ and $T_{2000}\boldsymbol{f}$; we also show the original signal $\boldsymbol{f}$, which is not shown in [**18**]. We immediately see from the top row (Figs. 2.13a-c) that the generalized translation really is not a translation operator, as remarked before. We then permute the ordering of the nodes for the second and third rows of figures (Figs. 2.13d-i), and we define the signal $\boldsymbol{g}$ on this permuted graph using the same kernel as before: $\hat{g}(\lambda_l) = \hat{f}(\lambda_l) = \mathrm{e}^{-5\lambda_l}$. The signals $\boldsymbol{f}$ and $\boldsymbol{g}$ differ because some of the Laplacian eigenvectors of this permuted graph differ in sign from those of the original graph. However, their graph Fourier transforms $\widehat{\boldsymbol{f}}$ and $\widehat{\boldsymbol{g}}$ are the same because they are defined by the same kernel, and thus their generalized translations $T_k\boldsymbol{f}$ and $T_k\boldsymbol{g}$ are the same. Therefore, Figures 2.13d-f illustrate that the generalized translation operates on a signal's graph Fourier transform $\widehat{\boldsymbol{g}}$, not the signal $\boldsymbol{g}$ itself.

On the other side of the coin, we construct the signal $\boldsymbol{h}$ by permuting the entries of $\boldsymbol{f}$ in the same manner that the nodes were permuted, and as such the signals are the same. However, their graph Fourier transforms differ due to the sign differences of the eigenvectors, and as a result the generalized translations $T_{100}\boldsymbol{h}$ and $T_{2000}\boldsymbol{h}$ do not match $T_{100}\boldsymbol{f}$ and $T_{2000}\boldsymbol{f}$, respectively. Thus, Figures 2.13g-i further illustrate that the generalized translation is, in fact, a kernel localization operator. To summarize these experiments using the Minnesota road network: it remains unclear as to what a translation of a signal on a graph should look like, or if translation on a graph is even meaningful in the first place, but what is clear is that the so-called generalized translation is not the answer.



(a) $\boldsymbol{f}$ (b) $T_{100}\boldsymbol{f}$ (c) $T_{2000}\boldsymbol{f}$

(d) $\boldsymbol{g}$ (e) $T_{100}\boldsymbol{g}$ (f) $T_{2000}\boldsymbol{g}$

(g) $\boldsymbol{h}$ (h) $T_{100}\boldsymbol{h}$ (i) $T_{2000}\boldsymbol{h}$

FIGURE 2.13. (a) A signal $\boldsymbol{f}$ on the MN road network generated via the kernel $\hat{f}(\lambda_l) = \mathrm{e}^{-5\lambda_l}$ and its generalized translations (b) $T_{100}\boldsymbol{f}$ and (c) $T_{2000}\boldsymbol{f}$. (d) We permuted the nodes of the graph and generated the signal $\boldsymbol{g}$ using the same kernel as before, $\hat{g}(\lambda_l) = \mathrm{e}^{-5\lambda_l}$. As a result of the permutation, the signs of the eigenvectors differ from those corresponding to the original graph, and hence the signals $\boldsymbol{f}$ and $\boldsymbol{g}$ differ. However, the generalized translations (e) $T_{100}\boldsymbol{g}$ and (f) $T_{2000}\boldsymbol{g}$ are the same as $T_{100}\boldsymbol{f}$ and $T_{2000}\boldsymbol{f}$, respectively. (g) Again working with the permuted graph, we form the signal $\boldsymbol{h}$ by permuting the entries in $\boldsymbol{f}$. While $\boldsymbol{f}$ and $\boldsymbol{h}$ are the same, the generalized translations (h) $T_{100}\boldsymbol{h}$ and (i) $T_{2000}\boldsymbol{h}$ differ from $T_{100}\boldsymbol{f}$ and $T_{2000}\boldsymbol{f}$, respectively. (The color scheme is the same for the three figures in each column.)

Given these issues with the graph Fourier transform and the generalized operations, we do not use these techniques in our own research. Rather than developing transforms that depend on a notion of frequency, we instead develop multiscale transforms. But before discussing our own

developments, we continue our review of previous research.

2.3.2. Methods based on Vertex Transformations

Whereas the previously mentioned transforms utilize operations in the graph Fourier domain, we now present various transforms that utilize operations in the vertex domain.

A common strategy is to utilize a hierarchical tree that organizes the vertices of a graph into clusters at various scales, an example of which can be seen in Figure 2.14. We denote these sets of vertices by $V_k^j \subseteq V =: V_0^0$, where j denotes the scale index (or level) in the hierarchical tree and k indexes the sets on level j. We define G_k^j to be the subgraph of G that is induced by restricting to the vertices in V_k^j and the edges between them. We often use the term "region" to refer to a subgraph G_k^j, especially when the nodes of the graph lie in $\mathbb{R}$, $\mathbb{R}^2$, or $\mathbb{R}^3$ because this emphasizes the spatial organization of the subgraphs. In addition, we use the term "subregion" to refer to a child subgraph.

Unless the hierarchical tree is provided along with the graph, it must be generated in one of two ways. The first is to utilize a bottom-up clustering approach in which we start with the individual vertices of the graph and recursively group them together according to their similarity, as indicated by the weight matrix W. The second method is to use a top-down partitioning approach in which we start with the entire graph and repeatedly partition it into subgraphs, typically in a manner that strives to generate subgraphs that are roughly equal in size while keeping similar vertices grouped together. Graph transforms that utilize a hierarchical tree have different requirements for the structure of the tree, but a typical set of requirements can be succinctly described as follows: (i) the root node of the hierarchical tree contains all N vertices of the graph; (ii) the leaf nodes of the tree each contain a single vertex; and (iii) each non-leaf node is split into two nodes. For a tree satisfying these requirements, such as the tree in Figure 2.14, it is straightforward to show that there are $N - 1$ non-leaf nodes.

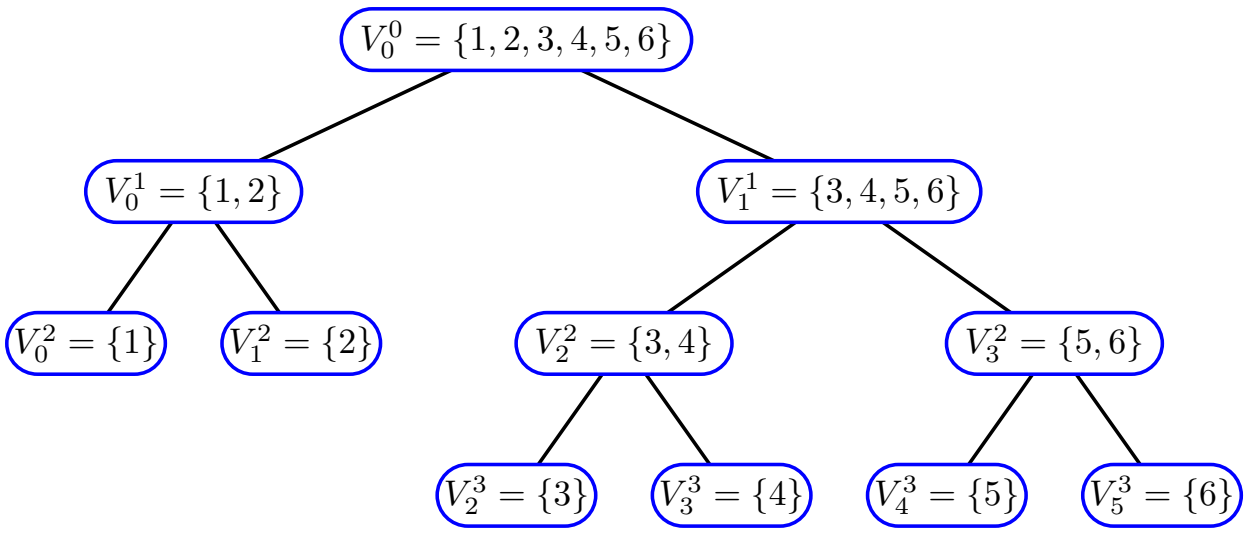


FIGURE 2.14. An example of a hierarchical tree for a graph with $N = 6$ nodes. We denote the sets of vertices by V_k^j, where j denotes their level and k indexes the sets on level j. At the top of the tree (i.e., on the coarsest level, $j = 0$) we have the root node, which includes all the vertices. Each leaf node in this tree corresponds to a single vertex i from the graph.

Using a hierarchical tree, several groups of researchers have generalized the Haar wavelet transform to the graph setting [**10**, **32**, **43**, **49**]. From the Haar scaling and wavelet functions shown in Figures 2.1a and 2.1d, it is easily seen that Haar scaling coefficients are averages of a function on an interval and that the wavelet coefficients are the differences of the averages on the two subintervals. Accordingly, each of these generalized Haar transforms proceeds by assigning one "wavelet" coefficient to each of the $N-1$ parent (i.e., non-leaf) nodes in the hierarchical tree, which is computed by taking the difference of the averages on its two children nodes. The remaining expansion coefficient is the scaling coefficient on the root node of the tree, which is equal to $\sqrt{N}$ times the average of the signal over the entire graph. The generalized Haar basis is orthonormal, and its coefficients range in scale from local to global.

Szlam et al. utilize a recursive partitioning of a graph to generate an orthonormal basis in a couple of different ways [**86**]. Their first method entails constructing the generalized Haar basis and then smoothing the basis functions using diffusion operators. As this smoothing operation destroys the orthogonality of their basis, their final step is to perform an orthogonalization procedure.

Their second approach is to generalize the local cosine dictionary on each subgraph using the graph/manifold version of the folding and unfolding operators initially proposed by Coifman and Meyer for functions on the real line (or on the regular 1-D lattice) [**12**]. Unfortunately, based on our considerable experience with the local cosine dictionary on regular lattices [**69**], we can predict that such generalized local cosines may not work well in practice. In fact, the direct use of such folding/unfolding operations in the graph setting may be unnecessary or even harmful for many applications.

Sharon and Shkolnisky use a subset of the Laplacian eigenvectors and a recursive partitioning tree to construct a multiresolution analysis and consequently multiwavelet bases [**74**]. (*Multiwavelets* are similar to the standard wavelets covered in §2.1, but the multiwavelet basis consists of translations and dilations of multiple scaling and/or wavelet functions [**56**, **57**].) The resulting basis depends upon a user-specified constant $1 \leq m \leq N$, with $m = 1$ leading to the generalized Haar basis and $m = N$ leading to the Laplacian eigenvectors. The first step of their algorithm is to generate a multiresolution analysis; that is, a sequence of nested approximation spaces $\mathcal{V}_0 \subset \mathcal{V}_1 \subset \mathcal{V}_2 \cdots$ corresponding to levels of the tree[10]. They do this by starting with the graph G and computing the first m eigenvectors of $L_{\text{rw}}(G)$, and they define $\mathcal{V}_0 := \text{span}\{\boldsymbol{\phi}_0^{\text{rw}}, \ldots, \boldsymbol{\phi}_{m-1}^{\text{rw}}\}$. Next, for each subgraph G_k^j in the partitioning tree with m or more vertices (i.e., $|V_k^j| \geq m$), they generate m linearly independent vectors by restricting the m global eigenvectors to the vertices V_k^j of the subgraph; if these restrictions are not linearly independent, they include Laplacian eigenvectors of $L_{\text{rw}}(G_k^j)$ as needed to achieve m linearly independent vectors. Finally, for subgraphs G_k^j for which neither of their children subgraphs have m or more vertices, they include eigenvectors of $L_{\text{rw}}(G_k^j)$ as needed to achieve $|V_k^j|$ linearly independent vectors. For each level j of the hierarchical tree, they define the space $\mathcal{V}_j$ as the span of all the vectors generated on the subgraphs of level j. Since the vectors on level $j+1$ are formed by restricting the vectors on level 0, these spaces satisfy $\mathcal{V}_j \subset \mathcal{V}_{j+1}$. Furthermore, letting $\mathcal{V}$ denote the space of all signals on the graph and using j_m to denote the deepest level of the tree having a subgraph with m or more vertices, we have that $\mathcal{V}_{j_m} = \mathcal{V}$. In the second step of their algorithm, they use an orthogonalization procedure on these nested approximation

[10]As explained in Table 1.1 in the Introduction, $\mathcal{V}_j$ denotes the spaces used by Sharon and Shkolnisky while V_j denotes a space in the multiresolution approximation for classical wavelets in Eq. (2.6). Note that the containment relations are reversed: $V_j \supset V_{j+1}$, but $\mathcal{V}_j \subset \mathcal{V}_{j+1}$.

spaces to generate complement spaces $\mathcal{W}_j$ such that $\mathcal{V}_j \perp \mathcal{W}_j$ and $\mathcal{V}_j \oplus \mathcal{W}_j = \mathcal{V}_{j+1}$. By induction, we have that

$$\mathcal{V} = \mathcal{V}_0 \oplus \mathcal{W}_0 \oplus \mathcal{W}_\oplus \cdots \oplus \mathcal{W}_{j_m - 1}. \tag{2.29}$$

The vectors whose spans define the spaces in (2.29) form an orthonormal basis such that (i) all but m basis vectors are orthogonal to the first m Laplacian eigenvectors of $L_{\text{rw}}(G)$ and (ii) all but $O(m)$ basis vectors have small support.

Another transform that utilizes a hierarchical tree is that of Rustamov [**63**], which is a generalization of the average-interpolating transform of Donoho et al. for manifold-valued data [**55**] to the setting of graphs. At each level $j \in [0, j_{\max})$ of the hierarchical tree, Rustamov computes the averages of the signal on each subgraph. He then uses the average values on a subgraph G_k^j and its neighbor subgraphs $\{G_{\tilde{k}}^j\}_{\tilde{k}}$ to impute the averages on the subgraphs of G_k^j, and he defines wavelet coefficients to be the difference between these imputed averages and the true averages.

Rustamov and Guibas developed a second transform which is based on the lifting scheme for classical wavelets (see, e.g., [**84**, **85**]), again making use of a hierarchical tree [**62**]. Starting with the generalized Haar transform, they use update and prediction operators, which are adaptively learned from a given set of signals, to design wavelets such that the expansion coefficients of a signal belonging to the same signal class become sparse. Jansen et al. have also designed a wavelet transform for signals on graphs that is based on the lifting scheme [**41**]. Starting with N scaling coefficients (which correspond to the canonical representation of the signal), they proceed by "lifting one coefficient at a time." Specifically, at each step they replace one scaling coefficient with a wavelet coefficient that they generate using update and prediction operators. As their method proceeds one coefficient at a time, their notion of scale is "more of a continuous concept and the fixed dyadic scales of the regular discrete wavelet transform no longer exist." [**41**]

Coifman et al. take a different approach, using the diffusion/random walk on a graph to build diffusion wavelets [**11**] and diffusion wavelet packets [**5**]. The underlying idea is that by taking dyadic powers of a diffusion operator U for which high powers have low numerical rank, they are able to coarsen the graph and construct a multiresolution approximation. At scale j, the scaling

functions are an orthonormal basis for the numerical range of U^{2^j}, obtained via a modified Gram-Schmidt procedure, and the wavelet functions are their orthogonal complements in the numerical range of $U^{2^{j-1}}$.

Having reviewed existing transforms and techniques for signals on graphs, we will now present our own transforms. Like many of the transforms covered in this subsection, we utilize a recursive partitioning of the graph. Hence, in the following chapter we set forth our criteria for the recursive partitioning tree and we describe one means of generating such a recursive partitioning.

CHAPTER 3

Recursive Graph Partitioning

We introduced the concept of a hierarchical tree in our discussion of Methods based on Vertex Transformations (§2.3.2). As a hierarchical tree serves as the foundation for both of our transforms, we now set forth our notation and requirements. We also present one means of generating a recursive partitioning that is suitable for our transforms.

We use j to denote the levels of the hierarchical tree, with $j = 0$ denoting the coarsest level and $j = j_{\max}$ denoting the finest level. We use K^j to denote the number of regions on level j, and we use $k \in [0, K^j)$ to index these regions. We define G_k^j to be the subgraph formed by restricting to the nodes in region k on level j and the edges between them, and we set $N_k^j := |V(G_k^j)|$, and as we did in §2.3.2, we define $V_k^j := V(G_k^j)$. We require that the only subgraph on level $j = 0$ is the entire graph, i.e., $K^0 = 1$, $G_0^0 = G$, and $N_0^0 = N$. We also require that each region on the finest level ($j = j_{\max}$) consists of a single vertex, and thus $N_k^{j_{\max}} = 1$ for $0 \leq k < K^{j_{\max}} = N$. While these two conditions could be relaxed[1], in this dissertation we maintain them as requirements because they standardize the notation and simplify the descriptions of the algorithms without negatively impacting the outcome. Figure 3.1 helps to illustrate these concepts, while Figure 3.2 shows a partial example on a subset of a dendritic tree.

[1]Technically, neither transform requires the coarsest level to contain the entire graph as its only region. Furthermore, while the GHWT requires that each node on level $j_{\max}$ contains only one vertex, the HGLET does not.

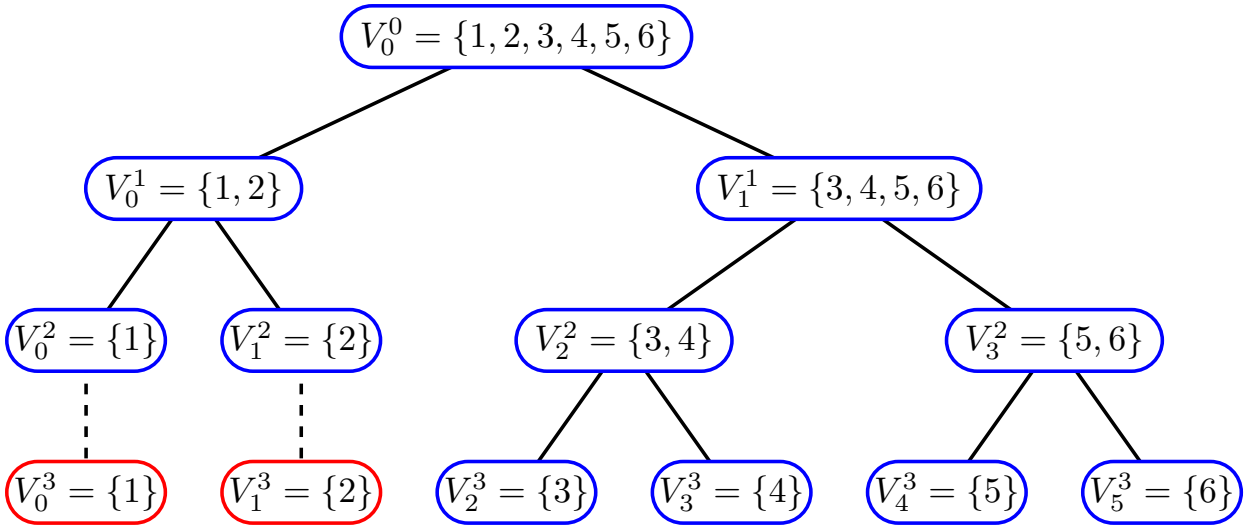
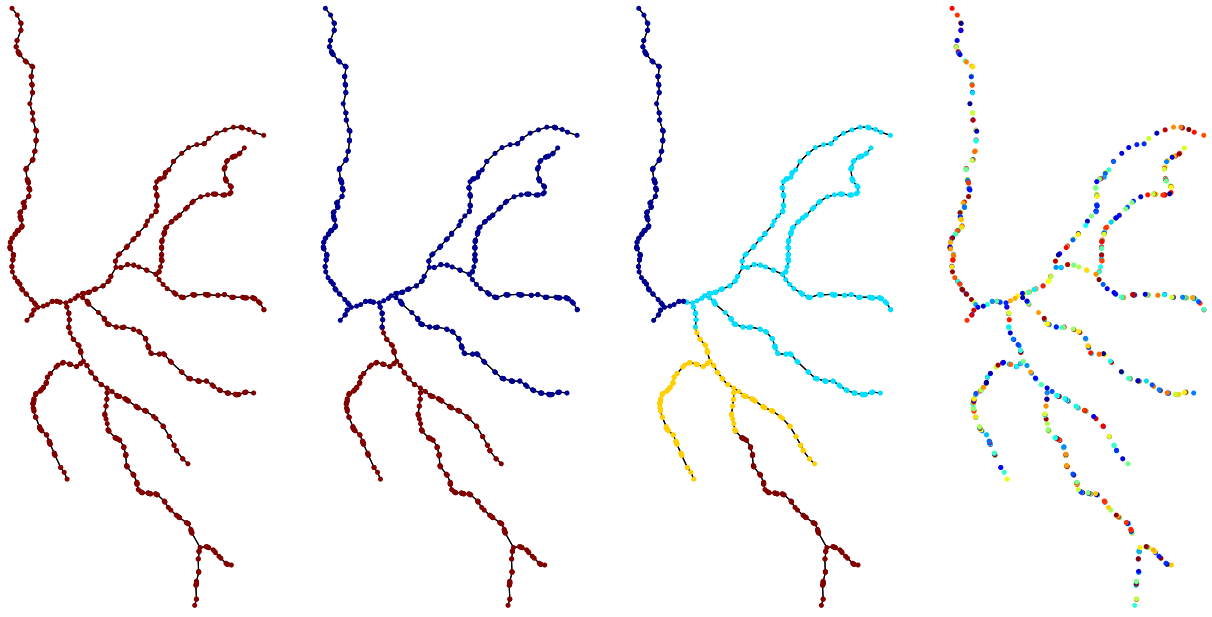


FIGURE 3.1. An example of a hierarchical tree for a graph with $N = 6$ nodes that conforms to our notation and requirements. The nodes encircled in red and connected by dashed lines are "copies" of singleton nodes. Whereas these nodes are not present in Fig. 2.14, we include them because our transforms require that all N nodes of the graph are present at each level j of the hierarchical tree.



(a) Level $j = 0$ (b) Level $j = 1$ (c) Level $j = 2$ (d) $j = 12 = j_{\max}$

FIGURE 3.2. A demonstration of recursive partitioning. In (a)-(c), colors correspond to different regions. In (d), each region is a single node, and as such all nodes are disconnected.

Both transforms require two additional conditions. First, all regions on each level of the tree are disjoint (i.e., if $k \neq \tilde{k}$ then $V(G_k^j) \cap V(G_{\tilde{k}}^j) = \emptyset$). Second, each region containing two or more nodes is partitioned. In addition, the GHWT specifically requires a recursive bisection, meaning that every region G_k^j on level j with more than one node is divided into exactly two regions on level $j+1$. While the HGLET is compatible with a region having an arbitrary number of subregions, for the sake of simplicity we maintain this recursive bisection criterion for the HGLET in this dissertation.

To summarize, the four conditions that we impose for the recursive partitioning are:

i. The coarsest level is the entire graph; that is, $G_0^0 = G$.
ii. At the finest level, each region is a single node; that is, $N_k^{j_{\max}} = 1$ for $0 \leq k < K^{j_{\max}} = N$.
iii. All regions on a given level are disjoint; that is, $V(G_k^j) \cap V(G_{\tilde{k}}^j) = \emptyset$ if $k \neq \tilde{k}$.
iv. Each region on level $j < j_{\max}$ containing two or more nodes is partitioned into exactly two regions on level $j+1$.

We now explain one method for generating a suitable recursive partitioning of a graph. Consider a connected subgraph G_k^j with two or more nodes. Form the associated Laplacian matrix $L(G_k^j)$; alternatively, we may use the random-walk normalized Laplacian $L_{\mathrm{rw}}(G_k^j)$. We compute the Fiedler vector $\boldsymbol{\phi}_1$, which is the first nonconstant eigenvector, and we partition the subgraph according to the signs of its entries. Justification of this approach comes from the fact that it yields an approximate minimizer of the bipartitioning criterion called the RatioCut (or the Normalized Cut) when L (or L_{rw}, respectively) is used [**29**, **88**], where

$$\begin{aligned} \mathrm{cut}(X, X^C) &:= \sum_{\substack{i \in X \\ j \in X^C}} W_{ij} \\ \mathrm{vol}(X) &:= \sum_{i \in X} d_i \\ \mathrm{RatioCut}(X, X^C) &:= \frac{\mathrm{cut}(X, X^C)}{|X|} + \frac{\mathrm{cut}(X, X^C)}{|X^C|} \\ \mathrm{NCut}(X, X^C) &:= \frac{\mathrm{cut}(X, X^C)}{\mathrm{vol}(X)} + \frac{\mathrm{cut}(X, X^C)}{\mathrm{vol}(X^C)}. \end{aligned} \tag{3.1}$$

Thus, we have partitioned G_k^j into two subgraphs, $G_{k'}^{j+1}$ and $G_{k'+1}^{j+1}$. (As the overall partitioning of the graph is not required to be perfectly balanced, nor can it be for non-dyadic signals, we have that $k \leq k' \leq 2k$).) Starting with the entire graph, G_0^0, we repeat this process until the graph is fully partitioned. Generating a recursive bipartitioning of a graph using Fiedler vectors is obviously not a novel idea – Simon discussed such a method in [**79**]. What is novel is our use of such a recursive bipartitioning to generate overcomplete dictionaries of orthonormal bases for analyzing signals on the graph. In fact, our transforms are compatible with hierarchical trees generated using different approaches, such as the diffuse interface model of Bertozzi and Flenner [**3**] or the local spectral method of Mahoney et al. [**44**]. This flexibility is certainly advantageous, since graph clustering and partitioning are quite active areas of research and new algorithms continue to be developed.

CHAPTER 4

Hierarchical Graph Laplacian Eigen Transform

4.1. Transform Overview

With this background in place, we now introduce the first of our two transforms: the Hierarchical Graph Laplacian Eigen Transform (HGLET) [**38**]. Using a recursive partitioning of the graph, the HGLET generates an overcomplete dictionary whose basis vectors' supports vary in size from a single node to the entire graph. We use $\boldsymbol{\phi}_{k,l}^{j}$ to denote the HGLET basis vectors[1], and we use $c_{k,l}^{j}$ to denote the corresponding expansion coefficients. As with the recursive partitioning, $j \in [0, j_{\max}]$ and $k \in [0, K^j)$ denote the level and region, respectively, to which a basis vector/coefficient corresponds. $l \in [0, N_k^j)$ indexes the vectors/coefficients from G_k^j. Thus, our indexing for the HGLET is similar to the wavelet packet notation in (2.11) and (2.12).

The HGLET basis vectors are formed by computing Laplacian eigenvectors on subgraphs G_k^j and extending them by zeros to the entire graph. These basis vectors may be the extended eigenvectors of: (a) the unnormalized Laplacian matrix L; (b) the random-walk normalized Laplacian L_{rw}; or (c) the symmetric normalized Laplacian L_{sym}.

Given a recursive partitioning of the graph, the HGLET algorithm proceeds as follows:

[1]To clarify, $\boldsymbol{\phi}_l$ refers to a Laplacian eigenvector and $\boldsymbol{\phi}_{k,l}^{j}$ refers to an HGLET basis vector.

Algorithm 2 HGLET (`HGLET_Analysis.m`)

1: **for** $j = j_{\max}, \ldots, 0$ **do**

2: **for** $k = 0, \ldots, K^j - 1$ **do**

3: If using the eigenvectors of the unnormalized Laplacian, construct $L(G_k^j)$. If using the eigenvectors of the random-walk or symmetric normalized Laplacian, construct $L_{\mathrm{sym}}(G_k^j)$. (NOTE: if the subgraph contains an isolated vertex, i.e., a vertex with degree 0, then we form the unnormalized Laplacian $L(G_k^j)$.)

4: Compute the eigenvectors $\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{N_k^j - 1}$ of the constructed Laplacian matrix using the singular value decomposition.

5: Compute the HGLET expansion coefficients $c_{k,l}^j$ such that

$$\boldsymbol{f} = \sum_{k=0}^{K^j-1} \sum_{l=0}^{N_k^j-1} c_{k,l}^j \boldsymbol{\phi}_{k,l}^j,$$

where the HGLET basis vector $\boldsymbol{\phi}_{k,l}^j$ is the extension by zeros of the lth eigenvector (i.e., $\boldsymbol{\phi}_l$, $\boldsymbol{\phi}_l^{\mathrm{rw}}$, or $\boldsymbol{\phi}_l^{\mathrm{sym}}$) to the entire graph. When using the eigenvectors of $L(G_k^j)$ or $L_{\mathrm{rw}}(G_k^j)$, this is done via

$$c_{k,l}^j := \left\langle \boldsymbol{f}|_{V(G_k^j)}, \boldsymbol{\phi}_l \right\rangle,$$

where the eigenvectors $\boldsymbol{\phi}_l$ correspond to the choice of Laplacian matrix. When using the eigenvectors of $L_{\mathrm{rw}}(G_k^j)$, we appeal to (2.13) and compute the coefficients via the weighted inner product

$$c_{k,l}^j := \left\langle \boldsymbol{f}|_{V(G_k^j)}, \boldsymbol{\phi}_l \right\rangle_{D(G_k^j)^{1/2}} = \boldsymbol{\phi}_l^* D(G_k^j)^{1/2} \boldsymbol{f}|_{V(G_k^j)}, \tag{4.1}$$

where $D(G_k^j)$ is the degree matrix and the eigenvectors $\boldsymbol{\phi}_l$ are those of $L_{\mathrm{sym}}(G_k^j)$.

6: **end for**

7: **end for**

The result of this algorithm is a set of highly redundant expansion coefficients. Figure 4.1 shows the HGLET basis vectors on an unweighted graph with 6 nodes, which serves as a simple example that we use to illustrate key features of our transform. The recursive partitioning for this graph is

illustrated in Figure 3.1. Using the hierarchical structure of the partitioning tree, we organize the basis vectors in a table. Indeed, observe that the structure of the table in Figure 4.1 is the same as the structure of the tree in Figure 3.1.

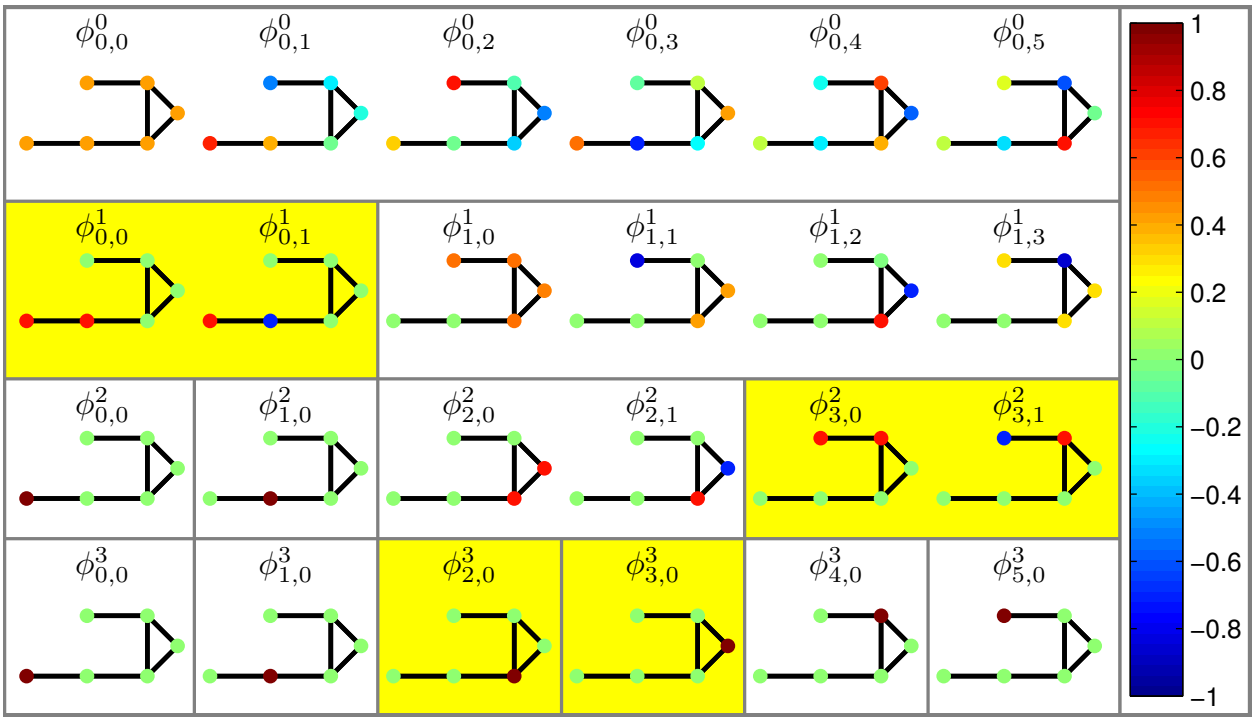


FIGURE 4.1. HGLET basis vectors on an unweighted graph with 6 nodes. Here, the graph was recursively partitioned using the Fiedler vector of the unnormalized Laplacian, and the HGLET basis vectors are the eigenvectors of the unnormalized Laplacian. The highlighted blocks are one example of an orthonormal basis that can be selected from the overcomplete dictionary of basis vectors. (The structure of the hierarchical partitioning tree is the same as in Figure 3.1.)

Keeping this structure in mind, it is clear that the HGLET dictionary contains a number of choosable orthonormal bases for signals on the vertices $V(G)$ of the graph. Indeed, from Step 4 of the HGLET algorithm we see that for a fixed j and k, the HGLET basis vectors $\{\boldsymbol{\phi}^j_{k,l}\}_{l\in[0,N^j_k)}$ form an orthonormal basis for signals on $V(G^j_k) \subseteq V(G)$. When using the eigenvectors of $L(G^j_k)$ or $L_{\text{sym}}(G^j_k)$, this basis is orthonormal with respect to the usual inner product, whereas using the eigenvectors of $L_{\text{rw}}(G^j_k)$ yields a basis that is orthonormal with respect to the degree-weighted inner product $\langle , \rangle_{D(G^j_k)}$. Furthermore, if G^j_k is partitioned into $G^{j+1}_{k'}$ and $G^{j+1}_{k'+1}$, then $V(G^j_k) = V(G^{j+1}_{k'}) \cup V(G^{j+1}_{k'+1})$

and

$$\begin{aligned} \operatorname{span}\{\boldsymbol{\phi}_{k,l}^{j}\}_l &= \operatorname{span}\{\boldsymbol{\phi}_{k',l}^{j+1}\}_l \oplus \operatorname{span}\{\boldsymbol{\phi}_{k'+1,l}^{j+1}\}_l \\ &= \operatorname{span}\left\{\boldsymbol{\phi}_{\tilde{k},l}^{\tilde{j}} \;\middle|\; V(G_{\tilde{k}}^{\tilde{j}}) \subseteq V(G_k^j)\right\}_l. \end{aligned} \tag{4.2}$$

Thus, we can construct an orthonormal basis consisting of bases on disjoint subgraphs from multiple levels of the HGLET dictionary. In terms of Figure 4.1, this means that we can construct an orthonormal basis by taking the basis vectors contained in any collection of blocks within the table that (a) do not overlap and (b) collectively span the full width of the table. The highlighted blocks illustrate one such basis.

Another result that follows from the underlying structure of the transform is that the HGLET can truly be viewed as a generalization of the block DCT from classical signals to signals on graphs. As discussed in §2.2, this is due to the fact that for an unweighted path graph P_N the eigenvectors of the unnormalized Laplacian $L(G)$ are the DCT-II basis vectors and the eigenvectors of $L_{\text{sym}}(G)$ are the DCT-I basis vectors. And as seen from (2.13), the eigenvectors of $L_{\text{rw}}(G)$ are related to those of $L_{\text{sym}}(G)$ by a factor of $D(G_k^j)^{-1/2}$. Thus, the HGLET on an unweighted path graph corresponds exactly to the block DCT. (That is, provided that the recursive partitioning scheme used for the HGLET is consistent with that of the block DCT.)

The computational cost of the HGLET is $O(N^3)$, which is due to computing the full set of eigenvectors of the $N \times N$ Laplacian matrix on level $j = 0$. Of course, when such a computation is prohibitively expensive, one could cut costs by performing the HGLET only on regions G_k^j with $N_k^j \leq N_{\max} < N$ nodes. With this modification in place, the cost would then be $O(N_{\max}^2 N)$, since there are approximately $N/N_{\max}$ regions with $N_{\max}$ or fewer nodes, and the cost for each of these regions is at most $O(N_{\max}^3)$.

Figure 4.2 shows some examples of HGLET basis vectors on a more elaborate graph, namely, the Minnesota road network (which we introduced in §2.3.2). These figures illustrate that for the dictionary of HGLET basis vectors $\{\boldsymbol{\phi}_{k,l}^j\}_{j,k,l}$, j provides a true index of scale (since basis vectors on level j have support size $\approx N/2^j$) while k serves as a location index. They also demonstrate how on a fixed level j, the supports of basis vectors from different regions are disjoint.

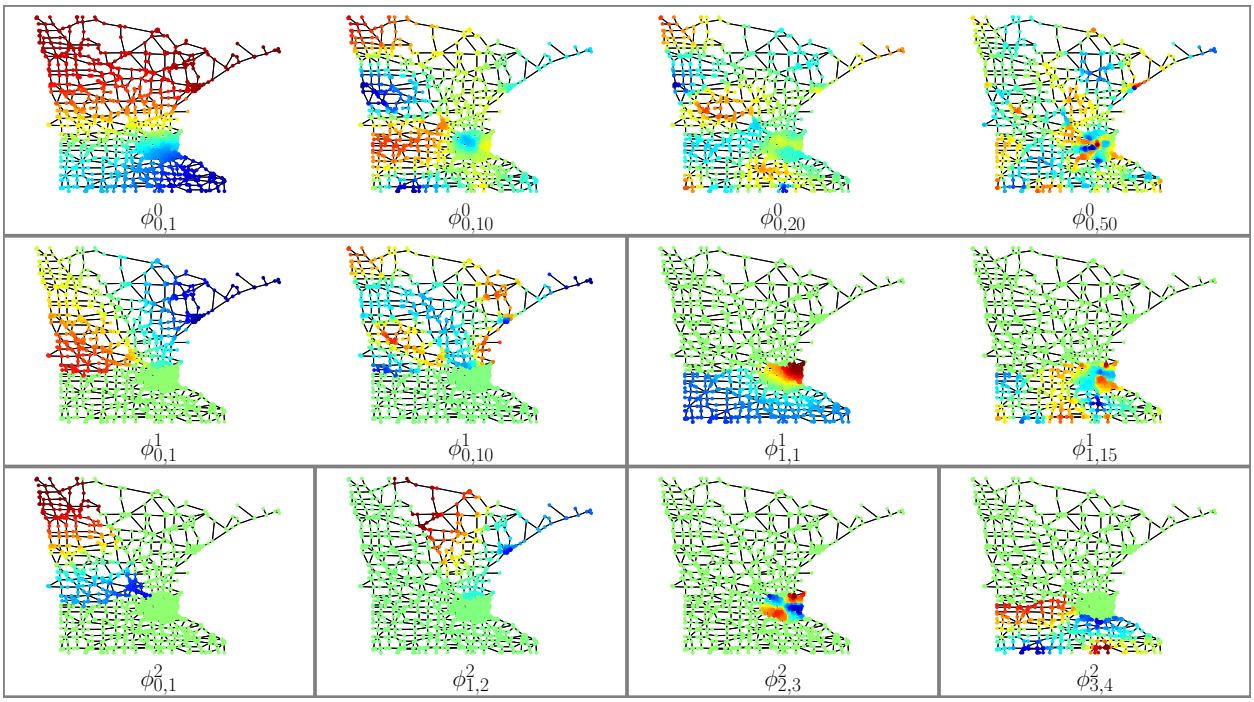


FIGURE 4.2. A subset of the HGLET basis vectors on the unweighted Minnesota road network ($N = 2640$ nodes and $M = 3302$ edges). The graph was recursively partitioned using the Fiedler vectors of the random-walk normalized Laplacians $L_{\mathrm{rw}}(G^j_k)$, and the basis vectors were generated using the unnormalized Laplacians $L(G^j_k)$. (Compare to the corresponding GHWT basis vectors in Figure 5.3.)

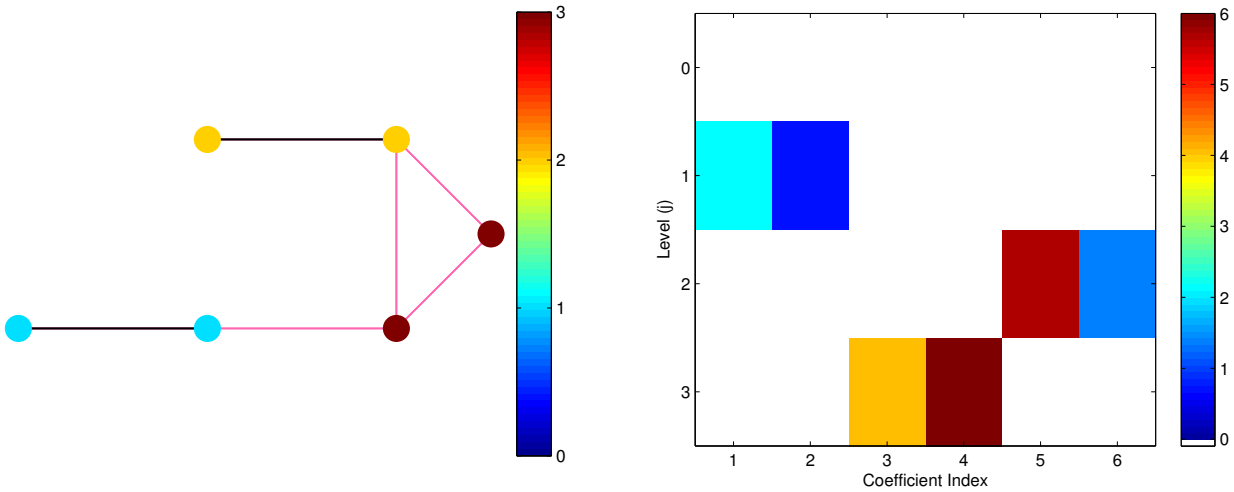
4.2. Basis Specification and Visualization

Of course, it is important to have a means of specifying bases from within the HGLET dictionary. Appealing to the *levels list* description method [**90**, §9.2] from classical wavelets, we succinctly describe a basis by recording the level j of each block of coefficients. As an example, the levels list description of the highlighted basis in Figure 4.1 is $(1, 3, 3, 2)$.

It is also desireable to be able to visualize a basis, and for the HGLET we can do this in two ways, which we illustrate in Figure 4.3 for the highlighted basis in Figure 4.1. First, for graphs in 1, 2, or 3 dimensions we can visualize the regions of the basis, as seen in Figure 4.3a. Whereas in Figures 4.1 and 4.2 the colors indicate the values of the function at the nodes, here the color of each node indicates the level $j \in [0, j_{\max}]$ of the subgraph whose associated basis vectors describe it. For example, the two nodes on the top are captured by basis vectors $\boldsymbol{\phi}^2_{3,0}$ and $\boldsymbol{\phi}^2_{3,1}$, so therefore they

are assigned the value 2. To further illustrate the regions in the basis, we use black to display intra-region edges (i.e., those which are left intact in the basis) and we use pink to display inter-region edges (i.e., those which are cut in the basis).

The second means of displaying a basis is to display the coefficients in a table, as seen in Figure 4.3b. The order of the coefficients corresponds to the order of the nodes on the bottom (finest) level of the hierarchical tree, and the rows of the table correspond to levels of the tree. For example, the two blue blocks on the left side of the table correspond to the two nodes on the bottom left of the graph. Thus, the locations of the coefficients in this table are the same as the locations of the highlighted basis vectors in Figure 4.1. The colors of the blocks correspond to the magnitudes of the coefficients; portions of the table which are white are not part of the basis.



(a) (b)

FIGURE 4.3. Visualizations of the highlighted basis with levels list description $(1, 3, 3, 2)$ from Figure 4.1. (a) A visualization of the regions whose corresponding basis vectors comprise the basis, with the colors of the nodes indicating the levels of the regions. (b) A display of the basis' expansion coefficients. Rows of the table indicate the level indices of the coefficients, and colors correspond to their magnitudes. The signal analyzed for this example is simply $(1, 2, 3, 4, 5, 6)^{\mathsf{T}}$.

CHAPTER 5

Generalized Haar-Walsh Transform

5.1. Transform Overview

We now present our second transform, the Generalized Haar-Walsh Transform (GHWT) [**37**]. Like the HGLET, the GHWT uses a recursive partitioning of the graph to generate an overcomplete dictionary, but in this case the basis vectors are piecewise constant on their support. We use $\boldsymbol{\psi}_{k,l}^{j}$ and $d_{k,l}^{j}$ to denote the GHWT basis vectors and expansion coefficients, respectively. As with the HGLET, $j \in [0, j_{\max}]$ and $k \in [0, K^j)$ denote level and region. In the case of the GHWT, we refer to l as the basis vector's/coefficient's *tag*, and it assumes N_k^j distinct values within the range $[0, 2^{j_{\max}-j})$. The tag is an integer which, when expressed in binary, specifies the sequence of low-frequency (averaging) and high-frequency (differencing) operations that were used to generate it. For example, $l = 6$ written in binary is 110, which means that the basis vector/expansion coefficient was produced by two differencing operations followed by an averaging operation. Within a given region k on level j, the tags are never duplicated, and thus they serve as unique identifiers for the basis vectors/coefficients within the region. We refer to coefficients with tag $l = 0$ as *scaling coefficients*, those with tag $l = 1$ as *Haar-like coefficients*, and those with tag $l \geq 2$ as *Walsh-like coefficients*.

The GHWT proceeds by defining an orthonormal basis on level $j_{\max}$ and obtaining the corresponding expansion coefficients. As each region on this level contains a single node, the canonical basis is the logical choice: $\boldsymbol{\psi}_{k,0}^{j_{\max}} := \mathbf{1}_{V(G_k^{j_{\max}})}$, where $k \in [0, N)$ and $|V(G_k^{j_{\max}})| = 1$. The expansion coefficients $d_{k,0}^{j_{\max}}$ are then obtained by simply reordering the input signal, $\boldsymbol{f}$. From here the algorithm proceeds recursively, using the expansion coefficients on level j to compute those on level $j-1$.

In our description of the algorithm, for a region G_k^j on level $j < j_{\max}$ with two or more nodes we denote its subregions by $G_{k'}^{j+1}$ and $G_{k'+1}^{j+1}$, as we have done in Chapters 3 and 4. If G_k^j has only

one node then it obviously cannot be partitioned, and we denote by $G_{k'}^{j+1}$ its duplicate on level $j+1$. Although we do not explicitly compute the GHWT basis vectors in Algorithm 3, they are generated in this same manner, using Eqs. (5.1)-(5.4), but with $\boldsymbol{\psi}$ in place of d. Given a recursive partitioning of the graph, the GHWT proceeds as follows:

Algorithm 3 GHWT (`GHWT_Analysis.m`)

1: Start on level $j_{\max}$. Define an orthonormal basis $\{\boldsymbol{\psi}_{k,0}^{j_{\max}}\}_{k=0}^{N-1}$ and obtain the corresponding expansion coefficients $\{d_{k,0}^{j_{\max}}\}_{k=0}^{N-1}$, as discussed above.

2: **for** $j = j_{\max}, \ldots, 1$ **do**

3: **for** $k = 0, \ldots, K^{j-1} - 1$ **do**

4: Compute the *scaling coefficient* on G_k^{j-1} as

$$d_{k,0}^{j-1} := \begin{cases} d_{k',0}^{j} & \text{if } N_k^{j-1} = 1 \\ \frac{\sqrt{N_{k'}^j} d_{k',0}^j + \sqrt{N_{k'+1}^j} d_{k'+1,0}^j}{\sqrt{N_k^{j-1}}} & \text{if } N_k^{j-1} > 1 \end{cases} \tag{5.1}$$

5: **if** $N_k^{j-1} > 1$, **then**

6: Compute the *Haar-like coefficient* as

$$d_{k,1}^{j-1} := \frac{\sqrt{N_{k'+1}^j} d_{k',0}^j - \sqrt{N_{k'}^j} d_{k'+1,0}^j}{\sqrt{N_k^{j-1}}}. \tag{5.2}$$

7: **end if**

8: **if** $N_k^{j-1} > 2$, **then**

9: Compute the *Walsh-like coefficients* as follows.

10: **for** $l = 1, \ldots, 2^{j_{\max}-j} - 1$ **do**

11: **if** neither subregion has a coefficient with tag l, **then** do nothing.

12: **if** (without loss of generality) subregion k' has a coefficient with tag l but subregion $k'+1$ does not, **then** set

$$d_{k,2l}^{j-1} := d_{k',l}^{j}. \tag{5.3}$$

13: **if** both subregions have coefficients with tag l, **then** compute

$$\begin{aligned} d_{k,2l}^{j-1} &:= \left(d_{k',l}^{j} + d_{k'+1,l}^{j}\right) \Big/ \sqrt{2} \\ d_{k,2l+1}^{j-1} &:= \left(d_{k',l}^{j} - d_{k'+1,l}^{j}\right) \Big/ \sqrt{2}. \end{aligned} \tag{5.4}$$

14: **end for**

15: **end if**

16: **end for**

17: **end for**

Figure 5.1 shows the GHWT basis vectors on a *weighted* path graph of length 6, with the highlighted blocks illustrating one possible orthonormal basis selection. We reduce the weight of the edge between the second and third nodes, and as a result the recursive partitioning tree for this path graph is the same as the one shown in Figure 3.1. (This partitioning tree also corresponds to that of the graph in Figure 4.1 for which the HGLET basis vectors are shown.) As with the HGLET, the structure of this table of GHWT basis vectors is the same as the structure of the partitioning tree. This explains why the tables of HGLET and GHWT basis vectors in Figures 4.1 and 5.1 have the same structure, even though the graphs and basis vectors clearly differ.

As with the HGLET, we can form a basis for signals on $V(G)$ by taking the union of bases on disjoint subgraphs from various levels of the recursive partitioning tree whose union is the entire graph G. This is due to the fact that for a subgraph G_k^j that is partitioned into subgraphs $G_{k'}^{j+1}$ and $G_{k'+1}^{j+1}$,

$$\begin{aligned} \mathrm{span}\{\boldsymbol{\psi}_{k,l}^j\}_l &= \mathrm{span}\{\boldsymbol{\psi}_{k',l}^{j+1}\}_l \oplus \mathrm{span}\{\boldsymbol{\psi}_{k'+1,l}^{j+1}\}_l, \\ &= \mathrm{span}\left\{\boldsymbol{\psi}_{\tilde{k},l}^{\tilde{j}} \mid V(G_{\tilde{k}}^{\tilde{j}}) \subseteq V(G_k^j)\right\}_l. \end{aligned} \tag{5.5}$$

This corresponds exactly to relation (4.2) for the HGLET. In terms of Figure 5.1, this means that we can form an orthonormal basis from the GHWT dictionary by taking the basis vectors from non-overlapping blocks that span the full width of the table, just as we did with the HGLET; the highlighted blocks are one such possibility. When the GHWT basis vectors are grouped by region, as in Figure 5.1, we call the result the *GHWT coarse-to-fine dictionary*.

For both the HGLET and GHWT, we have used the relationships in Eqs. (4.2) and (5.5), respectively, to group the basis vectors by their location index k. For the GHWT, we have a similar relationship involving the tags of the basis vectors: for a fixed $j > 0$ and l,

$$\begin{aligned} \mathrm{span}\{\boldsymbol{\psi}_{k,l}^j\}_k &= \mathrm{span}\{\boldsymbol{\psi}_{k,2l}^{j-1}\}_k \oplus \mathrm{span}\{\boldsymbol{\psi}_{k,2l+1}^{j-1}\}_k \\ &= \mathrm{span}\left\{\boldsymbol{\psi}_{k,\tilde{l}}^{\tilde{j}} \mid \tilde{j} \le j \text{ and } \lfloor \tilde{l}/2^{(j-\tilde{j})} \rfloor = l\right\}_k. \end{aligned} \tag{5.6}$$

By exploiting this relationship, we can organize the basis vectors in a new manner in which they are grouped by their tags, as seen in Figure 5.2, and we call this dictionary the *GHWT fine-to-coarse dictionary*. The blocks highlighted in yellow and green show two examples of orthonormal

bases. The highlighted green blocks form the Haar basis for signals on this graph, which has been studied in [**10**, **32**, **43**, **49**]. This is an important feature of the GHWT: the generalized Haar basis is contained within the structure of the the fine-to-coarse dictionary. Meanwhile, the highlighted yellow blocks are an example of a novel orthonormal basis that may be chosen from the fine-to-coarse dictionary. Note that neither of these two bases can be selected from the structure of the coarse-to-fine dictionary in Figure 5.1, and vice versa.

In the process of forming the fine-to-coarse dictionary, not only have we rearranged the basis vectors, but we have also changed the underlying structure of the dictionary. Indeed, note that the structures of the tables in Figures 5.1 and 5.2 differ. Whereas the coarse-to-fine dictionary inherited its structure from the partitioning tree, the fine-to-coarse dictionary's structure arises from the relationship (5.6) among the tags of its basis vectors/expansion coefficients. The GHWT fine-to-coarse dictionary is similar to a wavelet packet dictionary, such as the Haar-Walsh wavelet packet dictionary in Figure 2.4, in that the most localized functions are at the top, the most global are at the bottom, and it is structured according to the l indices of the functions. In fact, for a dyadic path graph equipped with a dyadic recursive partitioning, the GHWT corresponds exactly to the classical Walsh-Hadamard transform (with Paley-ordering; see page 21). Moreover, assuming that we already have a recursive partitioning of the graph that has $O(\log N)$ levels, the computational cost of the GHWT is $O(N \log N)$, which is the same as the cost of the Walsh-Hadamard transform.

The tag not only provides us with another means of grouping the basis vectors, it also imparts upon the basis vectors a general notion of frequency (or the so-called 'sequency' [**90**, Ch. 7]). As seen in (5.1)-(5.4), the tag reflects the sequence of averaging and differencing operations that were used to yield the basis vector. Generally speaking, larger l values indicate more oscillation, with exceptions occurring when imbalances in the partitioning tree necessitate the use of (5.3), as opposed to (5.4).

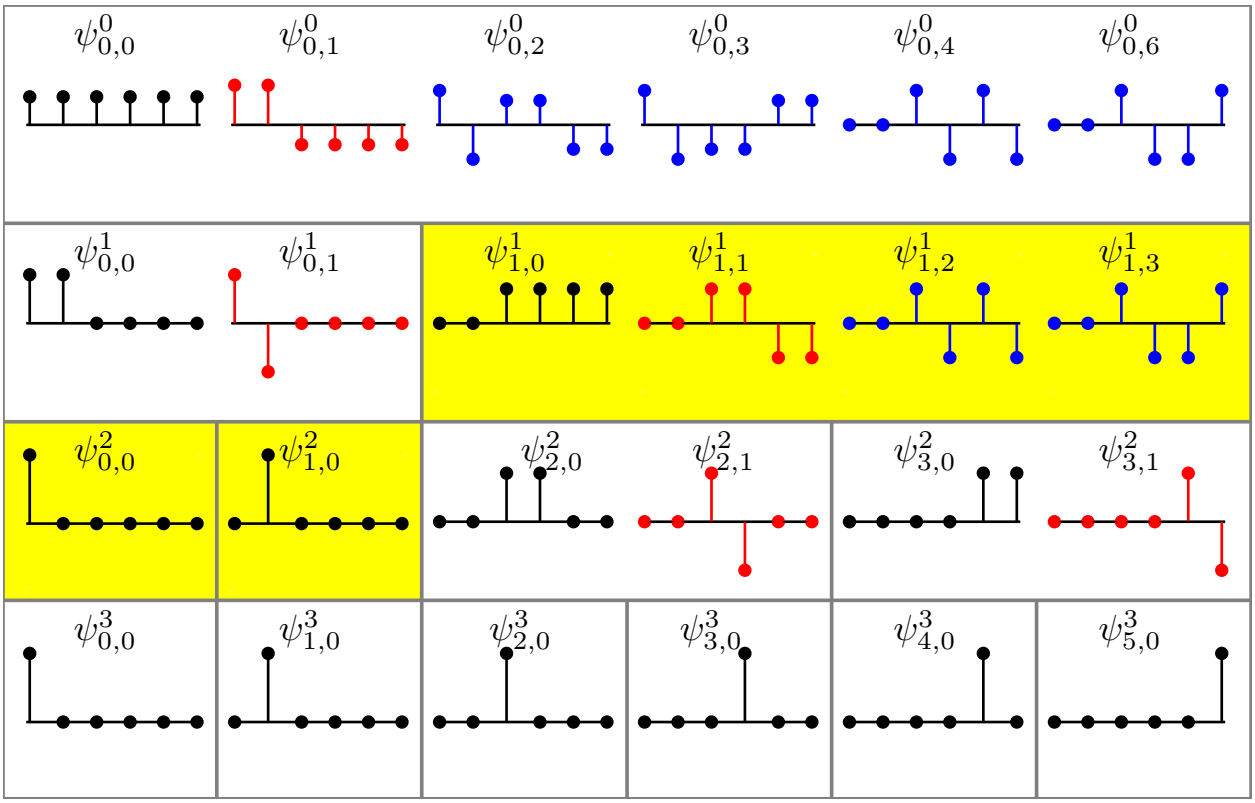


FIGURE 5.1. GHWT basis vectors on a weighted path graph of length 6. The weight between nodes 2 and 3 is $1/10$, whereas the other weights are 1, which explains why the first partition occurs off-center. The graph was recursively partitioned using the Fiedler vector of the unnormalized Laplacian. (The structure of the hierarchical partitioning tree is the same as in Figure 3.1.) Here, the basis vectors are grouped by region. Since the coarsest level is at the top and the finest level is at the bottom, we refer to this as the coarse-to-fine dictionary. The highlighted blocks illustrate an orthonormal basis which can be selected from this overcomplete dictionary, and its levels list description is $(2, 2, 1)$. Comparing this to the HGLET dictionary in Figure 4.1, we see that the structure of the recursive partitioning is the same, but the basis vectors differ. Also, note that here we have $\boldsymbol{\psi}^0_{0,6}$ in place of $\boldsymbol{\phi}^0_{0,5}$. This is because the l indices of HGLET basis vectors are $0, 1, \ldots, N^j_k - 1$, whereas the l indices of GHWT basis vectors are a subset of $[0, 2^{j_{\max}-j})$.

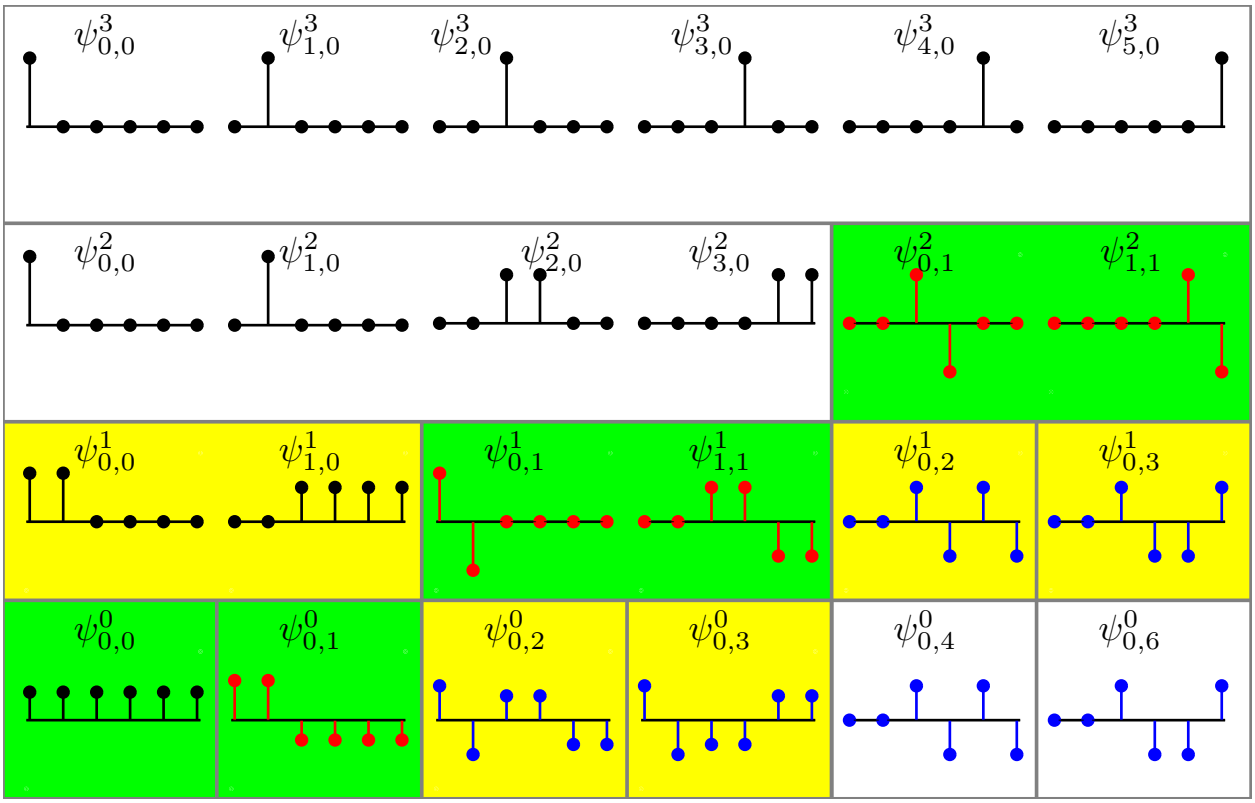


FIGURE 5.2. GHWT basis vectors on the same weighted path graph of length 6 as in Figure 5.1. Here, the basis vectors are grouped by tag, and we refer to this as the fine-to-coarse dictionary. The highlighted green blocks form the Haar basis for signals on this graph, while the highlighted yellow blocks are an example of yet another orthonormal basis that may be chosen from the fine-to-coarse dictionary. The levels list descriptions of the yellow and green highlighted bases are $(1, 0, 0, 1, 1)$ and $(0, 0, 1, 2)$, respectively. Comparing this grouping of basis vectors to that in Figure 5.1, we see that the structures of the dictionaries differ. Neither of these two highlighted bases can be selected from the structure of the coarse-to-fine dictionary, and vice versa. Indeed, note that neither of these levels list descriptions are valid basis specifications in the coarse-to-fine dictionary, nor is the levels list description from Figure 5.1 a valid basis specification here.

Figure 5.3 illustrates some GHWT basis vectors on the Minnesota road network. The structure of the table corresponds to the partitioning tree, and the figure represents a subset of the top three levels of the GHWT coarse-to-fine dictionary. The recursive partitioning tree is the same as was used for Figure 4.2, and thus the supports of the HGLET and GHWT basis vectors coincide. Moreover, it follows that the structures of the HGLET dictionary and the GHWT coarse-to-fine dictionary

are identical. We make use of this compatibility in Chapter 6 to select a *hybrid* orthonormal basis containing both HGLET and GHWT basis vectors.

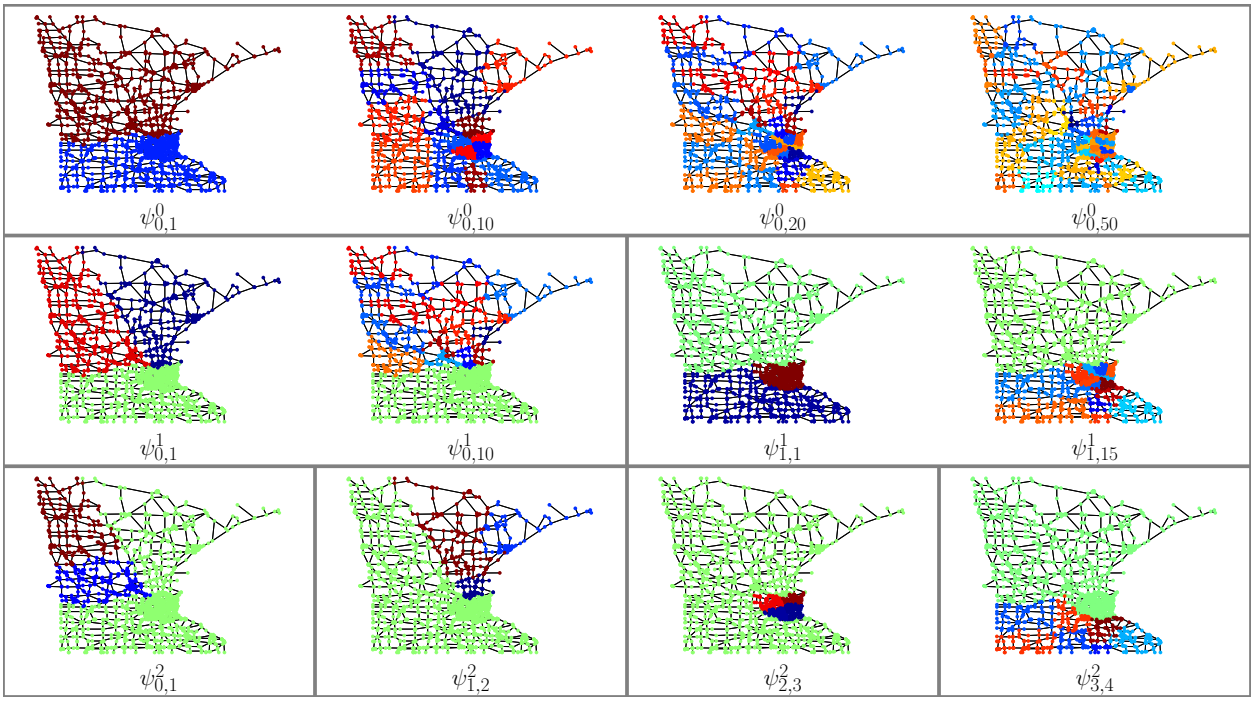
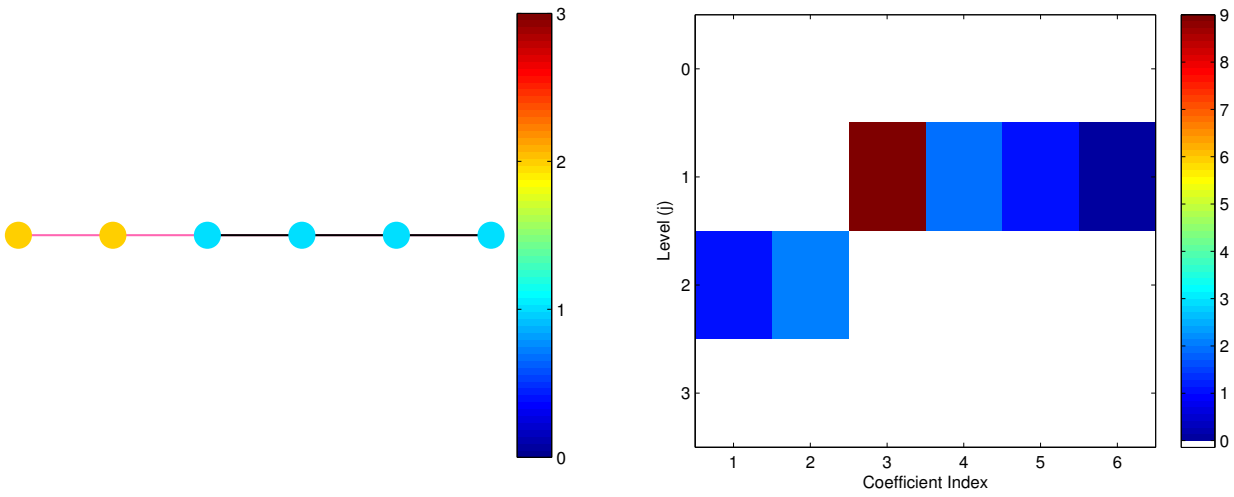


FIGURE 5.3. A subset of the GHWT basis vectors on the unweighted Minnesota road network ($N = 2640$ nodes and $M = 3302$ edges). The graph was recursively partitioned using the Fiedler vectors of the random-walk normalized Laplacians $L_{\mathrm{rw}}(G_k^j)$. (Compare to the corresponding HGLET basis vectors in Figure 4.2.)

5.2. Basis Specification and Visualization

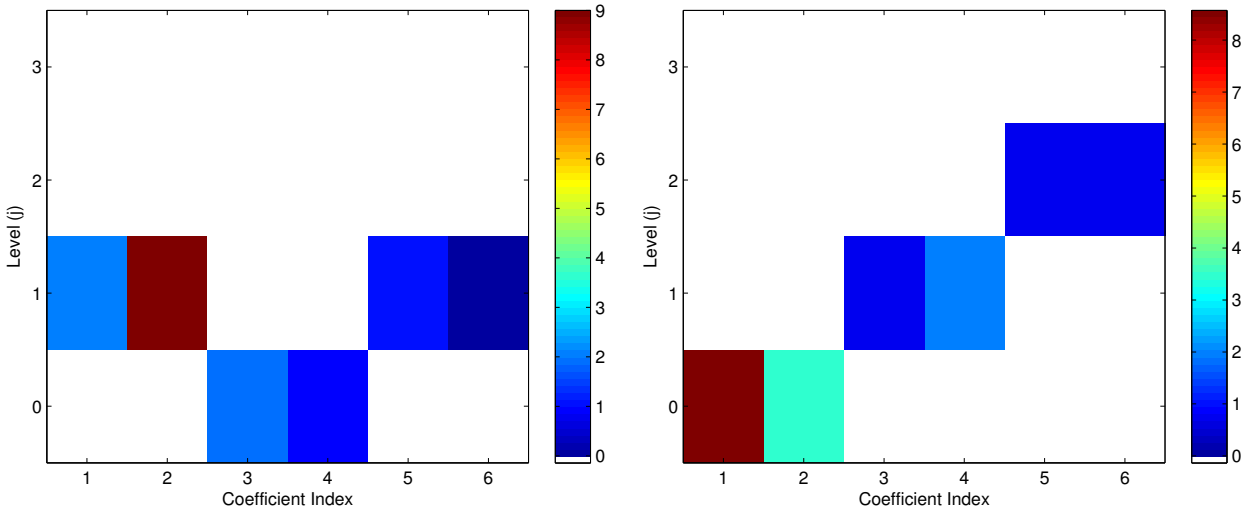
As with the HGLET, we need to be able to specify and visualize GHWT bases. Conveniently, we can continue to use the levels list description method. For the coarse-to-fine dictionary, this works exactly in the same manner as with the HGLET dictionary. Moreover, we can visualize a basis using the same two techniques that we used for the HGLET, as illustrated in Figure 5.4.



(a) (b)

FIGURE 5.4. Visualizations of the highlighted basis with levels list description $(2, 2, 1)$ from Figure 5.1. (a) A visualization of the regions whose corresponding basis vectors comprise the basis, with the colors of the nodes indicating the levels of the regions. (b) A display of the basis' expansion coefficients. Rows of the table indicate the level indices of the coefficients, and colors correspond to their magnitudes. The signal analyzed for this example is simply $(1, 2, 3, 4, 5, 6)^{\mathsf{T}}$.

For the fine-to-coarse dictionary the order of the levels is reversed, but the levels list description method is still applicable. However, while the coarse-to-fine dictionary is grouped by location index k, the fine-to-coarse dictionary is grouped by tag l, and as a result we cannot utilize the illustration of the graph regions (i.e., Figure 5.4a) for bases from the fine-to-coarse dictionary. Fortunately, we can still display tables of coefficients, as shown in Figures 5.5a and 5.5b for the highlighted yellow and green bases from Figure 5.2, respectively. As before, rows of the table indicate the level indices of the coefficients, colors correspond to their magnitudes, and portions of the table shown in white are not part of the basis.



(a) (b)

FIGURE 5.5. Displays of the expansion coefficients for the (a) yellow and (b) green highlighted bases from Figure 5.2. The levels list descriptions are $(1, 0, 0, 1, 1)$ and $(0, 0, 1, 2)$, respectively. The signal analyzed for this example is simply $(1, 2, 3, 4, 5, 6)^{\mathsf{T}}$.

CHAPTER 6

Best Basis Algorithms

Both the HGLET and the GHWT produce an overcomplete set of expansion coefficients for a given input signal; in the case of the GHWT, we may organize these expansion coefficients in either the coarse-to-fine arrangement or the fine-to-coarse arrangement. From here, the next logical step is to choose a basis. For this purpose, we adapt the best basis algorithm of Coifman and Wickerhauser [**14**] to our setting of multiscale transforms for signals on graphs. As with the classical best basis algorithm, we require a cost functional $\mathcal{J}$ that is suitable for the task at hand; e.g., a cost functional that favors sparse representations of input signals. A basis with a smaller cost is considered to be a better basis, and the basis from the dictionary/dictionaries considered that minimizes the cost functional is accordingly deemed the best basis.

We first consider the best basis algorithm for the HGLET. To improve clarity and simplify notation, we view the matrix of HGLET expansion coefficients $\{c_{k,l}^j\}_{j,k,l}$ as being organized in table format, just as we display the HGLET coefficients in Figure 4.3b; these coefficients may correspond to the unnormalized Laplacians $L(G_k^j)$, the random-walk normalized Laplacians $L_{\text{rw}}(G_k^j)$, or the symmetric normalized Laplacians $L_{\text{sym}}(G_k^j)$. The HGLET best basis algorithm makes use of Eq. (4.2), which tells us that each block of basis vectors in this table spans the same space as the union of all the vectors in blocks directly beneath it. Letting $\{b_i\}_{i\in[0,N)}$ denote the best basis expansion coefficients, the algorithm proceeds as follows:

Algorithm 4 HGLET Best Basis Algorithm (`HGLET_BestBasis.m`)

0: For a signal $\boldsymbol{f}$ on the graph, use the HGLET to generate the matrix of expansion coefficients, $c_{k,l}^{j}$. The HGLET L, L_{rw}, or L_{sym} variation may be used here.

1: Initialize the best basis as the level $j = j_{\max}$ basis; this is the bottom level of the table. That is, initialize $\{b_i\}_{i\in[0,N)} := \{c_{k,0}^{j_{\max}}\}_{k\in[0,N)}$.

2: **for** $j = j_{\max} - 1, \ldots, 0$ **do**

3: **for** $k = 0, \ldots, K^j - 1$ **do**

4: Let $I_{k,\bullet}^{j} \subseteq [0, N)$ denote the indices of the best basis coefficients whose corresponding basis vectors span the same space as $\{\boldsymbol{\phi}_{k,l}^{j}\}_l$. That is, $I_{k,\bullet}^{j}$ are the indices of the current best basis coefficients that fall beneath the coefficients $\{c_{k,l}^{j}\}_l$ in the table of coefficients.

5: **if** $\mathcal{J}\left(\{b_i\}_{I_{k,\bullet}^{j}}\right) \geq \mathcal{J}\left(\{c_{k,l}^{j}\}_l\right)$, **then** set $\{b_i\}_{I_{k,\bullet}^{j}} := \{c_{k,l}^{j}\}_l$.

6: **end for**

7: **end for**

The result is the set of expansion coefficients $\{b_i\}$, which correspond to the best basis for describing the signal $\boldsymbol{f}$ on the graph G. If using either unnormalized or symmetric normalized Laplacians, this basis is orthonormal.

With minor modifications, this same algorithm may be used for selecting a best basis from the GHWT expansion coefficients. In this case, we will obtain both a set of coarse-to-fine best basis coefficients (denoted by $\{b_i^{\text{c}}\}_{i\in[0,N)}$) and a set of fine-to-coarse best basis coefficients (denoted by $\{b_i^{\text{f}}\}_{i\in[0,N)}$), and we compare these two to determine which is the overall best basis. Again, for both the coarse-to-fine and fine-to-coarse dictionaries, a block of basis vectors spans the same space as all blocks directly beneath it, as illustrated in Figures 5.1 and 5.2. The algorithm is as follows:

Algorithm 5 GHWT Best Basis Algorithm (`GHWT_BestBasis.m`)

For a signal $\boldsymbol{f}$ on the graph, use the GHWT to generate the matrix of expansion coefficients, $d_{k,l}^j$.

With the coefficients organized in their coarse-to-fine arrangment, find the coarse-to-fine best basis coefficients $\{b_i^{\mathrm{c}}\}_{i\in[0,N)}$ by using Algorithm 4 with $c_{k,l}^j$ and $\boldsymbol{\phi}_{k,l}^j$ replaced by $d_{k,l}^j$ and $\boldsymbol{\psi}_{k,l}^j$, respectively.

Arrange the coefficients in the fine-to-coarse manner. Find the fine-to-coarse best basis coefficients $\{b_i^{\mathrm{f}}\}_{i\in[0,N)}$ as follows.

    Initialize the level $j = 0$ basis to be the best basis; this is the bottom level of the fine-to-coarse table. Accordingly, initialize $\{b_i^{\mathrm{f}}\}_{i\in[0,N)} := \{d_{0,l}^0\}_l$.

    for $j = 1, \ldots, j_{\max}$ **do**

        for $l = 0, \ldots, 2^{j_{\max}-j} - 1$ **do**

            Let $I_{\bullet,l}^j \subseteq [0, N)$ denote the indices of the best basis coefficients whose corresponding basis vectors span the same space as $\{\boldsymbol{\psi}_{k,l}^j\}_k$. That is, $I_{\bullet,l}^j$ are the indices of the current best basis coefficients which fall beneath the coefficients $\{d_{k,l}^j\}_k$ in the fine-to-coarse table of coefficients.

            if $\mathcal{J}\left(\{b_i^{\mathrm{f}}\}_{I_{\bullet,l}^j}\right) \geq \mathcal{J}\left(\{d_{k,l}^j\}_k\right)$, **then** set $\{b_i^{\mathrm{f}}\}_{I_{\bullet,l}^j} := \{d_{k,l}^j\}_k$.

        end for

    end for

Compare the coarse-to-fine best basis and the fine-to-coarse best basis.

if $\mathcal{J}\left(\{b_i^{\mathrm{c}}\}_i\right) > \mathcal{J}\left(\{b_i^{\mathrm{f}}\}_i\right)$, **then**

    The fine-to-coarse best basis is the overall best basis. Set $\{b_i\}_{i\in[0,N)} := \{b_i^{\mathrm{f}}\}_{i\in[0,N)}$.

else

    The coarse-to-fine best basis is the overall best basis. Set $\{b_i\}_{i\in[0,N)} := \{b_i^{\mathrm{c}}\}_{i\in[0,N)}$.

end if

For the signal $\boldsymbol{f}$ on G, Algorithm 5 returns the overall best basis expansion coefficients. These coefficients originate from either the coarse-to-fine or the fine-to-coarse dictionary, and the basis to which they correspond is orthonormal.

As we noted in Chapter 5, both the HGLET dictionary and the GHWT coarse-to-fine dictionary consist of basis vectors grouped by region, and as a result these dictionaries have the same

structure. Furthermore, we have three variations of the HGLET, corresponding to the choice of Laplacian matrix used: L, L_{rw}, and L_{sym}. Thus, we can generate four sets of overcomplete expansion coefficients that correspond to the same recursive partitioning and therefore have the same hierarchical structure. We can exploit this fact to devise a *hybrid* best basis algorithm which allows for disjoint regions to be described by different transforms. The idea is simple: we consider four sets of coefficients corresponding to the three HGLET variations and the GHWT coarse-to-fine dictionary, and at each subgraph G_k^j in the hierarchical tree we compare the corresponding expansion coefficients for all four transforms to the current best basis coefficients. Of course, for each block of best basis coefficients we will need to keep track of their corresponding transform.

Let $\{c_{k,l}^j\}$, $\{\tilde{c}_{k,l}^j\}$, and $\{\hat{c}_{k,l}^j\}$ denote the expansion coefficients corresponding to the HGLET using the eigenvectors of L, L_{rw}, and L_{sym}, respectively. As before, $\{d_{k,l}^j\}$ denotes the GHWT expansion coefficients. The algorithm is very similar to the HGLET Best Basis Algorithm (Algorithm 4), with the significant difference occurring at Step 5.

Algorithm 6 HGLET & GHWT Hybrid Best Basis Algorithm (`HGLET_GHWT_BestBasis.m`)

0: For a signal $\boldsymbol{f}$ on the graph, use the HGLET with each of the three Laplacians and the GHWT to generate the four matrices of expansion coefficients: $c_{k,l}^j$, $\tilde{c}_{k,l}^j$, $\hat{c}_{k,l}^j$, and $d_{k,l}^j$.

1: Initialize the best basis as the level $j = j_{\max}$ basis for the HGLET with the unnormalized Laplacian L. That is, initialize $\{b_i\}_{i\in[0,N)} := \{c_{k,0}^{j_{\max}}\}_{k\in[0,N)}$.

2: **for** $j = j_{\max} - 1, \ldots, 0$ **do**

3: **for** $k = 0, \ldots, K^j - 1$ **do**

4: Let $I_{k,\bullet}^j \subseteq [0, N)$ denote the indices of the best basis coefficients whose corresponding basis vectors span the same space as $\{\boldsymbol{\phi}_{k,l}^j\}_l$. That is, $I_{k,\bullet}^j$ are the indices of the current best basis coefficients which fall beneath the coefficients $\{c_{k,l}^j\}_l$ in the table of coefficients.

5: **if** $\mathcal{J}\left(\{b_i\}_{I_{k,\bullet}^j}\right) \geq \min\left\{\mathcal{J}\left(\{c_{k,l}^j\}_l\right), \mathcal{J}\left(\{\tilde{c}_{k,l}^j\}_l\right), \mathcal{J}\left(\{\hat{c}_{k,l}^j\}_l\right), \mathcal{J}\left(\{d_{k,l}^j\}_l\right)\right\}$, **then** set $\{b_i\}_{I_{k,\bullet}^j}$ to be the expansion coefficients of the minimum cost and record the transform from which the coefficients came.

6: **end for**

7: **end for**

The resulting best basis coefficients correspond to basis vectors that are supported on disjoint sets of vertices $V(G_k^j)$ whose union is the set $V(G)$ of all vertices. We also record the particular transform used to generate each region's coefficients. To reconstruct the signal using this hybrid basis, we simply reconstruct the signal on the separate subgraphs using their respective transforms and combine the results. As for the GHWT fine-to-coarse dictionary, it has been excluded from this algorithm because it does not conform to the same structure as the HGLET dictionaries and the GHWT coarse-to-fine dictionary. However, if desired, one may compare the best basis returned by Algorithm 6 to the fine-to-coarse best basis returned by the GHWT best basis algorithm and keep the basis that minimizes the cost functional $\mathcal{J}$. This ensures that the set of expansion coefficients corresponding to the resulting basis has the lowest cost out of all choosable HGLET and GHWT bases.

Exactly how many bases are there to choose from? For a general graph, a recursive formula is used to count the number of choosable bases. First we consider the HGLET dictionary. For a fixed level $j < j_{\max}$ and region k, let $\mathcal{C}_{k,\bullet}^j$ denote the number of choosable bases for signals on the vertices $V(G_k^j)$. In other words, $\mathcal{C}_{k,\bullet}^j$ is the number of choosable bases which span the same space as $\{\boldsymbol{\phi}_{k,l}^j\}_l$. We have that

$$\mathcal{C}_{k,\bullet}^j = \begin{cases} 1 & \text{if } V(G_k^j) = 1 \\ \mathcal{C}_{k',\bullet}^{j+1}\mathcal{C}_{k'+1,\bullet}^{j+1} + 1 & \text{otherwise,} \end{cases} \tag{6.1}$$

where $G_{k'}^{j+1}$ and $G_{k'+1}^{j+1}$ are the children subgraphs of G_k^j, as usual. By proceeding from the bottom level of the hierarchical tree up to the root node, we can compute the total number of choosable bases in the HGLET dictionary. This same method also applies to the GHWT coarse-to-fine dictionary.

We can use the same strategy for the GHWT fine-to-coarse dictionary as well. Here, for a fixed level $j > 0$ and tag l, we let $\mathcal{C}_{\bullet,l}^j$ denote the number of choosable bases which span the same space as $\{\boldsymbol{\psi}_{k,l}^j\}_k$. Similar to (6.1), we have the relationship

$$\mathcal{C}_{\bullet,l}^j = \begin{cases} 1 & \text{if } \left|\{\boldsymbol{\psi}_{k,l}^j\}_k\right| = 1 \\ \mathcal{C}_{\bullet,2l}^{j-1}\mathcal{C}_{\bullet,2l+1}^{j-1} + 1 & \text{otherwise.} \end{cases} \tag{6.2}$$

By starting at the bottom of the fine-to-coarse dictionary and proceeding upwards, we arrive at the total number of choosable bases in the dictionary. Note that for each fixed $j \in [0, j_{\max}]$, the basis $\{\boldsymbol{\psi}_{k,l}^j\}_{k,l}$ (i.e., row j in Figs. 5.1 and 5.2) is contained in both the coarse-to-fine and fine-to-coarse dictionaries. Our convention is to attribute these bases to the coarse-to-fine dictionary, and therefore we subtract $(j_{\max} + 1)$ from the number of choosable bases that we compute for the fine-to-coarse dictionary.

Table 6.1 shows the number of choosable bases for several graphs that we consider in this dissertation. Note that our toy example for the HGLET (Fig. 4.1) has the same number of choosable bases as our toy example for the GHWT (Figs. 5.1 and 5.2). This is because the same recursive partitioning tree (Fig. 3.1) is used for both graphs.

	N	$j_{\max}$	HGLET	GHWT Coarse-To-Fine	GHWT Fine-To-Coarse
Fig. 4.1	6	3	11	11	7
Figs. 5.1 and 5.2	6	3	11	11	7
Subset of Dendritic Tree (Fig. 3.2)	687	12	4.6×10^{118}	4.6×10^{118}	1.4×10^{119}
Toronto Road Network (Fig. 7.5)	2202	14	$\approx 10^{368}$	$\approx 10^{368}$	$\approx 10^{379}$
MN Road Network (Figs. 4.2 and 5.3)	2640	14	$\approx 10^{450}$	$\approx 10^{450}$	$\approx 10^{453}$

TABLE 6.1. The number of choosable bases from the HGLET and GHWT dictionaries for several graphs. For each of these graphs the number of choosable bases exceeds the $2^{N/2}$ lower bound for the number of choosable wavelet packet bases, as mentioned in §2.1. (For reference: $10^{118} > 2^{391}$, $10^{368} > 2^{1222}$, and $10^{450} > 2^{1494}$.)

Clearly, we have a massive number of bases from which to choose. What can be said about the basis returned by the best basis algorithm (Algorithm 4, 5, or 6)? Generalizing the result obtained by Coifman and Wickerhauser in [**14**], we have the following guarantee.

PROPOSITION 6.1. *Suppose that $\mathcal{J}$ is a cost functional such that for all sequences $\{x_i\}$ and $\{y_i\}$ and integers $\alpha < \beta < \gamma$,*

$$\begin{aligned} &\text{if} \quad \mathcal{J}\left(\{x_i\}_{i\in[\alpha,\beta)}\right) \le \mathcal{J}\left(\{y_i\}_{i\in[\alpha,\beta)}\right) \\ &\text{and} \quad \mathcal{J}\left(\{x_i\}_{i\in[\beta,\gamma)}\right) \le \mathcal{J}\left(\{y_i\}_{i\in[\beta,\gamma)}\right), \\ &\text{then} \quad \mathcal{J}\left(\{x_i\}_{i\in[\alpha,\gamma)}\right) \le \mathcal{J}\left(\{y_i\}_{i\in[\alpha,\gamma)}\right). \end{aligned} \tag{6.3}$$

Given a signal $\boldsymbol{f}$ on a graph G and a hierarchical tree for the graph, the set $\{b_i\}_{i\in[0,N)}$ of expansion coefficients returned by the best basis algorithm (Algorithm 4, 5, or 6) is the set that minimizes $\mathcal{J}$ over all choosable sets of coefficients in the dictionary (or dictionaries) considered.

PROOF. We will prove the result for Algorithm 4; the proofs for Algorithms 5 and 6 proceed in a similar manner.

Since the HGLET best basis algorithm proceeds one level at a time, from level $j_{\max}$ to level 0, let us define $\{b_i^j\}_{i\in[0,N)}$ to be the set of best basis coefficients after processing levels $j_{\max}, j_{\max}-1, \ldots, j$. For $k \in [0, K^j)$, we see from Step 5 of Algorithm 4 that $\mathcal{J}\left(\{b_i^j\}_{I_{k,\bullet}^j}\right) \le \mathcal{J}\left(\{b_i^{j+1}\}_{I_{k,\bullet}^j}\right)$.

Assume that there exists a choosable set of expansion coefficients $\{a_i\}$ whose cost is smaller than that of the best basis coefficients $\{b_i\}$ returned by Algorithm 4; i.e., $\mathcal{J}\left(\{a_i\}_{i\in[0,N)}\right) < \mathcal{J}\left(\{b_i\}_{i\in[0,N)}\right)$. Let j^* denote the first (i.e., largest) j such that there exists $k^* \in [0, K^{j^*})$ for which

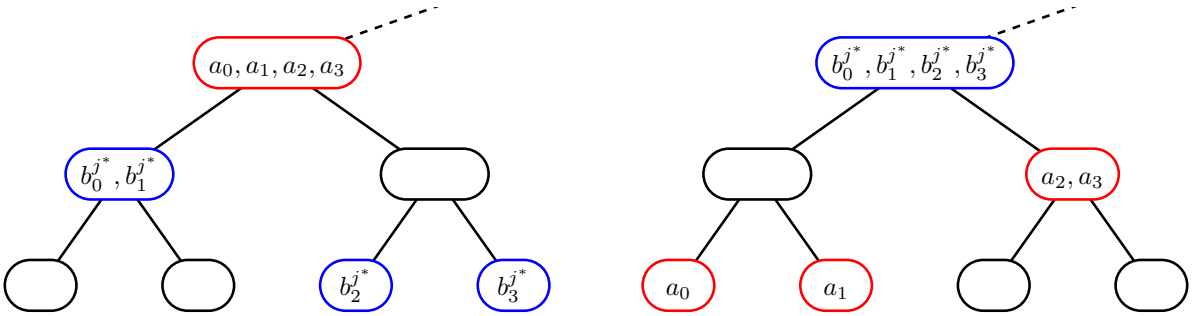
$$\mathcal{J}\left(\{a_i\}_{I_{k^*,\bullet}^{j^*}}\right) < \mathcal{J}\left(\{b_i\}_{I_{k^*,\bullet}^{j^*}}\right). \tag{6.4}$$

We know that there must be some j^* and k^* for which this strict inequality holds because if $\mathcal{J}\left(\{a_i\}_{I_{k,\bullet}^j}\right) \ge \mathcal{J}\left(\{b_i\}_{I_{k,\bullet}^j}\right)$ for all j and k, then (6.3) would imply $\mathcal{J}\left(\{a_i\}_{i\in[0,N)}\right) \ge \mathcal{J}\left(\{b_i\}_{i\in[0,N)}\right)$. Regarding these sets of coefficients $\{a_i\}_{I_{k^*,\bullet}^{j^*}}$ and $\{b_i\}_{I_{k^*,\bullet}^{j^*}}$, there are two possibilities to consider, which we illustrate in Figure 6.1.

Case 1. The coefficients $\{a_i\}$ contain the HGLET coefficients $\{c_{k^*,l}^{j^*}\}_l$, while the coefficients $\{b_i^{j^*}\}$ contain HGLET coefficients corresponding to subgraphs on levels $j > j^*$. However, in Step 5 of the HGLET best basis algorithm we see that $\{b_i^{j^*}\}_{I_{k^*,\bullet}^{j^*}}$ is found by comparing $\mathcal{J}\left(\{b_i^{j^*+1}\}_{I_{k^*,\bullet}^{j^*}}\right)$ and $\mathcal{J}\left(\{c_{k^*,l}^{j^*}\}_l\right)$ and retaining the the coefficients of lower cost. Thus, $\mathcal{J}\left(\{b_i^{j^*}\}_{I_{k^*,\bullet}^{j^*}}\right) \le \mathcal{J}\left(\{c_{k^*,l}^{j^*}\}_l\right) = \mathcal{J}\left(\{a_i\}_{I_{k^*,\bullet}^{j^*}}\right)$, which contradicts (6.4).

Case 2. The coefficients $\{b_i^{j^*}\}$ contain the HGLET coefficients $\{c_{k^*,l}^{j^*}\}_l$, while the coefficients $\{a_i\}$ contain HGLET coefficients corresponding to subgraphs on levels $j > j^*$. Specifically, since j^* is the largest j for which (6.4) occurs, we have that $\{a_i\}_{I_{k^*,\bullet}^{j^*}} = \{b_i^{j^*+1}\}_{I_{k^*,\bullet}^{j^*}}$. However, we know $\mathcal{J}\left(\{b_i^{j^*}\}_{I_{k^*,\bullet}^{j^*}}\right) \leq \mathcal{J}\left(\{b_i^{j^*+1}\}_{I_{k^*,\bullet}^{j^*}}\right) = \mathcal{J}\left(\{a_i\}_{I_{k^*,\bullet}^{j^*}}\right)$, which contradicts (6.4).

Therefore, the assumption is false and the coefficients $\{b_i\}_{i\in[0,N)}$ returned by the HGLET best basis algorithm minimize the cost functional $\mathcal{J}$ over all choosable sets of expansion coefficients from the HGLET dictionary considered. □



(a) Case 1. (b) Case 2.

FIGURE 6.1. Examples of the two cases from the proof of Proposition 6.1. We use blue to identify the coefficients selected by the HGLET best basis algorithm (Algorithm 4) after processing levels $j_{\max}, \ldots, j^*$, and we use red to demarcate coefficients from the set of choosable expansion coefficients $\{a_i\}$ whose cost, by our assumption, is smaller than that of the coefficients $\{b_i\}$ returned by the best basis algorithm.

An important consequence of this result is that the GHWT best basis is at least as good as the Haar basis, according to the cost functional $\mathcal{J}$, because the Haar basis is one of the choosable bases from the GHWT fine-to-coarse dictionary. Another guarantee is that the hybrid best basis obtained via Algorithm 6 is at least as good as any basis from any one of the four dictionaries considered, once again according to $\mathcal{J}$.

As for the cost functional itself, first note that the conditions we require in (6.3) are more general than requiring the cost functional to be additive, as in [**14**]. Common examples of cost functionals which satisfy our criteria include the p-norms $\sum_i |x_i^p|^{1/p}$ for $1 \leq p < 2$ (we do not use the 2-norm because for an orthonormal basis it would result in equality in all cost comparisons), the quasinorms $\sum_i |x_i^p|^{1/p}$ for $0 < p < 1$, the zero pseudonorm $|\{x_i \mid x_i \neq 0\}|$, and the Shannon entropy

of the expansion coefficient distribution (see Eq. (9.1) on page 113). Note that it is not necessary for the cost functional to be convex, since we are simply using it for comparison purposes and we are not performing an optimization search. Of course, the choice of cost functional depends on the task at hand. As the $0 < p < 1$ quasinorms and the $1 \leq p < 2$ p-norms promote sparsity, we have found that they work well for approximation (see Chapter 7). Motivated by the theory behind wavelet shrinkage [**28**], we have used these same cost functionals for denoising (see Chapter 8). Another cost functional that we use for denoising is the Minimum Description Length criterion (MDL, see Chapter 9); even though the MDL does not satisfy (6.3), we can still use it as the cost functional for the best basis algorithm.

CHAPTER 7

Approximation of Signals on Graphs

7.1. Theoretical Results

Classical wavelets have been highly successful for approximation and compression. Examples of their use include the JPEG 2000 image compression standard [**47**] and wavelet orthogonal frequency-division multiplexing (OFDM), which is a means of data encoding commonly used in digital communication [**53**]. As theoretical justification for their use, recall from §2.1 that the number of vanishing moments is a key feature in the design of wavelets. It follows that if a wavelet has p vanishing moments, then the wavelet coefficients for a function with p or more derivatives will decay like 2^{jp} as $j \to -\infty$ [**83**, §7.1]. Furthermore, results on approximation error bounds and wavelet coefficient decay rates have been proven for signals of various types (e.g., Lipschitz, Hölder, Sobolev, Besov, and bounded variation; see [**22**, **23**] and [**45**, Ch. 9]).

As usual, proving similar results for signals on graphs is challenging because we lack the concepts and tools used for classical signals. For example, we lack polynomials, and thus the concept of vanishing moments on a graph domain is undefined. We do not have notions of continuity and derivatives at our disposal. And since any two norms on a finite-dimensional space are equivalent (see, e.g., [**36**, §5.4]), working with normed spaces is not helpful. In short, we do not know which fundamental signals on a graph are important to be able to represent efficiently (i.e., an analog of the polynomials), nor do we have classes of functions for which we would like to have provable results. For these reasons, theoretical results for approximation of signals on graphs are generally lacking, but there have been some developments.

For a graph equipped with a hierarchical tree, Coifman et al. [**9**,**10**,**32**] define a Hölder seminorm and use it to prove various results for the graph Haar basis (which, of course, is a choosable basis from the GHWT fine-to-coarse dictionary). They begin by using the hierarchical tree to define a

distance function between nodes of a graph,

$$d(m,n) := \min\{N_k^j \mid m,n \in V(G_k^j)\}.$$

For a constant $0 < \alpha \leq 1$, they define the Hölder seminorm of a function $\boldsymbol{f}$ on the graph as

$$C_H(\boldsymbol{f}) := \sup_{m \neq n} \frac{|\boldsymbol{f}(n) - \boldsymbol{f}(m)|}{d(m,n)^\alpha}.$$

With these definitions in place, we now extend their result for the generalized Haar transform to our own transforms.

THEOREM 7.1. *For a graph G equipped with a hierarchical tree, suppose that a signal $\boldsymbol{f}$ is Hölder continuous with exponent α and constant $C_H(\boldsymbol{f})$. Then the HGLET (with unnormalized Laplacian L) and GHWT coefficients with $l \geq 1$ satisfy*

$$|c_{k,l}^j| \leq C_H(N_k^j)^{\alpha+1/2}$$
$$|d_{k,l}^j| \leq C_H(N_k^j)^{\alpha+1/2}.$$

PROOF. Our proof follows that of [**9**], although here we use vectors and summations rather than integrals and functions. For the coefficients from the HGLET with unnormalized Laplacian L, we have

$$
\begin{aligned}
c_{k,l}^j &= \left|\left\langle \boldsymbol{f}, \boldsymbol{\phi}_{k,l}^j \right\rangle\right| \\
&= \left|\left\langle \boldsymbol{f} - \left\langle \boldsymbol{f}, \boldsymbol{\phi}_{k,0}^j \right\rangle \boldsymbol{\phi}_{k,0}^j, \boldsymbol{\phi}_{k,l}^j \right\rangle\right| \qquad (\text{since } \left\langle \boldsymbol{\phi}_{k,l}^j, \boldsymbol{\phi}_{k,0}^j \right\rangle = 0) \\
&\leq \left\| \boldsymbol{f} - \left\langle \boldsymbol{f}, \boldsymbol{\phi}_{k,0}^j \right\rangle \boldsymbol{\phi}_{k,0}^j \right\|_2 \|\boldsymbol{\phi}_{k,l}^j\|_2 \qquad (\text{Cauchy-Schwarz}) \\
&= \left(\sum_{n \in V_k^j} \left| \boldsymbol{f}(n) - \sum_{m \in V_k^j} \frac{\boldsymbol{f}(m)}{N_k^j} \right|^2 \right)^{1/2} \qquad (\text{since } \boldsymbol{\phi}_{k,0}^j = \mathbf{1}/\sqrt{N_k^j} \text{ and } \|\boldsymbol{\phi}_{k,l}^j\|_2 = 1) \\
&= \left(\sum_{n \in V_k^j} \left| \sum_{m \in V_k^j} \frac{\boldsymbol{f}(n) - \boldsymbol{f}(m)}{N_k^j} \right|^2 \right)^{1/2} \\
&\leq \left(\sum_{n \in V_k^j} \left| \sum_{m \in V_k^j} \frac{C_H d(m,n)^\alpha}{N_k^j} \right|^2 \right)^{1/2} \\
&\leq \left(\sum_{n \in V_k^j} \left| \sum_{m \in V_k^j} \frac{C_H (N_k^j)^\alpha}{N_k^j} \right|^2 \right)^{1/2} \\
&= \left(\sum_{n \in V_k^j} \left(C_H (N_k^j)^\alpha \right)^2 \right)^{1/2} \\
&= C_H (N_k^j)^{\alpha + 1/2}.
\end{aligned}
$$

The proof proceeds identically for the GHWT, with $d_{k,l}^j$ and $\boldsymbol{\psi}_{k,l}^j$ replacing $c_{k,l}^j$ and $\boldsymbol{\phi}_{k,l}^j$, respectively.

□

REMARK 7.1. *The proof of Theorem 7.1 does not apply to the HGLET with L_{rw} because its basis vectors are not orthonormal with respect to the standard inner product, and it does not apply to the HGLET with L_{sym} because the zeroth eigenvector of L_{sym} is not a constant vector.*

Sharon and Shkolnisky [**74**] achieve coefficient bounds by a different approach. They generalize vanishing moments to the graph setting, saying that a function φ on the graph has m vanishing

moments if it is orthogonal to the first m Laplacian eigenvectors (either those of L or L_{sym})[1]. That is,

$$\langle \varphi, \boldsymbol{\phi}_l \rangle = 0 \qquad \text{for } 0 \leq l < m.$$

They use $E_m(\boldsymbol{f})$ to denote the smallest approximation error of a signal $\boldsymbol{f}$ using the first m Laplacian eigenvectors; since the eigenvectors are orthonormal, we have that

$$E_m(\boldsymbol{f}) = \left\| \sum_{l=m}^{N-1} \langle \boldsymbol{f}, \boldsymbol{\phi}_l \rangle \, \boldsymbol{\phi}_l \right\|_2 . \tag{7.1}$$

They prove that for a normalized function φ that has m vanishing moments and is supported on the vertices $\mathcal{K} \subset V(G)$,

$$| \langle \boldsymbol{f}, \varphi \rangle | \leq E_m(\boldsymbol{f}) \sqrt{|\mathcal{K}|}.$$

However, the factor $\sqrt{|\mathcal{K}|}$ is unnecessary, since

$$\begin{aligned}
| \langle \boldsymbol{f}, \varphi \rangle | &= \left| \left\langle \sum_{l=0}^{N-1} \langle \boldsymbol{f}, \boldsymbol{\phi}_l \rangle \, \boldsymbol{\phi}_l, \varphi \right\rangle \right| \\
&= \left| \left\langle \sum_{l=m}^{N-1} \langle \boldsymbol{f}, \boldsymbol{\phi}_l \rangle \, \boldsymbol{\phi}_l, \varphi \right\rangle \right| \\
&\leq \left\| \sum_{l=m}^{N-1} \langle \boldsymbol{f}, \boldsymbol{\phi}_l \rangle \, \boldsymbol{\phi}_l \right\|_2 \|\varphi\|_2 \qquad \text{(Cauchy-Schwarz)} \\
&= E_m(\boldsymbol{f}).
\end{aligned}$$

In fact, this result is obvious when analyzed from a different perspective. Let $\{\varphi_l\}_{l=0}^{N-1}$ denote an orthonormal basis such that $\varphi_l = \boldsymbol{\phi}_l$ for $0 \leq l < m$ and $\varphi_m = \varphi$. Then for a signal $\boldsymbol{f}$, we have

$$\begin{aligned}
| \langle \boldsymbol{f}, \varphi \rangle |^2 = | \langle \boldsymbol{f}, \varphi_m \rangle |^2 &\leq \sum_{l=m}^{N-1} | \langle \boldsymbol{f}, \varphi_l \rangle |^2 = E_m(\boldsymbol{f})^2 \\
&\leq \sum_{l=0}^{N-1} | \langle \boldsymbol{f}, \varphi_l \rangle |^2 = \|\boldsymbol{f}\|_2^2.
\end{aligned}$$

[1]It is interesting to note that here Sharon and Shkolnisky have used Laplacian eigenvectors as analogs of the polynomials, whereas Hammond et al. [**34**] use Laplacian eigenvectors in place of complex exponentials when defining their graph Fourier transform. Presumably, Sharon and Shkolnisky's reasoning for doing so is because Laplacian eigenvectors are a standard option for approximating signals on graphs, just as the polynomials are for approximating classical 1-D signals.

Of course $|\langle \boldsymbol{f}, \varphi \rangle|$ is less than the norm of the projection of $\boldsymbol{f}$ onto a subspace of the orthonormal basis that includes the span of φ.

We derive a slightly more interesting result for the GHWT, and in particular, the fine-to-coarse dictionary. Let $\mathcal{E}_m(\boldsymbol{f})$ denote the smallest approximation error of a signal $\boldsymbol{f}$ using the global Walsh basis vectors (i.e., the level $j = 0$ GHWT basis vectors) $\boldsymbol{\psi}^0_{0,0}, \boldsymbol{\psi}^0_{0,1}, \ldots, \boldsymbol{\psi}^0_{0,m-1}$. Note that this could be fewer than m basis vectors, since there may be some $l \in [2, m)$ for which there is no $\boldsymbol{\psi}^0_{0,l}$; for example, there is no $\boldsymbol{\psi}^0_{0,5}$ in Figure 5.2. We prove the following bound for GHWT coefficients.

PROPOSITION 7.1. *Consider a signal $\boldsymbol{f}$ on a graph. All GHWT expansion coefficients $d^j_{k,l}$ for which $l > \lfloor (m-1)/2^j \rfloor$ satisfy*

$$|d^j_{k,l}| \leq \mathcal{E}_m(\boldsymbol{f}).$$

PROOF. Let $d^{\hat{\jmath}}_{\hat{k},\hat{l}}$ be a GHWT expansion coefficient, where $\hat{l} > \lfloor (m-1)/2^{\hat{\jmath}} \rfloor$. The first step in the proof is to select an orthonormal basis containing $\{\boldsymbol{\psi}^0_{0,l}\}_{0 \leq l < m}$ and $\boldsymbol{\psi}^{\hat{\jmath}}_{\hat{k},\hat{l}}$. Let $m_{\hat{\jmath}} := \lfloor (m-1)/2^{\hat{\jmath}} \rfloor$. We have

$$\begin{aligned} m_{\hat{\jmath}} + 1 &> \frac{m-1}{2^{\hat{\jmath}}} \\ 2^{\hat{\jmath}}(m_{\hat{\jmath}} + 1) &> m - 1 \\ m^* := 2^{\hat{\jmath}}(m_{\hat{\jmath}} + 1) &\geq m. \end{aligned}$$

We can also derive the following equalities, which we will need momentarily:

$$\begin{aligned} \frac{m^*}{2^{\hat{\jmath}}} &= m_{\hat{\jmath}} + 1 \\ \left\lfloor \frac{m^* - 1}{2^{\hat{\jmath}}} \right\rfloor &= m_{\hat{\jmath}}. \end{aligned} \tag{7.2}$$

We claim that $\{\boldsymbol{\psi}^0_{0,l}\}_{0 \leq l < m^*} \cup \{\boldsymbol{\psi}^{\hat{\jmath}}_{k,l}\}_{0 \leq k < K^{\hat{\jmath}};\ l > m_{\hat{\jmath}}}$ is an orthonormal basis that meets our needs (this basis is choosable from the GHWT fine-to-coarse dictionary). To see that this is indeed a basis, we appeal to (5.6), which tells us that

$$\begin{aligned} \mathrm{span}\{\boldsymbol{\psi}^{\hat{\jmath}}_{k,l} \mid l \leq m_{\hat{\jmath}}\}_k &= \mathrm{span}\{\boldsymbol{\psi}^0_{k,l} \mid \lfloor l/2^{\hat{\jmath}} \rfloor \leq m_{\hat{\jmath}}\}_k \\ &= \mathrm{span}\{\boldsymbol{\psi}^0_{k,l} \mid l < m^*\}_k, \end{aligned}$$

where we use (7.2) to arrive at the second line. It follows that the proposed set is indeed a basis.

In terms of this basis, we have that

$$\mathcal{E}_m(\boldsymbol{f}) = \left\| \sum_{m \leq l < m^*} \left\langle \boldsymbol{f}, \boldsymbol{\psi}^0_{0,l} \right\rangle \boldsymbol{\psi}^0_{0,l} + \sum_{m_{\hat{\jmath}} < l} \sum_{0 \leq k < K^{\hat{\jmath}}} \left\langle \boldsymbol{f}, \boldsymbol{\psi}^{\hat{\jmath}}_{k,l} \right\rangle \boldsymbol{\psi}^{\hat{\jmath}}_{k,l} \right\|_2 .$$

Computing the expansion coefficient, we find that

$$\begin{aligned}
|d^{\hat{\jmath}}_{\hat{k},\hat{l}}| :&= \left| \left\langle \boldsymbol{f}, \boldsymbol{\psi}^{\hat{\jmath}}_{\hat{k},\hat{l}} \right\rangle \right| \\
&= \left| \left\langle \sum_{0 \leq l < m^*} \left\langle \boldsymbol{f}, \boldsymbol{\psi}^0_{0,l} \right\rangle \boldsymbol{\psi}^0_{0,l} + \sum_{m_{\hat{\jmath}} < l} \sum_{0 \leq k < K^{\hat{\jmath}}} \left\langle \boldsymbol{f}, \boldsymbol{\psi}^{\hat{\jmath}}_{k,l} \right\rangle \boldsymbol{\psi}^{\hat{\jmath}}_{k,l},\ \boldsymbol{\psi}^{\hat{\jmath}}_{\hat{k},\hat{l}} \right\rangle \right| \\
&= \left| \left\langle \sum_{m \leq l < m^*} \left\langle \boldsymbol{f}, \boldsymbol{\psi}^0_{0,l} \right\rangle \boldsymbol{\psi}^0_{0,l} + \sum_{m_{\hat{\jmath}} < l} \sum_{0 \leq k < K^{\hat{\jmath}}} \left\langle \boldsymbol{f}, \boldsymbol{\psi}^{\hat{\jmath}}_{k,l} \right\rangle \boldsymbol{\psi}^{\hat{\jmath}}_{k,l},\ \boldsymbol{\psi}^{\hat{\jmath}}_{\hat{k},\hat{l}} \right\rangle \right| \\
&\leq \left\| \sum_{m \leq l < m^*} \left\langle \boldsymbol{f}, \boldsymbol{\psi}^0_{0,l} \right\rangle \boldsymbol{\psi}^0_{0,l} + \sum_{m_{\hat{\jmath}} < l} \sum_{0 \leq k < K^{\hat{\jmath}}} \left\langle \boldsymbol{f}, \boldsymbol{\psi}^{\hat{\jmath}}_{k,l} \right\rangle \boldsymbol{\psi}^{\hat{\jmath}}_{k,l} \right\|_2 \|\boldsymbol{\psi}^{\hat{\jmath}}_{\hat{k},\hat{l}}\|_2 \qquad \text{(Cauchy-Schwarz)} \\
&= \mathcal{E}_m(\boldsymbol{f}).
\end{aligned}$$

□

In terms of the fine-to-coarse dictionary in Figure 5.2, the signficance of this is that it provides a bound for all GHWT coefficients whose blocks lie entirely to the right of $\boldsymbol{\psi}^0_{0,m}$. This proposition is not useful for the GHWT coarse-to-fine dictionary because we cannot choose a basis which contains some but not all of the Walsh basis vectors. Likewise, the inequality (7.1) involving Laplacian eigenvectors that Sharon and Shkolnisky derived is not useful for the HGLET because we cannot choose a basis with some but not all of the global Laplacian eigenvectors.

Sharon and Shkolnisky also define a generalization of Besov spaces in the graph setting. For a fixed orthonormal basis $\{\varphi_l\}_{l=0}^{N-1}$ and a parameter $\tau > 0$, they define the τ-measure of a function $\boldsymbol{f}$ as

$$|\boldsymbol{f}|_\tau := \left(\sum_{l=0}^{N-1} |\left\langle \boldsymbol{f}, \varphi_l \right\rangle|^\tau \right)^{1/\tau} . \tag{7.3}$$

They note that for all signals, the τ-measure satisfies

$$\|\boldsymbol{f}\|_2 \leq |\boldsymbol{f}|_\tau \leq N^{\frac{1}{\tau}-\frac{1}{2}}\|\boldsymbol{f}\|_2.$$

They define discrete analogs of the Besov spaces as

$$B_{\tau,M} = \{\boldsymbol{f} \;\big|\; |\boldsymbol{f}|_\tau < M \text{ and } \|\boldsymbol{f}\| = 1\}, \qquad \text{where } 0 < \tau < 2, \quad 1 \leq M \leq N^{\frac{1}{\tau}-\frac{1}{2}}.$$

Following the notation of [**22**], let $P_n\boldsymbol{f}$ denote the best nonlinear n-term approximation of $\boldsymbol{f}$ in the basis. Sharon and Shkolnisky prove the following bound on the approximation error.

THEOREM 7.2. *[**74**] For a fixed orthonormal basis $\{\varphi_l\}_{l=0}^{N-1}$ and a parameter $0 < \tau < 2$,*

$$\|\boldsymbol{f} - P_n\boldsymbol{f}\|_2 \leq \frac{|\boldsymbol{f}|_\tau}{n^\beta}, \tag{7.4}$$

where $|\boldsymbol{f}|_\tau$ corresponds to $\{\varphi_l\}_{l=0}^{N-1}$ and $\beta = \frac{1}{\tau} - \frac{1}{2}$.

As the HGLET (with L and L_{sym} but not with L_{rw}) and GHWT yield overcomplete dictionaries of orthonormal bases, this theorem applies directly to any basis we select from those dictionaries; for the GHWT, this includes both the coarse-to-fine and fine-to-coarse dictionaries. Furthermore, note that the τ-measure satisfies the requirements (6.3) from Proposition 6.1 for our best basis algorithms. Therefore, we have the following corollary.

COROLLARY 7.1. *For a signal $\boldsymbol{f}$, consider one or more dictionaries of orthonormal expansion coefficients (i.e., those corresponding to the HGLET with L, the HGLET with L_{sym}, GHWT coarse-to-fine, or GHWT fine-to-coarse). For $\tau \in (0, 2)$, using the τ-measure as the cost functional for the appropriate best basis algorithm (Algorithm 4, 5, or 6) yields the choosable orthonormal basis which minimizes $|\boldsymbol{f}|_\tau$ and therefore has the best bound for nonlinear approximation error in* (7.4).

Of course, this corollary does not tell us which τ-measure should be used as the best basis cost functional in order to achieve the best approximation bound in (7.4). Fortunately, the best basis search is quick and inexpensive, and thus we can perform the search over a range of τ values (e.g., $\tau = 0.1, 0.2, \ldots, 1.9$), obtaining a set of best basis coefficients for each one. We can then specify a constant n (e.g., $n = [0.1N]$) and choose the particular τ and corresponding basis which minimizes

the upper bound $|\boldsymbol{f}|_\tau/n^\beta$. However, in practice this does not always lead to the best choice of τ because the bound is not tight enough.

What we can do instead is to search over a range of τ values and choose the particular best basis that yields the smallest cumulative relative error. To do this, we find the N best basis expansion coefficients for each τ and then compute a vector of length N containing the relative approximation errors when $1, 2, \ldots, N$ coefficients are retained[2]. This is easily done for orthonormal bases; for bases which are not orthonormal, this can still be accomplished in a simple manner by forming the $N \times N$ matrix of best basis vectors. We then take the sum of this vector of relative errors; in other words, letting $P_n\boldsymbol{f}$ denote the best n-term nonlinear approximation of $\boldsymbol{f}$ with respect to the basis, we compute

$$\text{cumulative relative error } = \sum_{n=1}^{N} \|\boldsymbol{f} - P_n\boldsymbol{f}\|_2/\|\boldsymbol{f}\|_2. \tag{7.5}$$

We search over the range of τ values and select the basis which minimizes this sum. In terms of Figures 7.2 and 7.6, we are selecting the τ that yields the smallest area under the relative error curve. As we will use this strategy often, we formally describe it in Algorithm 7. (Of course, the user could specify a different range of τ values over which to search.)

[2]The relative error will be zero when N coefficients are retained, and thus we only need a vector of length $N-1$. However, it is convenient to have this vector be of the same length as the vector of coefficients, and thus we include the Nth entry.

Algorithm 7 Best Basis Algorithm with Minimal Relative Error Criterion (`HGLET_GHWT_BestBasis_minrelerror.m`)

0: For a signal $\boldsymbol{f}$ on the graph, use the HGLET and/or GHWT to generate one or more dictionaries of expansion coefficients

1: **for** $\tau = 0.1,\ 0.2,\ \ldots, 1.9$ **do**

2: Perform the appropriate best basis search (Algorithm 4, 5, or 6) using the τ-measure (7.3) as the cost functional; that is, for a set of expansion coefficients $\{b_i\}_{i\in I}$, the associated cost is

$$\mathcal{J}\left(\{b_i\}_{i\in I}\right) = \left(\sum_{i\in I} |b_i|^\tau\right)^{1/\tau}.$$

3: Compute a vector $\boldsymbol{r}_\tau \in \mathbb{R}^N$ of relative reconstruction errors when $1, 2, \ldots, N$ coefficients are retained

4: **end for**

5: Return the set of expansion coefficients corresponding to τ^*, where

$$\tau^* = \underset{\tau\in\{0.1,0.2,\ldots,1.9\}}{\arg\min} \sum_{n=1}^{N} \boldsymbol{r}_\tau(n).$$

Although we refer to this as a best basis algorithm, in actuality it is merely a strategy (or a meta algorithm) for using the best basis algorithms (Algorithms 4, 5, and 6). Note that we can use this method for the HGLET with L_{rw} even though the basis is not orthonormal with respect to the standard inner product. However, Theorem 7.2 and Corollary 7.1 will not apply to the resulting basis. Another fact worth mentioning is that this minimal relative error criterion does not satisfy the requirements (6.3) from Proposition 6.1, as numerical examples have demonstrated. In other words, comparing sets of expansion coefficients from within a dictionary based on the sums of their relative reconstruction errors does not lead to the basis which has the smallest sum of relative reconstruction errors.

7.2. Experimental Results

Having proven some theoretical approximation results for our transforms, we now present some experimental results comparing our methods to other transforms. The signals that we analyze are shown in Figures 7.1 and 7.5. We emphasize that both of these data sets are real, thereby avoiding the concern of designing a synthetic signal that is either unrealistic or biased towards certain transforms. For example, designing a signal using Laplacian eigenvectors may confer an unfair advantage to the HGLET or methods based on the graph Fourier transform.

In addition to the graph Haar basis, the graph Walsh basis (i.e., level $j = 0$ of the GHWT coarse-to-fine dictionary), and the eigenvectors of the unnormalized Laplacian L, we compare our methods to two other graph transforms. For the sake of a fair comparison, we only considered critically sampled transforms (which ruled out [**34**,**76**,**77**,**78**]). Granted, the transforms considered use a fixed basis while our methods involve choosing a basis from an overcomplete dictionary, but this is the fairest comparison we can make. Another consideration when selecting transforms to compare against was that we wanted to include one method from each of the two broad categories we covered in our review of graph transforms in §2.3: those based on the graph Fourier transform and those based on vertex transformations. With these criteria in mind, the two transforms that we chose were the graph-QMF [**51**] and Laplacian multiwavelets [**74**]. As we mentioned in our description of Laplacian multiwavelets (§2.3.2), a parameter m needs to be specified. We used two values, both of which are used in example code that the authors provide: $m = 10$ and $m = \lfloor N/20 \rfloor$.

As for our own transforms, we use the HGLET best basis, the GHWT best basis, and the hybrid best basis (Algorithms 4, 5, and 6). For the hybrid best basis algorithm, we consider all four dictionaries: HGLET with L, HGLET with L_{rw}, HGLET with L_{sym}, and GHWT coarse-to-fine. In addition, for the dendritic tree data set, we then compare this hybrid best basis to the GHWT fine-to-coarse best basis, thereby ensuring that we select "the best of the best bases." In order to avoid the need to specify a cost functional, we utilize the minimal relative error criterion (Algorithm 7), which determines the best τ-measure to be used as the cost functional for the appropriate best basis algorithm (Algorithm 4, 5, or 6). To generate the partitioning tree for our transforms, we perform recursive bipartitioning using the Fiedler vector of L_{rw}, as described in Chapter 3; we use this same method to generate the partitioning tree required by Laplacian multiwavelets.

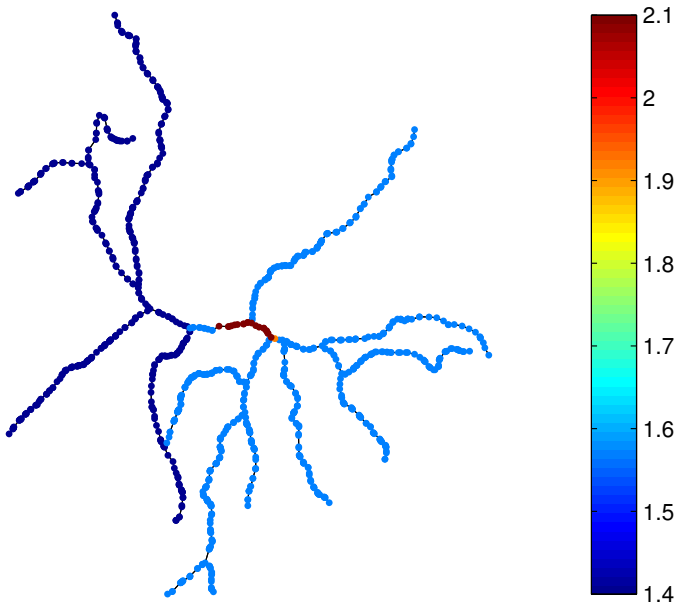
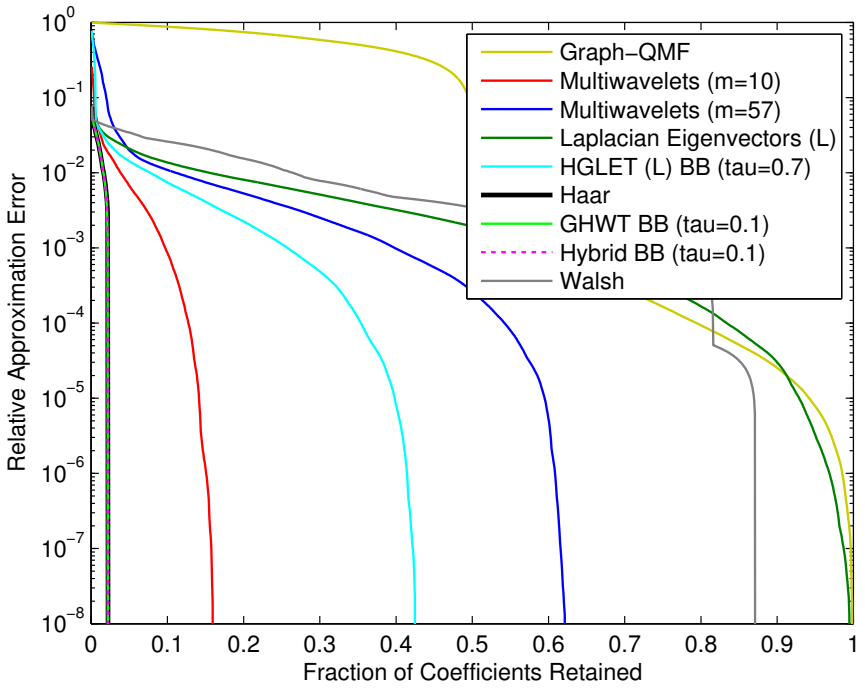


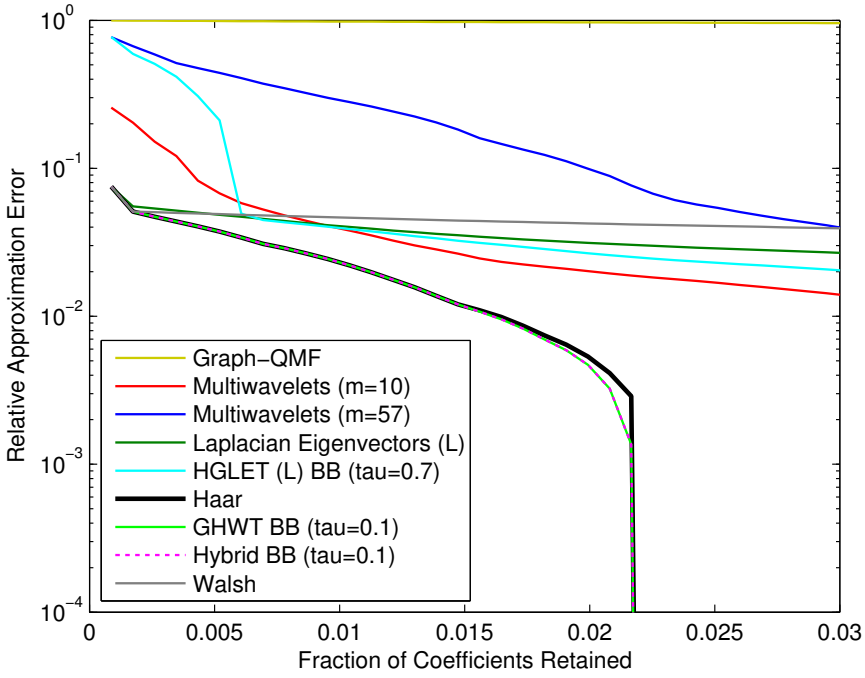
FIGURE 7.1. A dendritic tree ($N = 1154$ nodes and $M = 1153$ edges), with the values of the signal corresponding to the thickness of the dendrite. A subset of this graph was used for the recursive partitioning illustration in Figure 3.2.

Figure 7.1 shows the thickness data on the dendritic tree, which was measured by Coombs et al. [**16**]. Admittedly, this is a simple data set due to the fact that the signal assumes only four unique values, but this is due to limitations in measurement precision and not by design. Figure 7.2 shows relative approximation errors for the transforms we compared. For this signal, the best approximation results were achieved by the GHWT best basis, which comes from the fine-to-coarse dictionary and is illustrated in Figure 7.3. This same basis is selected by the hybrid best basis algorithm, as it was found to have a smaller cumulative relative error than the basis selected from the four dictionaries. This was followed closely by the Haar basis. It comes as no surprise that the Haar basis performs very well for this piecewise constant signal. Moreover, the similarity in performance between the GHWT fine-to-coarse best basis and the Haar basis can be explained by the similarity in their bases, which share all but two basis functions. We also see that the HGLET (L) best basis outperforms the Laplacian (L) eigenvectors, which is expected since the Laplacian eigenvectors are a choosable basis from the HGLET dictionary. The advantage of the HGLET best basis is that its basis vectors have varying degrees of localization, as illustrated in Figure 7.4, whereas the Laplacian eigenvectors are global (except for some eigenvectors corresponding to larger

eigenvalues which may be localized, as in Fig. 2.8). As Figure 7.4 shows, the HGLET best basis reflects the structure of the signal and thus can serve as a means of analyzing a signal on a graph. Continuing with our analysis of the relative error curves, we see that the Walsh basis performs similarly to the Laplacian eigenvectors, which is not surprising because they are both global bases.



(a)



(b)

FIGURE 7.2. (a) Relative approximation error as a function of coefficients kept for the dendritic tree data set (Figure 7.1). (b) A zoomed-in version of the figure.

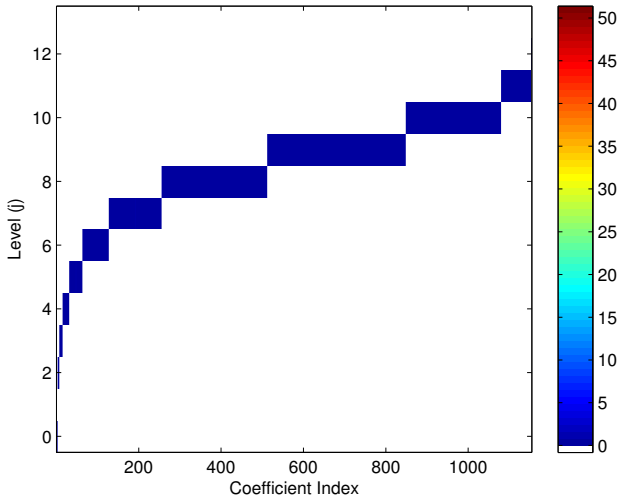
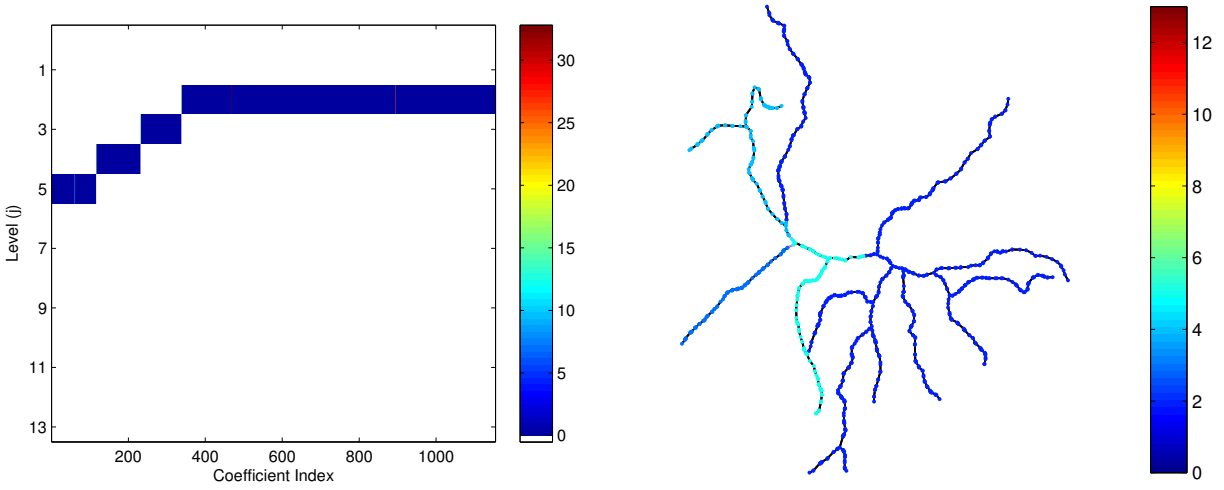


FIGURE 7.3. The locations of the GHWT best basis coefficients in the fine-to-coarse dictionary for the dendritic tree thickness data. These coefficients differ from the Haar coefficients only in two places, namely, the third and fourth coefficients. Color corresponds to the magnitude of the coefficients, although the fact that so many coefficients are zero or nearly zero makes it difficult to notice the small number of larger coefficients in the bottom left corner of the figure. (The fact that level $j = 0$ is at the bottom of the vertical axis this indicates that the basis originates from the fine-to-coarse dictionary.)



(a) (b)

FIGURE 7.4. (a) The locations of the HGLET (L) best basis coefficients from within the dictionary. Once again, color corresponds to the absolute values of the coefficients. (b) An illustration of the regions from which the best basis coefficients originate. The color of the nodes corresponds to their level $j \in [0, j_{\max}]$, and partitioned edges are drawn in pink. (In order to see these edges it is necessary to zoom in.)

For our second experiment we introduce a new data set: traffic volume data on the Toronto road network, as seen in Figure 7.5. This information is made publicly available by the city of Toronto[3]. The traffic data was collected over 24 hour windows (i.e., it is not the case that all intersections were monitored over the same 24 hour time span). Using the street names and intersection coordinates included in the data set, we generated the road network of Toronto. This graph and its corresponding signal are freely distributed as part of the MTSG Toolbox.

[3] http://www1.toronto.ca/wps/portal/contentonly?vgnextoid=417aed3c99cc7310VgnVCM1000003dd60f89RCRD

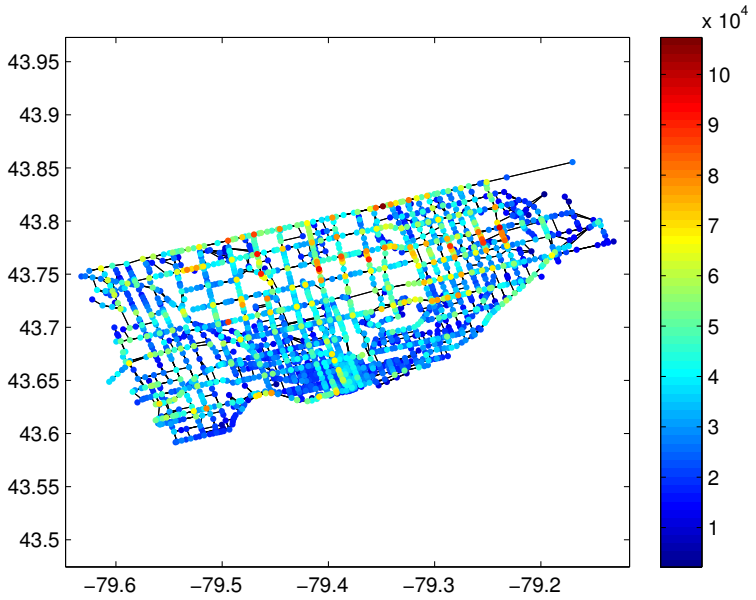
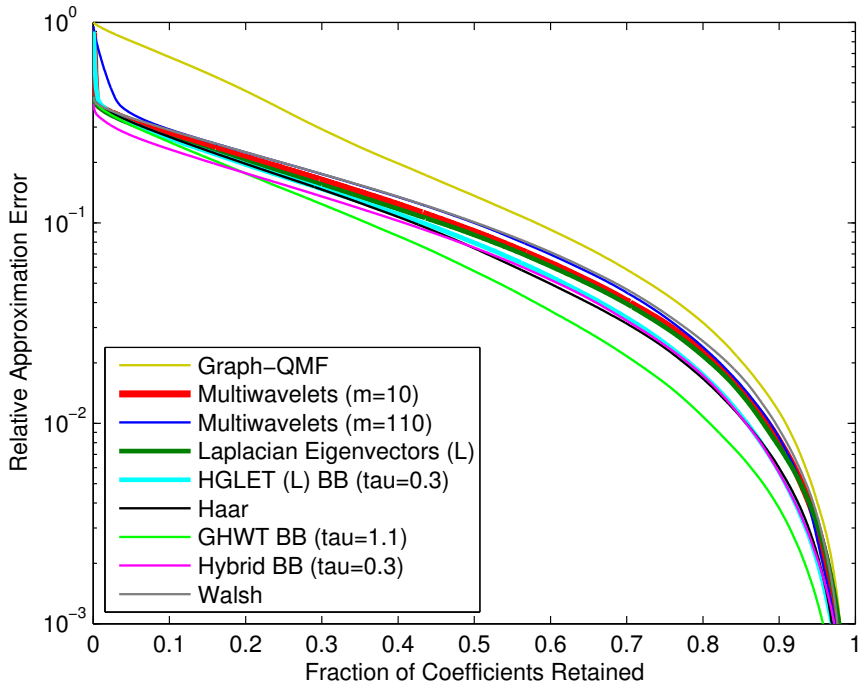


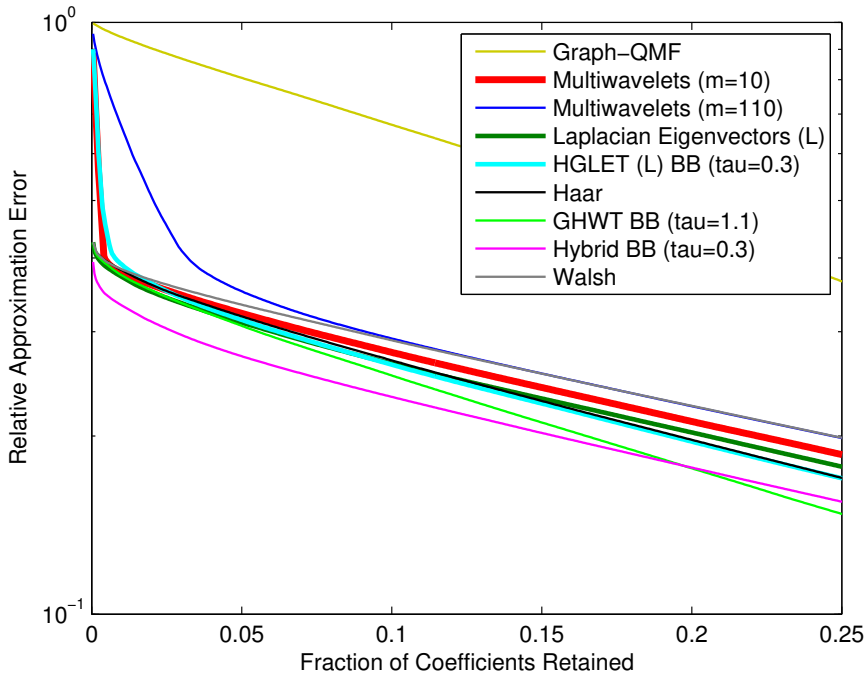
FIGURE 7.5. Traffic volume data over a 24 hour period at intersections in the road network of Toronto ($N = 2202$ nodes and $M = 4877$ edges).

As we did for the dendritic tree, Figure 7.6 shows the relative approximation errors for the Toronto data set. Once again, the best performance is achieved by the GHWT best basis (which originates from the fine-to-coarse dictionary), although when very few coefficients are retained it is outperformed by the hybrid best basis. Unlike in our experiments with the dendritic tree, for the Toronto road network we do not compare the cost of the hybrid best basis (i.e., the output of Algorithm 6) to the cost of the GHWT best basis. Although the GHWT best basis has a lower cumulative relative error, we display the results for the hybrid best basis so that the two can be compared. As with the dendritic tree, the GHWT best basis comes from the fine-to-coarse dictionary, and we illustrate the locations of the selected coefficients in Figure 7.7a. Unlike with the dendritic tree, the structure of the GHWT best basis differs radically from that of the Haar basis (Fig. 7.7b). Recalling that the basis vectors are global on level $j = 0$ and become more localized as j increases, we see that the GHWT best basis has far more basis vectors with large supports. Furthermore, given that the number of oscillations in the basis vectors on a particular level j generally increases from left to right (as our $N = 6$ example in Figure 5.2 illustrates), we note that the GHWT best basis contains basis vectors with much more oscillation than those in the Haar basis, which assume only two distinct nonzero values. Thus, the best basis algorithm

validates what we would expect: more oscillatory basis vectors are advantageous for representing this signal. However, it is also necessary to have some basis vectors which are more localized, as evidenced by the fact that the Walsh basis is outperformed by the GHWT best basis and the Haar basis. As for the hybrid best basis, this is actually the set of eigenvectors of L_{sym}. Intuitively, this makes sense because we expect that intersections involving more streets will have more traffic volume, and the degree normalization of L_{sym} should help its eigenvectors to capture this.



(a)



(b)

FIGURE 7.6. (a) Relative approximation error as a function of coefficients kept for the Toronto traffic volume data set (Figure 7.5). (b) A zoomed-in version of the figure.

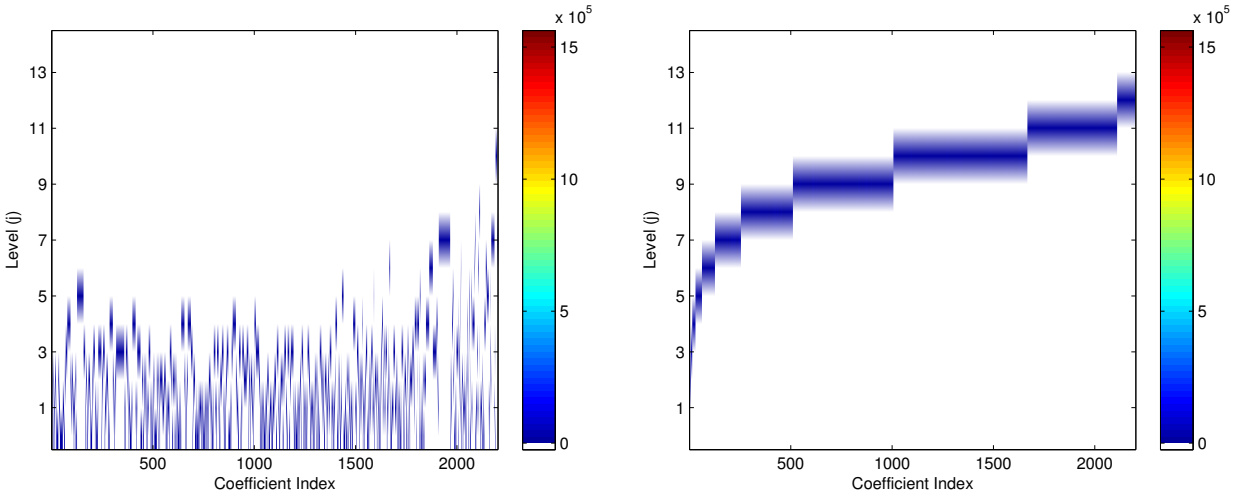


FIGURE 7.7. The locations of (a) the GHWT best basis coefficients and (b) the Haar coefficients within the fine-to-coarse dictionary for the Toronto traffic data.

Following the GHWT best basis and the hybrid best basis, we find that the Haar basis and the HGLET (L) best basis perform similarly. Figure 7.8 illustrates the coefficients and subgraphs which correspond to the HGLET best basis. As with the GHWT best basis, the HGLET best basis is comprised mainly of more global (i.e., smaller j) basis vectors. Furthermore, the subgraphs chosen by the best basis algorithm (Fig. 7.8b) serve as a decomposition of the signal into regions of similar characteristics.

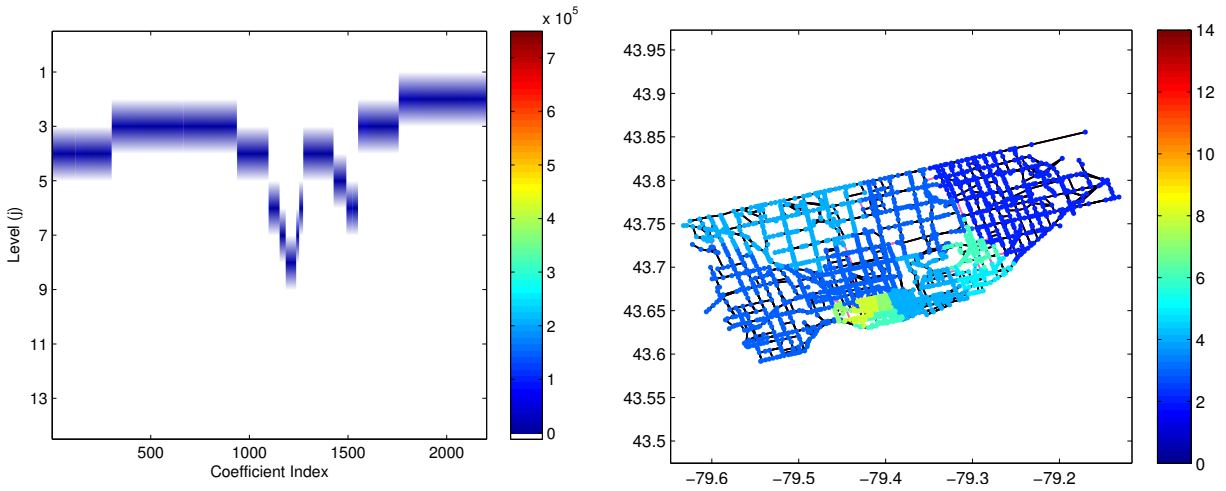


FIGURE 7.8. (a) The locations of the HGLET (L) best basis coefficients from within the dictionary, with color corresponding to the magnitude of the coefficients. (b) An illustration of the regions from which the best basis coefficients originate. The color of the nodes denotes their level $j \in [0, j_{\max}]$, and edges drawn in pink are partitioned. (Zooming in may be necessary in order to see these edges.)

7.3. Summary

We have presented both theoretical and experimental results demonstrating the effectiveness of our transforms for approximation of signals on graphs. We adapted results from Coifman et al. [**9**,**10**,**32**] and Sharon and Shkolnisky [**74**] to our own transforms. Moreover, we showed that the use of a τ-measure as the cost functional for the best basis algorithm minimizes the approximation bound (7.4) over all choosable bases considered.

We validated this theory with approximation experiments on non-synthetic data sets. The GHWT best basis algorithm demonstrated that it can identify the Haar basis when it is well-suited to the signal, and it can also select a very different basis which outperforms the Haar basis. Similarly for the HGLET, we showcase that the best basis algorithm can select the global Laplacian eigenvectors when they efficiently capture the signal, and it can choose a basis which decomposes the signal into regions of similar characterisitics, thus enabling the basis to outperform the Laplacian eigenvectors.

CHAPTER 8

Denoising of Signals on Graphs

Building upon their effectiveness for approximation, classical wavelets have also been applied to the task of denoising with much success. The reasons why this works are because: (1) a basis that is efficient for approximation concentrates the majority of a signal's energy into a small number of large coefficients; and (2) "Gaussian white noise in any one orthogonal basis is again a white noise in any other (and with the same amplitude)" [**25**]. Based on these insights, Donoho et al. devised wavelet shrinkage [**28**], which yields nearly optimal nonlinear estimators. Their method is simple and straightforward: apply the wavelet transform to the signal, soft-threshold the coefficients (excluding the scaling coefficients), and then reconstruct.

We employ this same strategy in order to denoise signals on graphs using our transforms. Of course, a precursor step when denoising with the HGLET and GHWT is to first select a best basis. As with our approximation experiments, we do so by using the minimal relative error criterion (Algorithm 7) to select the best τ-measure for the appropriate best basis algorithm (Algorithm 4, 5, or 6). Having selected a basis, the next step is to threshold the coefficients. For a threshold $T > 0$, we soft-threshold HGLET coefficients as

$$\mathfrak{c}_{k,l}^j = \begin{cases} c_{k,l}^j & \text{if } l = 0 \\ \text{sign}(c_{k,l}^j)(|c_{k,l}^j| - T)_+ & \text{otherwise.} \end{cases}$$

We obtain soft-thresholded GHWT coefficients $\mathfrak{d}_{k,l}^j$ in the same manner, again leaving coefficients with $l = 0$ unchanged. The final step is to reconstruct the signal using the thresholded coefficients.

A key aspect of this denoising procedure is determining the appropriate threshold T. We will explain our method of selecting T as we present some examples. Figure 8.1 shows a mutilated Gaussian on the Minnesota road network and a noisy version of this signal with white Gaussian noise added to make its SNR 5.00 dB. For the sake of transparency, the formula that we use to

compute the signal-to-noise ratio of a signal $\widehat{\boldsymbol{f}}$ and its noise-free counterpart $\boldsymbol{f}$ is

$$\text{SNR} = 20 \log_{10} \frac{\|\boldsymbol{f}\|_2}{\|\widehat{\boldsymbol{f}} - \boldsymbol{f}\|_2}.$$

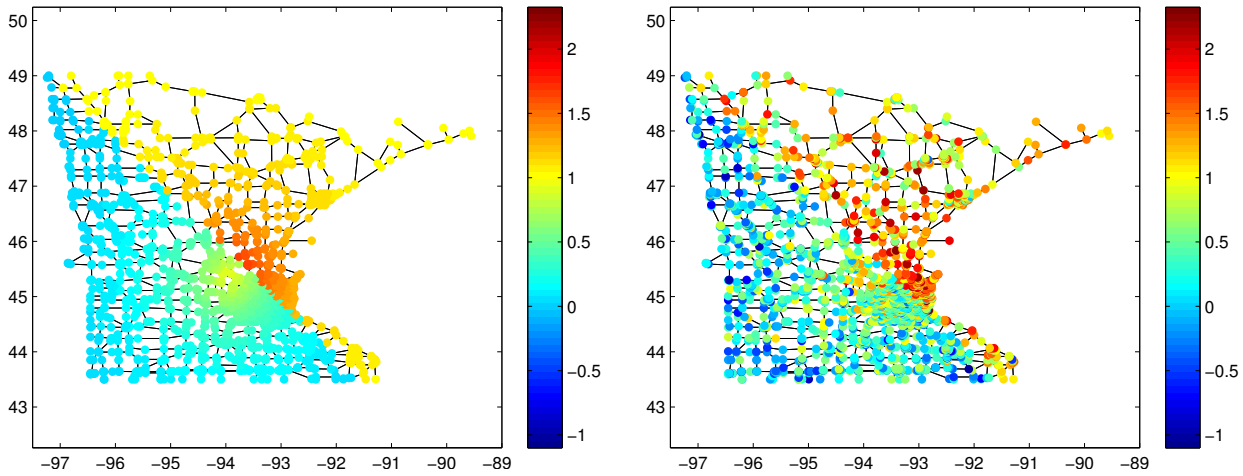
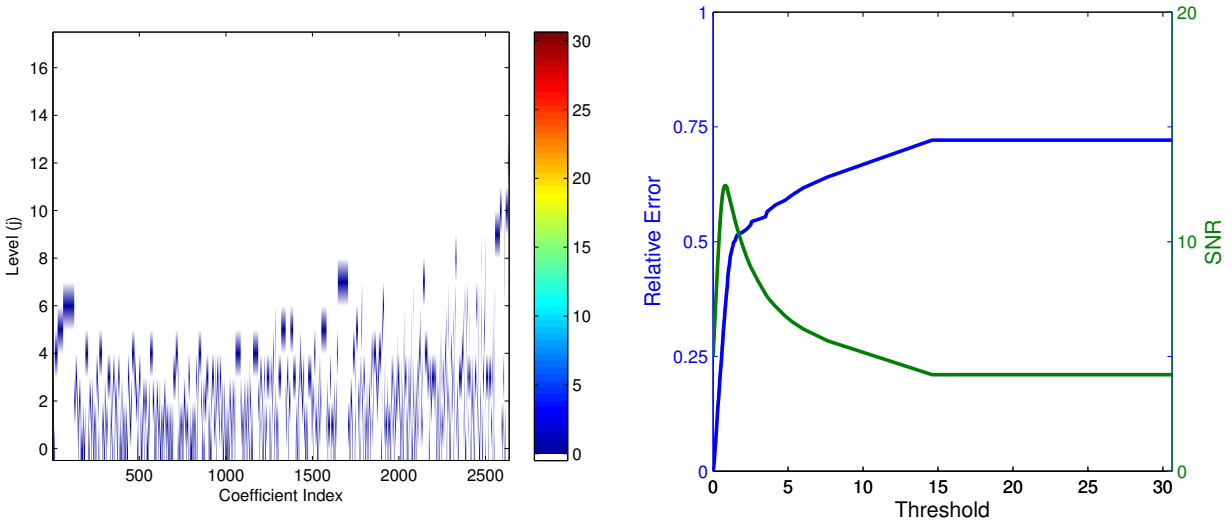


FIGURE 8.1. (a) A mutilated Gaussian on the Minnesota road network ($N = 2636$ vertices, $M = 3293$ edges, inverse Euclidean edge weights). (b) A noisy version of the mutilated Gaussian with SNR 5.00 dB.

Before we can determine a threshold, we first analyze the signal with the GHWT and select a basis. For this purpose we use the minimal relative error best basis algorithm (i.e., Algorithm 7), and we display the table of coefficients of the resulting basis in Figure 8.2a. We generate a curve of the relative reconstruction errors (i.e., the relative error in the reconstruction of the noisy signal) in which we use as thresholds zero and all but the largest coefficient magnitude. For this task we use hard-thresholding, and thus the best n term nonlinear approximation of the signal corresponds to hard-thresholding with the $(n+1)$st largest coefficient magnitude. We also generate a curve of the signal-to-noise ratios, again using as thresholds zero and all but the largest coefficient magnitude, but this time we use soft-thresholding. Both of these curves are displayed in Figure 8.2b. Of course, computing the SNR curve requires the noise-free signal, but computing the relative error curve does not and thus it can be used to aid in denoising efforts.

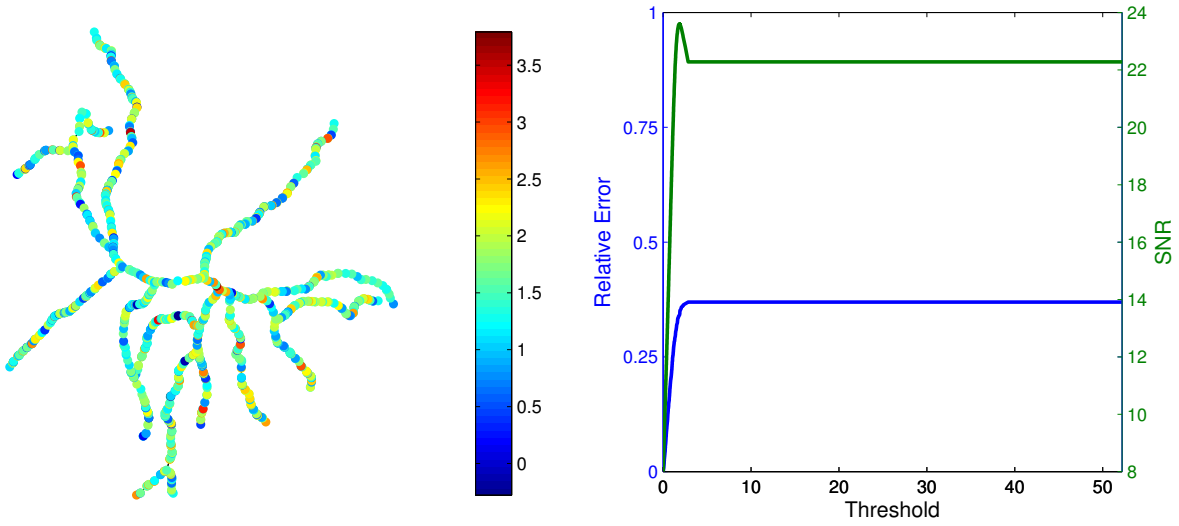


(a) (b)

FIGURE 8.2. (a) The table of coefficients for the GHWT best basis ($\tau = 0.9$) for the noisy mutilated Gaussian in Figure 8.1b. As in our approximation experiments, we use the minimal relative error best basis algorithm to determine the cost functional and select the basis. (b) Relative error (for reconstruction of the noisy signal) and signal-to-noise ratio as functions of the threshold for the mutilated Gaussian on the Minnesota road network. Hard-thresholding is used for generating the relative error curve, while soft-thresholding is used for the SNR curve.

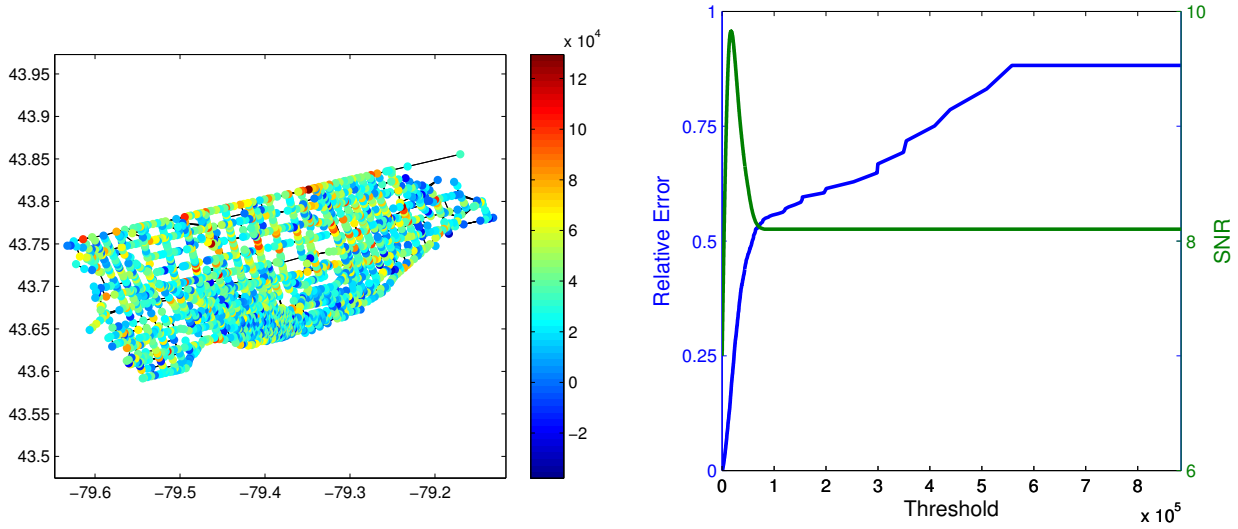
Note the behavior of the two curves: the SNR curve rises quickly as the threshold increases from zero, while the relative error curve starts dropping rapidly when the threshold decreases towards zero. After attaining its maximum, the SNR curve falls quickly to the SNR of the noisy signal (5.00 dB). We observe similar behavior for noisy versions of the previously introduced dendritic tree data (Figure 8.3) and Toronto traffic data (Figure 8.4). As we lower the threshold (i.e., proceed from right to left in the plots), the reconstruction error steadily declines while the threshold is relatively large. This is because, as mentioned at the start of this chapter, a basis that is efficient for approximation concentrates the majority of the signal's energy into a small number of large coefficients, and these are the coefficients retained at the higher thresholds. When the threshold is high, only a few coefficients are retained, which explains why the relative error curve is constant on the right side of the plot and nearly flat in the middle of the plot. On the other hand, there are a large number of small coefficients which capture the detail and noise in the signal. As the

threshold decreases, more and more of these are retained, which explains the rapid decrease in the relative error of the reconstructions of the noisy signal.



(a) (b)

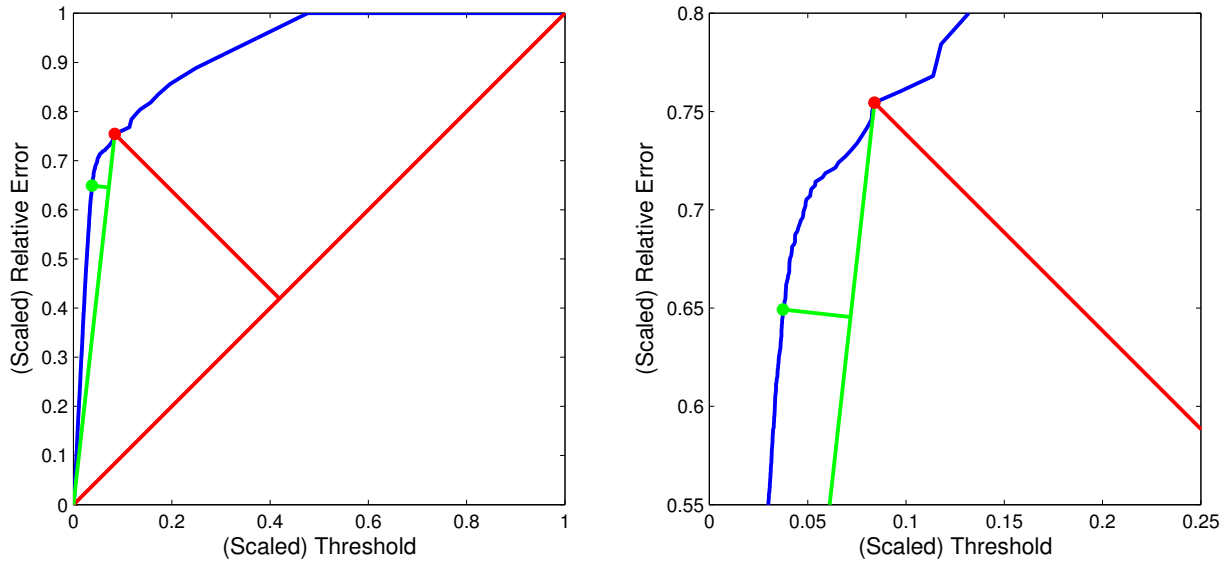
FIGURE 8.3. (a) A noisy version of the dendritic tree data from Figure 7.1 with SNR 8.00 dB ($N = 1154$, $M = 1153$). (b) Using the GHWT best basis ($\tau = 0.9$), we generate relative error and SNR curves as we did for the mutilated Gaussian on the Minnesota road network.



(a) (b)

FIGURE 8.4. (a) A noisy version of the Toronto traffic data from Figure 7.5 with SNR 7.00 dB ($N = 2202$, $M = 4877$). (b) Relative error and SNR curves for the HGLET (L) best basis ($\tau = 0.3$).

As we see from Figures 8.2b, 8.3b, and 8.4b, the peak SNR occurs soon after the relative error starts to drop quickly as the threshold decreases toward zero. The intuition here is simple: we want to retain the coefficients that capture detail in the signal while thresholding those which capture the noise (and ultimately lead to a relative reconstruction error of zero and the original SNR value of the noisy signal). Empirically, we have found the following elbow detection scheme to work well for determining a threshold, which we illustrate in Figure 8.5 using the GHWT best basis relative error curve for the noisy mutilated Gaussian (Fig. 8.2b); the rescaling is merely for illustrative purposes, and it does not affect the index of the point returned. First, we draw a line (shown in red) from the first point on the relative error curve to the last. We then find the point on the curve with the largest orthogonal distance from this line. We repeat the process a second time, drawing a line from this point to the first point (shown in green) and finding the point on the relative error curve with the greatest orthogonal distance from that line. This point on the relative error curve (again shown in green) is the threshold that we use for denoising. The reason why we iterate this elbow detection scheme twice is because we seek a threshold that is lower than that at which the relative error curve starts to drop rapidly towards zero. We do not iterate a third time because doing so would drive the threshold too low, causing too much of the noise to be retained.



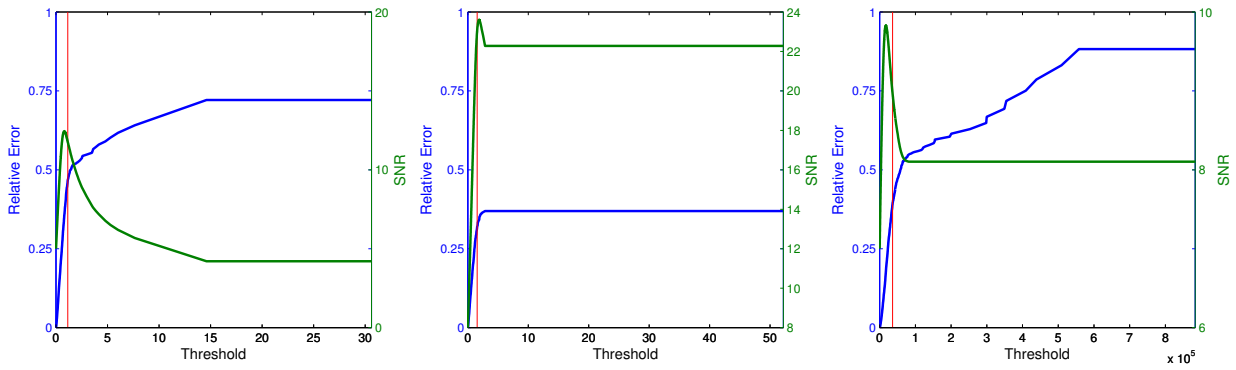
(a) (b)

FIGURE 8.5. (a) An illustration of the method that we use to determine a threshold from the relative error curve. The curve seen here is a rescaled version of the relative error curve for the mutilated Gaussian (Figure 8.2b). (b) A zoomed-in version of the figure.

At this point we now formally describe our denoising experiments. We consider three signals: a mutilated Gaussian on the Minnesota road network, thickness data on the dendritic tree, and the traffic volume data for Toronto. For the Minnesota road network we use inverse Euclidean distances as the edge weights, since the coordinates of the nodes were explicitly used to generate the signal, whereas for the other two data sets we simply use binary adjacency matrices. We add noise to these three signals such that the signal-to-noise ratios are 5.00 dB for the mutilated Gaussian, 8.00 dB for the dendritic tree, and 7.00 dB for the Toronto traffic data. (Lower SNR values for each of the signals were investigated, but in such cases it was found that the noise obscured the signal and denoising was infeasible.) We recursively partition the graphs using Fiedler vectors of L_{rw}, as described in Chapter 3, and we analyze the noisy signals using each of the three HGLET variations (L, L_{rw}, and L_{sym}) and the GHWT. Using the minimal relative error best basis algorithm (Algorithm 7), we compute the HGLET (L) best basis, the GHWT best basis, and the hybrid best basis selected from the three HGLET dictionaries and the GHWT coarse-to-fine dictionary. For comparison, we also consider the Haar basis, the Walsh basis (i.e., level $j = 0$ of the GHWT coarse-to-fine dictionary),

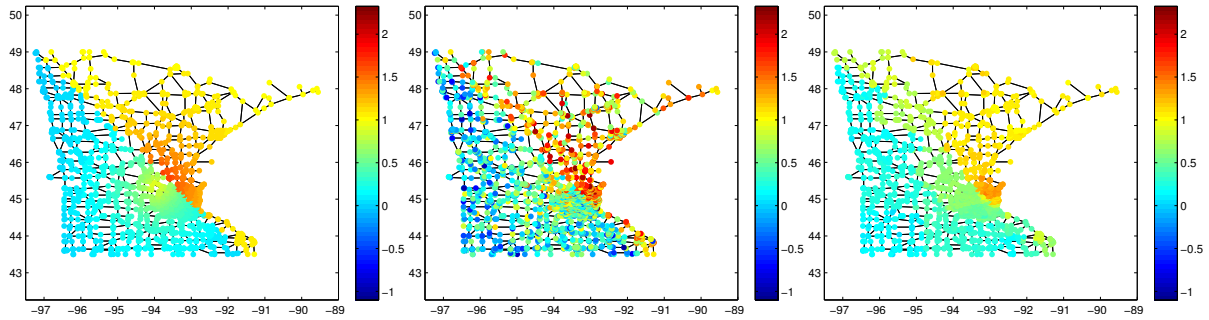
and the eigenvectors of the unnormalized Laplacian L. For each basis we generate a relative error curve, and from this curve we determine the threshold using the aforementioned elbow detection scheme. We soft-threshold the coefficients (leaving coefficients with $l = 0$ unchanged), reconstruct the signal, and compute the SNR.

Figure 8.6 shows the results of our threshold selection method for the previously shown relative error and SNR curves of the noisy mutilated Gaussian, dendritic tree, and Toronto traffic data sets. As in Figures 8.2b, 8.3b, and 8.4b, these curves correspond to use of the GHWT best basis for the mutilated Gaussian and dendritic tree data and the HGLET (L) best basis for the Toronto traffic data. Side-by-side comparisons of original, noisy, and denoised versions of these data sets can be found in Figures 8.7, 8.8, and 8.9. A summary of the full results from this experiment can be found in Table 8.1.



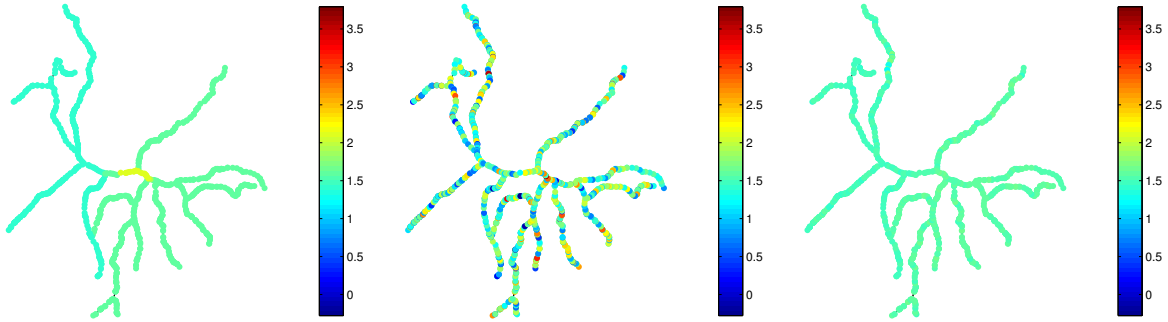
(a) (b) (c)

FIGURE 8.6. The vertical red lines indicate the thresholds selected based on relative error curves for the noisy (a) mutilated Gaussian on the Minnesota road network, (b) dendritic tree thickness data, and (c) Toronto traffic volume data. The relative error and SNR curves are the same as those in Figures 8.2b, 8.3b, and 8.4b, respectively.



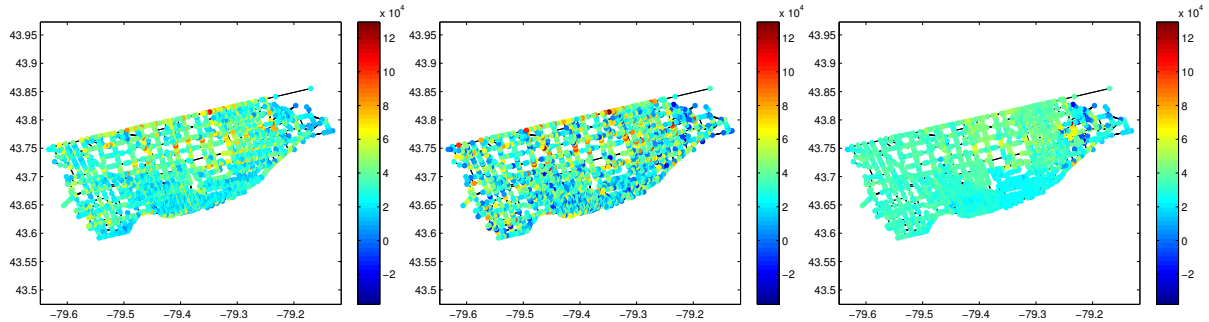
(a) Original (b) 5.00 dB (c) 11.76 dB

FIGURE 8.7. The (a) original, (b) noisy, and (c) denoised versions of the mutilated Gaussian on the Minnesota road network. The GHWT best basis ($\tau = 0.9$) was used.



(a) Original (b) 8.00 dB (c) 23.03 dB

FIGURE 8.8. The (a) original, (b) noisy, and (c) denoised versions of the thickness data on the dendritic tree. This denoising was done using the GHWT best basis ($\tau = 0.9$).



(a) Original (b) 7.00 dB (c) 8.96 dB

FIGURE 8.9. The (a) original, (b) noisy, and (c) denoised versions of the traffic volume data on the Toronto road network. The HGLET (L) best basis ($\tau = 0.3$) was used here.

	MN Mutilated Gaussian (5.00 dB)		Dendritic Tree (8.00 dB)		Toronto (7.00 dB)	
HGLET (L) Best Basis	10.16 dB	($\tau = 0.5$)	20.85 dB	($\tau = 0.1$)	**8.96 dB**	($\tau = 0.3$)
Laplacian Eigenvectors (L)	11.96 dB		22.56 dB		8.26 dB	
GHWT Best Basis	11.76 dB	($\tau = 0.9$)	**23.03 dB**	($\tau = 0.9$)	8.27 dB	($\tau = 1.0$)
Haar	**12.34 dB**		22.68 dB		8.29 dB	
Hybrid Best Basis	10.59 dB	($\tau = 0.5$)	22.29 dB	($\tau = 0.3$)	8.82 dB	($\tau = 0.3$)
Walsh	10.87 dB		21.57 dB		8.14 dB	

TABLE 8.1. Denoising results for the noisy versions of the mutilated Gaussian (Fig. 8.1b), dendritic tree thickness data (Fig. 8.3a), and traffic volume data for Toronto (Fig. 8.4a).

These experimental results demonstrate the effectiveness of the HGLET and GHWT, along with the best basis algorithms, for denoising signals on graphs. Perhaps a surprising result is the strong performance of the GHWT best basis for the mutilated Gaussian, since the signal is smooth aside from its jump discontinuity but the basis vectors are piecewise constant. One contributing factor is that the GHWT best basis comes from the fine-to-coarse dictionary, as seen in Figure 8.2a. Unlike the coarse-to-fine and HGLET dictionaries, the fine-to-coarse dictionary contains choosable bases for which basis vectors from different levels have overlapping supports. Thus, global basis vectors can capture the general characteristics of the signal while localized basis vectors contribute the finer scale details. In fact, for all three signals the GHWT best basis originated from the fine-to-coarse dictionary.

Further research on the use of these transforms for denoising is needed. Topics requiring investigation include the choice of a cost functional for denoising and the ideal basis or bases to consider. In the spirit of the best basis algorithm, it may be possible to devise a criterion by which to compare the denoising performances of various bases and choose the best one.

CHAPTER 9

Simultaneous Segmentation, Denoising, and Compression of 1-D Signals

9.1. Methods

Having discussed uses of the HGLET and GHWT for signals on graphs, we now apply them to classical signals. Specifically, in this chapter we discuss an iterative procedure involving the HGLET and the best basis algorithm to simultaneously segment and denoise classical 1-dimensional signals, which we treat as signals on unweighted path graphs. Our method is in the same spirit as [**70**], but in place of the polyharmonic local sine transform we use our HGLET. Doing so affords us more flexibility in our segmentation, as we no longer have to work within a dyadic constraint on the segment lengths. The objective here is three-fold: 1) to divide the signal into segments of similar characteristics; 2) to reduce the noise in the signal; and 3) to achieve better approximation and compression of the underlying signal.

Our iterative algorithm involves four components, which we repeat until convergence. The first step is to recursively partition the graph. In Chapter 3 we presented a means of generating a recursive partitioning using Fiedler vectors of graph Laplacians, since the Fiedler vectors of L and L_{rw} are approximate minimizers of the RatioCut and Normalized Cut bipartitioning criteria, respectively. For general graphs these minimization problems are NP-hard, but for the special case of path graphs Eqs. (3.1) simplify. For a path of length N, we can search over the $N-1$ possible bipartitions and directly find the minimizer. Hence, for these experiments we partition by computing the minimizer of the Normalized Cut problem,

$$\underset{n\in[1,N)}{\arg\min} \frac{W_{n,n+1}}{\sum_{i=1}^{n} d_i} + \frac{W_{n,n+1}}{\sum_{i=n+1}^{N} d_i},$$

instead of using the Fiedler vector. We associate the path graph G with the set of integers $\{1, 2, \ldots, N\}$, and since the ordering of the vertices is preserved in our partitioning, we can further associate each subgraph G_k^j with a subset of consecutive integers $I_k^j \subseteq \{1, 2, \ldots, N\}$.

The next step in our method is to analyze the graph signal using the HGLET with the eigenvectors of each of the three Laplacians, which are the variants of the DCTs as we explained earlier. Again, we exploit the simplicity of the graph: instead of performing the HGLET by computing the Laplacian eigenvectors we use the DCT-II (for L) and the DCT-I (for L_{rw} and L_{sym}). Using the DCT to perform the HGLET for these path graphs speeds up the transform considerably, and it also emphasizes the connection between the HGLET and the block DCT's.

The third step is to select a best basis from the three HGLET dictionaries using Algorithm 6. As our cost functional we use the minimum description length (MDL) criterion [**33**, **59**]. Unlike an ℓ^p norm or quasinorm, the MDL not only takes into account the cost of the expansion coefficients but also the cost of the model parameters, and it chooses between two or more models for input data by considering their total costs in terms of bits. In our experiments, these model parameters that the MDL determines are: (i) the segmentation configuration of the signal (i.e., the set of disjoint intervals such that $\cup_i I_{k_i}^{j_i} = [1, N]$, which we quantize via the levels list description method) and (ii) a flag to specify the HGLET variation selected for each segment. For each model we compute the number of bits needed to encode the model parameters and quantized expansion coefficients of the input data that are necessary to fit the model to the data. In accordance with the MDL, we then select the model that best captures the nature of the underlying signal in the input data using the fewest bits. Thus, the MDL provides an intelligible, objective, parameter-free means of choosing between competing models. By using the MDL cost functional to perform the best basis search, we are searching for the segmentation whose structure is choosable from the current partitioning tree that allows us to most efficiently represent the signal. Furthermore, the resulting quantized coefficients can be used to denoise the signal since the MDL automatically selects the precision and threshold that best capture the noise-free portion of the signal. We discuss the specifics of the MDL for our method in §9.2.

The MDL-guided best basis search yields two outputs: a segmentation of the signal and the corresponding set of quantized expansion coefficients. The fourth component of our iterative method is

to modify the edge weights of the graph using this segmentation from the best basis algorithm. The purpose of doing so is to encourage edges between regions of similar characteristics to be preserved in the next iteration and to encourage those edges between regions of different characteristics to be cut. We have tried several different means of doing so, and the method we have found most effective in our experiments is to cut the edges that are 5% and 10% to the left and to the right of each partition in the best basis. That is, suppose the first and last nodes in a segment of the signal are n_1 and n_2. We set $\Delta_5 = \max\{1, [0.05(n_2 - n_1)]\}$, and if $n_1 \neq 1$ then we set the weight of the edge between nodes $n_1+\Delta_5$ and $n_1+\Delta_5+1$ to zero. If $n_2 \neq N$ then we set the weight of the edge between nodes $n_2 - \Delta_5$ and $n_2 - \Delta_5 + 1$ to zero. We proceed likewise for $\Delta_{10} = \max\{1, [0.1(n_2 - n_1)]\}$.

We then iterate this process. We generate a new recursive partitioning of the signal, which will differ from the previous recursive partitioning due to the modified edge weights. It is important to mention that we generate this recursive partitioning in such a way that the basis selected in the previous iteration is choosable from the dictionaries in the subsequent iteration; our scheme for doing so is illustrated in Figure 9.1. Specifically, we start with the previous best basis segmentation and use clustering and partitioning to form a full hierarchical tree; if the NCut has more than one minimum in a particular segment of the signal, we cut the edge of the leftmost minimum. We then analyze the signal again using the three HGLET variations, although we treat the graph as being connected and unweighted. This is because the purpose of modifying the edge weights is to influence the partitioning, and we want to preserve the relationship between the HGLET on a path graph and the block DCTs. As the recursive partitioning of the signal is different, the expansion coefficients will be different as well. We then find a new best basis and corresponding segmentation, and we modify the edge weights as before; when modifying edge weights, we always start with a connected, unweighted path graph, and thus the cuts from previous iterations are no longer in effect. We repeat this process until it converges to a particular basis, which gives us both a segmentation of the signal and a set of quantized coefficients. Empirically, we have observed that convergence occurs between 6 and 15 iterations. We denoise the signal by reconstructing with these quantized and thresholded HGLET coefficients.

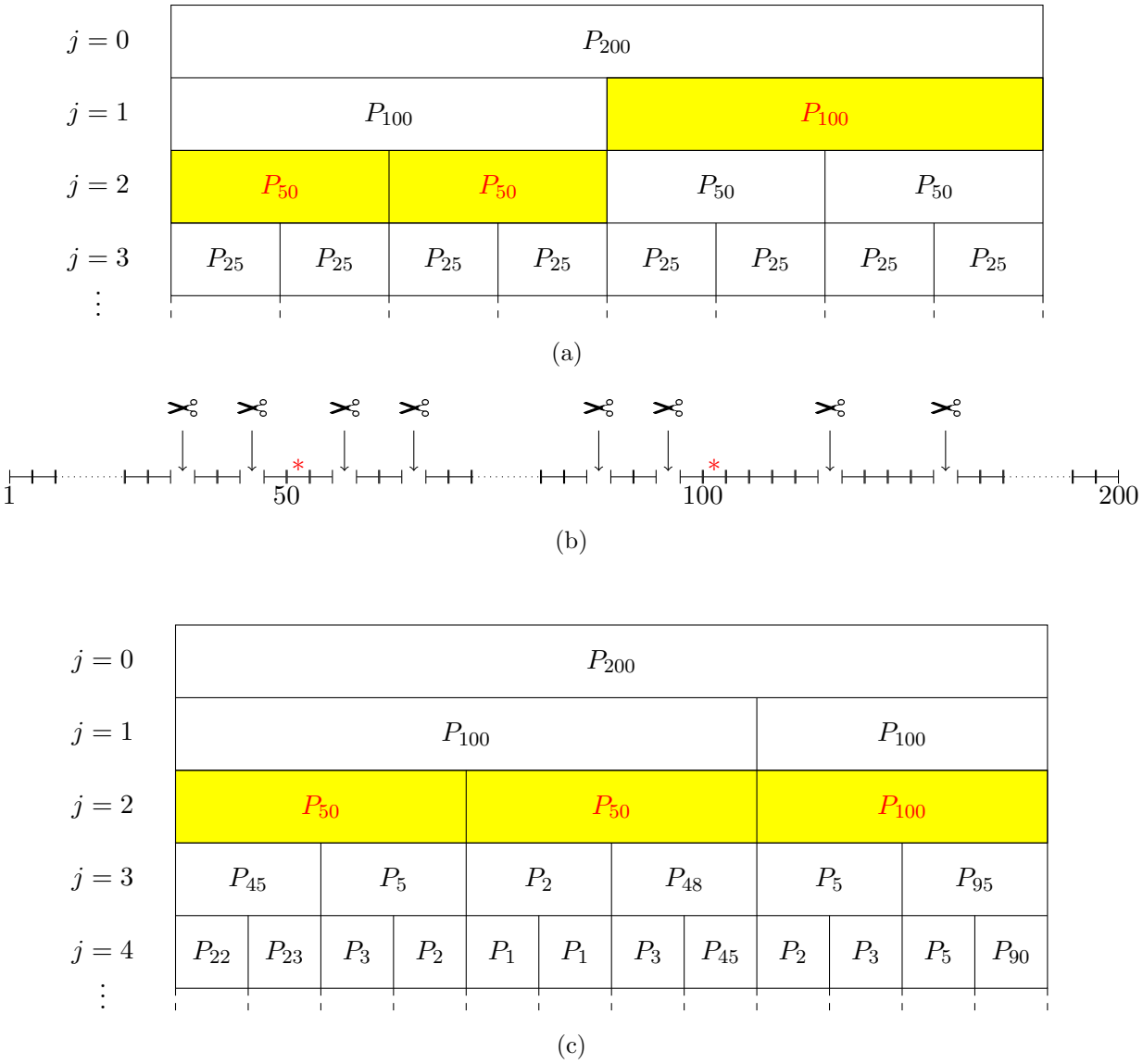


FIGURE 9.1. An illustration of our edge weight modification scheme and also our method of generating a new hierarchical tree for the graph. (a) A recursive partitioning of a path graph of length 200. The highlighted blocks illustrate the segmentation chosen by the best basis algorithm. (b) The path graph after cutting the edges that are 5% and 10% to the left and to the right of each partition in the best basis; these cut edges are indicated by the arrows with scissors. Meanwhile, the red asterisks in this figure denote the breakpoints in the current best basis segmentation. (c) The new hierarchical tree for the graph. In order to ensure that the previous iteration's best basis segmentation is choosable, we use it as our starting point, and we cluster segments as we move upwards and partition them as we move downwards. (Although this hierarchical tree does not satisfy the fourth requirement from Chapter 3 – "Each region on level $j < j_{\max}$ containing two or more nodes is partitioned into exactly two regions on level $j + 1$" (page 51) – our transforms will still work.)

Before presenting experimental results, we first describe the MDL.

9.2. The Minimum Description Length (MDL)

The principle behind the minimum description length criterion is simple: given two or more models for representing a signal, choose the model that best captures the true nature of the signal using the least amount of bits. Of course, when quantizing the signal as a stream of bits we need to specify the model we are using, its parameters, and the expansion coefficients of the signal. A more comprehensive description of the MDL can be found in [**58**, **64**, **70**]; here, we focus on applying the MDL principle to our specific scenario.

Consider a signal $\boldsymbol{f}$ of length N (which may be a portion of a longer signal). Our representation of $\boldsymbol{f}$ will be a concatenation of representations of signals $\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_n$, where $\boldsymbol{f} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_n]$. For each $\boldsymbol{f}_i$, we have the following costs:

(a) specifying the transform used for $\boldsymbol{f}_i$ (i.e., HGLET L, HGLET L_{rw}, or HGLET L_{sym})
- 2 bits $= \lceil \log_2(\#\text{ of transforms considered}) \rceil$

(b) specifying the levels list entry that corresponds to $\boldsymbol{f}_i$
- $\lceil \log_2(j_{\max} + 1) \rceil$ bits

In addition, we have the following costs that pertain to the signal as a whole (which we explain below):

(1) specifying the quantization precision $\delta = 2^{-q}$, where $q \in \{1, 2, 3, 4, 5\}$ is the integer which minimizes the MDL cost

(2) the quantization of the expansion coefficients corresponding to the specified transform

(3) the quantization of the noise

(4) the codelength of $\boldsymbol{f}$ given the above parameters.

The MDL of a representation of $\boldsymbol{f}$ is the sum of costs (a), (b), and (1)-(4). Observe that costs (a) and (b) are associated with specifying the model, whereas costs (1)-(3) are for storing the numbers that describe the signal. The final "cost" (4) relates to the expected accuracy of our model and its associated parameters in representing $\boldsymbol{f}$.

Of course, the magnitude of the signal should be taken into account. Certainly, we would not expect to use the same absolute precision to quantize a signal of mean 10^9 and a signal of mean 10^{-3}. Therefore, prior to computing the MDL we rescale the signal and its expansion coefficients by a factor of $\sqrt{N}/\|\boldsymbol{f}\|_2$. The 2-norm of the rescaled signal is $\sqrt{N}$; i.e., the same as the norm of the constant signal $\mathbf{1} \in \mathbb{R}^N$, which is the nonzero signal of length N that minimizes the MDL cost. Thus, this rescaling takes into account both the norm and the length of the signal. As this is done prior to the MDL computation and the factor is the same regardless of the chosen precision $\delta = 2^{-q}$, we do not include this rescaling factor in our MDL cost. On a related note, although restricting the precision to be $\leq 2^{-5}$ may seem a bit crude at first glance, keep in mind that this precision is for the expansion coefficients, not the signal values themselves. Since a good basis for the signal concentrates the majority of its energy into a small number of coefficients, a precision of 2^{-5} is not so crude after all. (The user may specify a higher precision, e.g. 2^{-7} or 2^{-8}, although this will increase the computational cost of the algorithm.)

Addressing the model specification costs for each segment $\boldsymbol{f}_i$ of the signal, MDL Cost (a) is simply 2 bits because we must specify which of the three HGLET variations is used. MDL Cost (b) is the cost of each entry in the levels list description of the basis, which is of the form $(j_1, j_2, \ldots, j_n)$, with each $j_i \in [0, j_{\max}]$. Hence, the levels list cost associated with each segment $\boldsymbol{f}_i$ is $\lceil \log_2(j_{\max} + 1) \rceil$.

Moving on to the quantization costs for the full signal, the first cost is simply

$$\text{MDL Cost (1)} \ = \log_2 \delta^{-1} = q,$$

which is the codelength required to specify the precision $\delta = 2^{-q}$. Before computing costs (2)-(4), we first need to quantize the signal, which we do using a uniform quantizer [**73**, §9.4]. This involves three key steps: scaling, rounding, and thresholding. (Recall that we have already rescaled the coefficients by the factor $\sqrt{N}/\|\boldsymbol{f}\|_2$.) We divide all of the coefficients by 2δ, and then we round to the nearest integer. Thus, for each coefficient $c^j_{k,l}$ we have that

$$\left| \frac{c^j_{k,l}}{2\delta} - \left[\frac{c^j_{k,l}}{2\delta} \right] \right| \leq 0.5 \quad \Leftrightarrow \quad \left| c^j_{k,l} - 2\delta \left[\frac{c^j_{k,l}}{2\delta} \right] \right| \leq \delta.$$

For denoising, we use the MDL to select a threshold $T \in \mathbb{N}$ and we set all of the scaled and rounded coefficients which are less than or equal to T to zero, except those coefficients $c^j_{k,l}$ with $l = 0$. Here, we choose the $T \in \{0, 1, 2\}$ which minimizes the sum of costs (2)-(4); we limit our search for T to these values so as not to increase the computational cost.

We now have a set of quantized integer coefficients, which we will refer to as $\{\tilde{c}^j_{k,l}\}$, and we can continue the MDL calculation. For MDL Cost (2), we use an upper bound on the Huffman codelength for the sequence of integers [**17**, §5.6]:

$$\text{MDL Cost (2)} \ = N(H(\boldsymbol{p}) + 1),$$

where $\boldsymbol{p}$ is the probability mass function for the integers and H is the *Shannon entropy* [**17**, §2.1],

$$H(\boldsymbol{p}) := -\sum_n \boldsymbol{p}(n) \log_2 \boldsymbol{p}(n). \tag{9.1}$$

Next, we compute the maximum likelihood estimate of the noise,

$$\hat{\sigma}^2 = \begin{cases} \frac{1}{N} \sum_{l=0}^{N-1} \left(c^j_{k,l} - 2\delta \tilde{c}^j_{k,l} \right)^2 & \text{for HGLET with } L \text{ and } L_{\text{sym}} \\ \frac{1}{N} \left\| \sum_{l=0}^{N-1} \boldsymbol{\phi}^j_{k,l} (c^j_{k,l} - 2\delta \tilde{c}^j_{k,l}) \right\|_2^2 & \text{for HGLET with } L_{\text{rw}}, \end{cases}$$

As the HGLET with L_{rw} basis is not orthonormal with respect to the standard inner product, the computation of the MLE requires reconstructing the signal. For numerical purposes, we impose that $\hat{\sigma}^2 \geq 2^{-52} = \texttt{eps}$ (machine precision), since we will later need to take the logarithm of it. MDL Cost (3) is the codelength of the noise, quantized with precision δ:

$$\text{MDL Cost (3)} \ = L^*([\hat{\sigma}^2/2\delta]).$$

Here, the function L^* gives the codelength for any integer $z \in \mathbb{Z}$ and is defined as [**58**, **70**]

$$\log^* |z| := \log |z| + \log\log |z| + \cdots = \sum_{k>0} \max \left(\log^{(k)} |z|, 0 \right)$$

$$L^*(z) := \begin{cases} 1 & \text{if } z = 0, \\ \log^* |z| + \log 4c_0 & \text{otherwise,} \end{cases}$$

where $c_0 \approx 2.865064$ is derived so equality holds in the Kraft inequality: $\sum_{z=-\infty}^{\infty} 2^{-L^*(z)} \leq 1$.

The final MDL cost is computed as

$$\text{MDL Cost (4)} = \frac{N}{2} \log_2(2\pi e \hat{\sigma}^2).$$

This is a lower bound on the codelength of $\boldsymbol{f}$ given the model parameters: the transform used, the levels list description, the precision δ, the quantized and thresholded coefficients $\{\tilde{c}^j_{k,l}\}$, and the quantized noise $[\hat{\sigma}^2/2\delta]$. This "cost" is often negative in our experiments, and it can even make the MDL negative. This occurs when $\hat{\sigma}^2$ is small (specifically, when $\hat{\sigma}^2 < \frac{1}{2\pi e}$), which means that the signal is easily quantized with high precision. Obviously, it is not meaningful to say that it requires a negative number of bits to represent a signal. However, MDL Cost (4) is important because it rewards representations that accurately reconstruct the signal and punishes those that do not. Effectively, it is our "quality assurance policy," so to speak, in that it forces the MDL finds a balance between efficiency of storage and accuracy of reconstruction.

Although the MDL is considered "parameter-free," as with all numerical computing there are parameters that are hard-coded into its implementation, as seen above. For example, we specify that the precision is $\delta = 2^{-q}$ for some integer $q \in \{1, 2, 3, 4, 5\}$ and that the threshold $T \in \{0, 1, 2\}$.

It is often the case that the basis returned by the MDL-guided best basis algorithm has partitions that are very close to one another. This is simply a consequence of numerics: even for a segment on which a given signal is constant, it may be more efficient to store two easily discretized coefficients than to store one larger coefficient which is more difficult to discretize. As a very simple example, it is easier to quantize $(1, 1)$ than to quantize $(\sqrt{2}, 0)$. Therefore, in our display of the reconstructed signal we find each segment shorter than $[N/50]$ and absorb it into its neighbor segments. This factor was chosen because for the signals analyzed in this work, it seemed reasonable to impose that there are no more than 50 total segments; one could certainly choose a different constant.

As a final remark about the MDL, we note that the MDL does not satisfy the conditions (6.3) from Proposition 6.1. Using ancestry terminology, this occurs when two pairs of child regions may each have a higher MDL cost than their parent region, but collectively the four child regions have a lower MDL cost than the union of the two parent regions. We can still use the MDL as the best basis cost functional, but we cannot guarantee that the resulting basis has the lowest MDL

cost out of all the choosable bases. As a consequence, it is sometimes the case that the MDL cost will increase from one iteration to the next, even though the previous basis is choosable from the current dictionaries. To circumvent this, our algorithm reverts to the state from the previous iteration, generates a slightly different hierarchical tree, and continues until convergence. With this modification in place, the costs of the iterations is a nonincreasing sequence.

9.3. Experimental Results

Here we show some experimental results using our method for simultaneous segmentation, denoising, and compression. Although we do not actually generate a compressed bitstream for representing the signals, we determine the precision $\delta = 2^{-q}$ and threshold T that enable one to easily generate said bitstream.

The first signal we consider is "Msignal" from the WaveLab software package [**26**], which we display in Figure 9.2a. The signal is of length $N = 256$, with the left side being piecewise smooth and noise-free and the right side having what appears to be significant noise. This presents a challenging problem: we obviously want to differentiate between the noise-free and noisy components, and we also want to accurately reconstruct the noise-free portion. Our algorithm converges after 9 iterations and the result is displayed in Figure 9.2b, with segments represented by the HGLET with L shown in blue and those represented by the HGLET with L_{rw} in red; for this signal, no segments are represented by the HGLET with L_{sym}. The quantization precision is $\delta = 2^{-5}$, and the quantized integer threshold is $T = 2$. Our method successfully identifies three constant regions in the signal, and it partitions the remainder of the noise-free portion into two roughly symmetric piecewise smooth segments. The relative error for the noise-free portion is 1.11%. Meanwhile, the noisy portion of the signal is kept intact and is likewise reproduced almost exactly, with the relative reconstruction error being 1.05%.

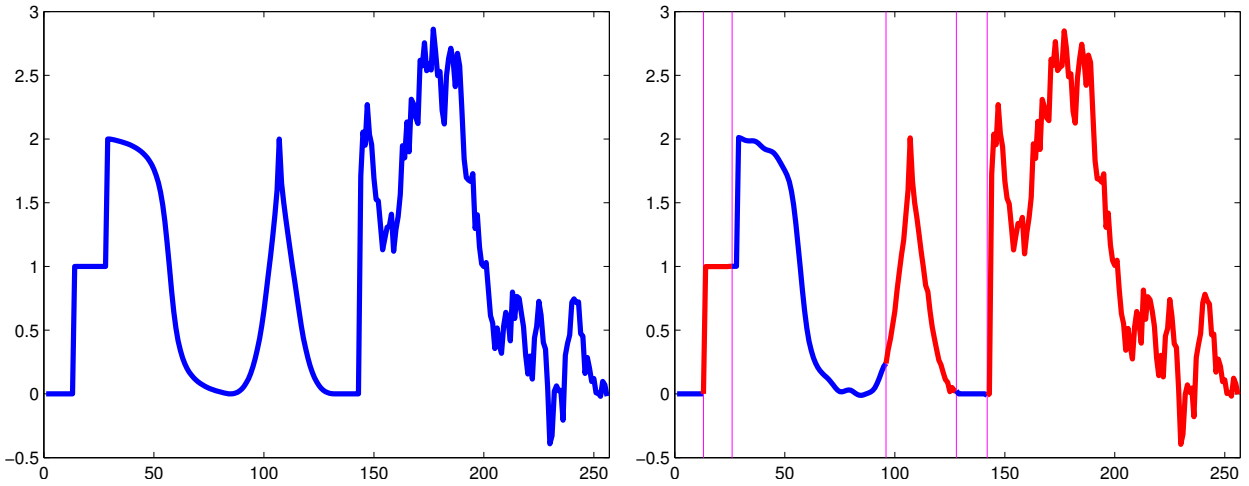
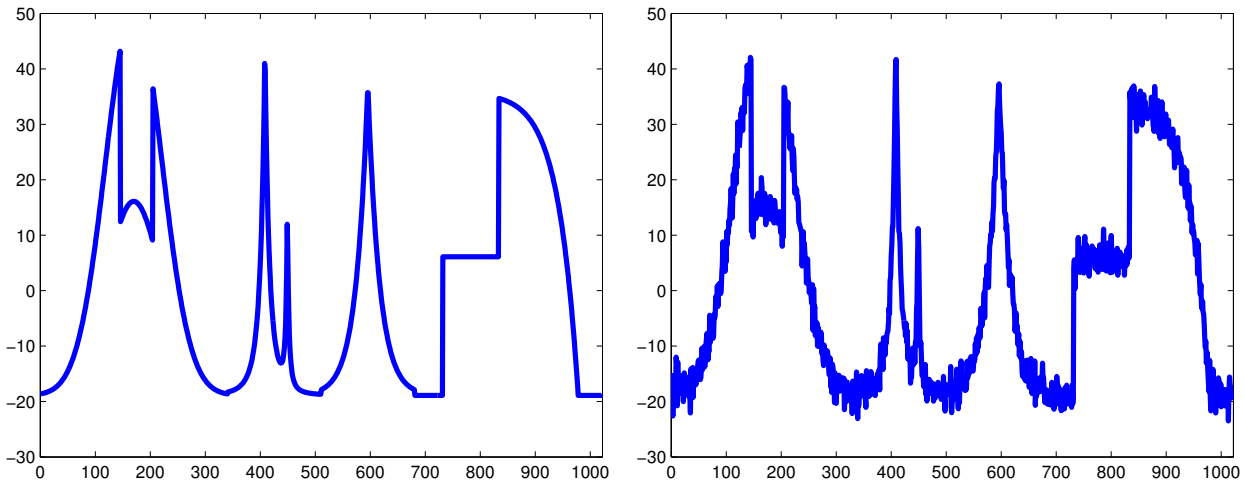


FIGURE 9.2. (a) "Msignal," which has length $N = 256$, and (b) the result of our algorithm. The regions in blue and red are represented by the HGLET with L and HGLET with L_{rw}, respectively.

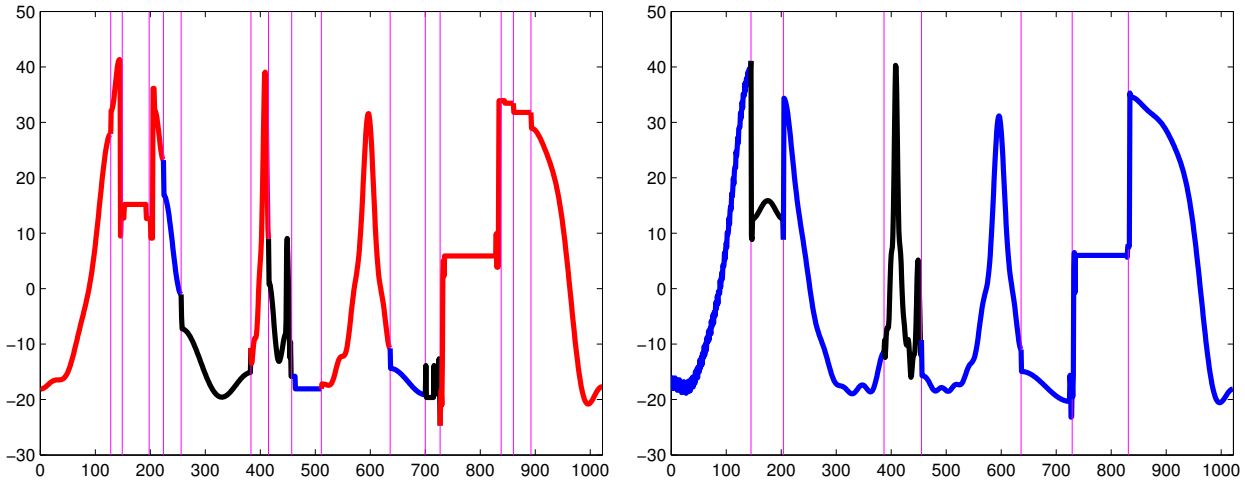
For our second experiment we use the "Piece-Regular" signal, shown in Figure 9.3b, which is also from WaveLab. In order to showcase the versatility of our method, we make the length of the signal $N = 1021$ (which is a prime number). We add Gaussian noise to the signal, bringing its SNR to 20 dB, and we display the noisy signal in Figure 9.3b. To demonstrate that the iterative nature of our method is a necessity, in Figure 9.4a we show the result after the first round. The algorithm has crudely identified some of the regions of different characteristics, but the locations of the partitions are not yet refined. Once more, the regions in blue correspond to the HGLET with L and those in red correspond to the HGLET with L_{rw}. The regions in black are represented by the HGLET with L_{sym}. Figure 9.4b shows the final result after 11 iterations. The selected precision and threshold are $\delta = 2^{-5}$ and $T = 2$, respectively. The SNR of the resulting signal is 23.85 dB, and while this is a somewhat modest increase from the original SNR of 20 dB, qualitatively we see that much of the noise has been removed, although we do observe the Gibbs phenomenon at some of the partitions. For comparison purposes, in Figure 9.5 we show the result of translation-invariant denoising with soft-thresholding ($T = \sqrt{\log N}$) using a Symmlet 8 wavelet, as Coifman and Donoho did in [**8**]. (In order to use their WaveLab code, we extend the clean and noisy signals in Figure 9.3 to length 1024 by repeating their first value and their last two values.) Their method does a better

job of minimizing the Gibbs phenomenon, which is to be expected due to its translation invariance. However, our method does a better job of capturing the sharp transitions and constant segment on the right side of the signal. Furthermore, our method also segments and compresses the signal.



(a) (b)

FIGURE 9.3. (a) The noise-free "Piece-Regular" signal of length $N = 1021$. (b) The noisy signal with an SNR of 20 dB.



(a) (b)

FIGURE 9.4. (a) The result after one iteration of our algorithm. (b) The final result after 11 rounds with an SNR of 23.85 dB.

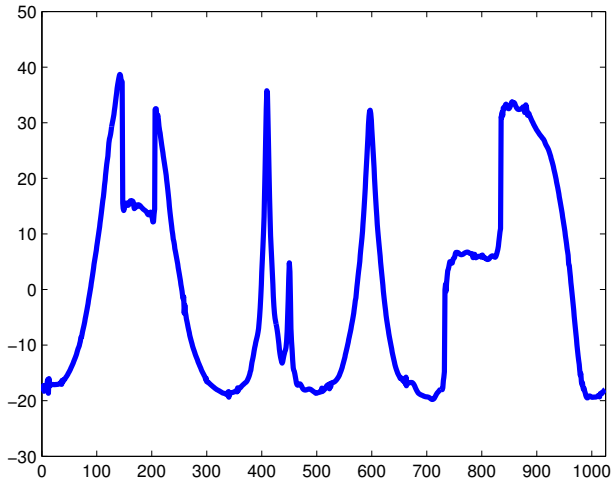
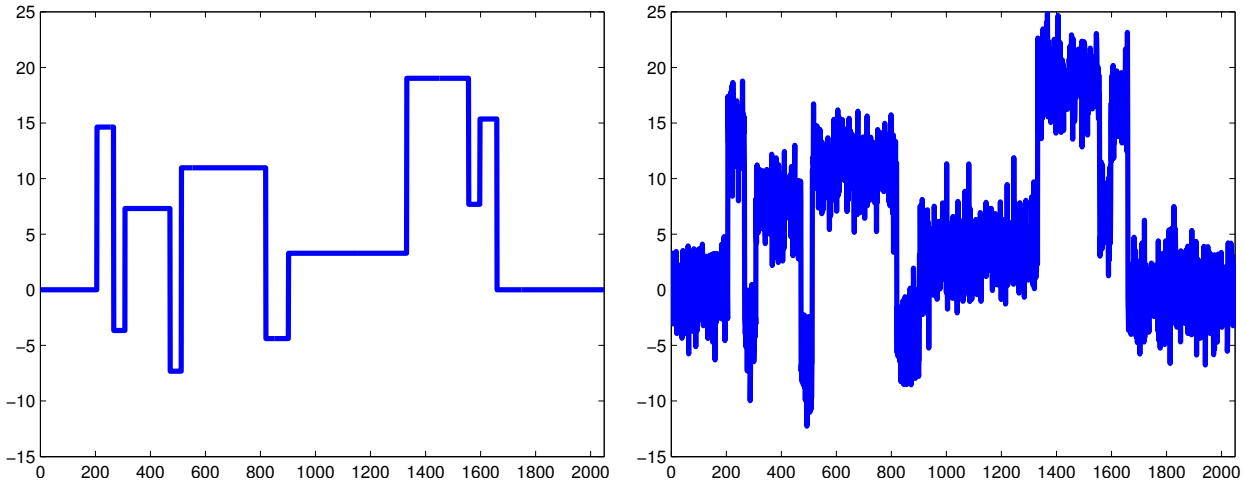


FIGURE 9.5. The "Piece-Regular" signal from Figure 9.3b after translation-invariant denoising with soft-thresholding using the Symmlet 8 wavelet. The threshold is $T = \sqrt{\log N}$ and the SNR of the resulting signal is 24.67 dB.

For our last signal we use a noisy version of the "Blocks" signal[1] ($N = 2048$) which originally appeared in [**27**]. The noise-free signal (Fig. 9.6a) consists of 12 constant segments of varying lengths, and the noisy signal (Fig. 9.6b) has an SNR of 11.95 dB. After 8 iterations, our algorithm converges to the segmented and denoised signal shown in Figure 9.7a. The precision of the representation is $\delta = 2^{-5}$, and the threshold is $T = 2$. As before, blue, red, and black regions correspond to the HGLET with L, L_{rw}, and L_{sym}, respectively[2]. Most of the noise is removed from the signal, and the SNR of the denoised signal is 18.26 dB. Furthermore, most of the 12 regions are identified. Although the region around node 500 is not precisely identified in Figure 9.7a, this is because it comprises only 2% of the signal and has thus been absorbed into its neighbor regions. Indeed, in Figure 9.7b we show the result of our algorithm without absorbing regions shorter than $[N/50]$ in length, and we see that this region is detected. However, we also note that there are a couple undesirable partitions that remain. As with the Piece-Regular signal, we compare our results to translation-invariant denoising with soft-thresholding ($T = \sqrt{\log_2 N}$) using the Symmlet 8 wavelet in Figure 9.8. Although this yields a higher SNR, our method does a much better job of capturing

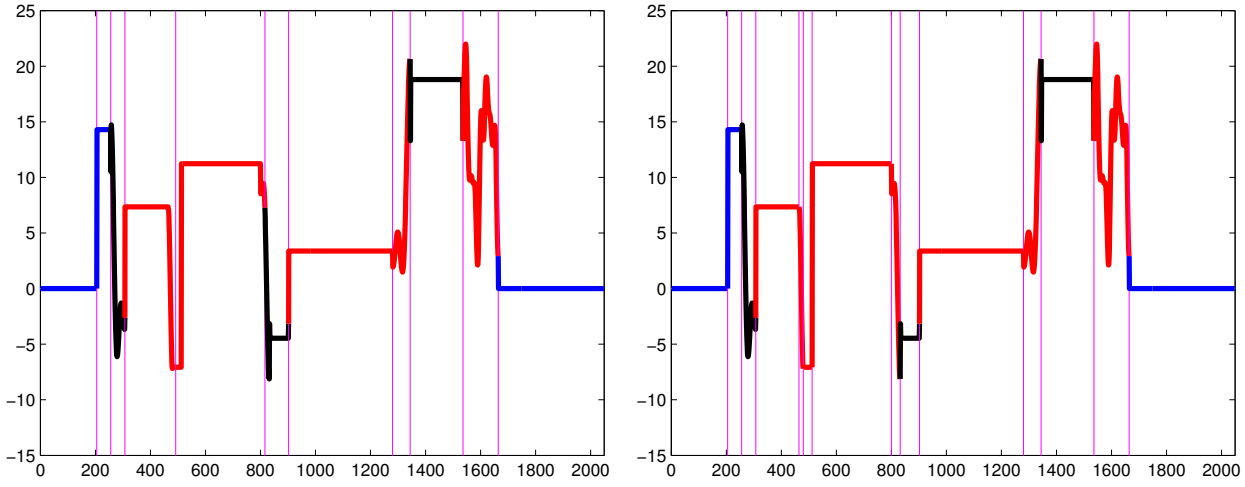
[1]The clean and noisy "Blocks" signals are available from `ftp://ftp.sas.com/pub/neural/dojo/dojo.html`.
[2]Although it could be argued that the GHWT is the natural choice of transform for denoising this "Blocks" signal, experimental results showed that it captured too much of the noise and that better results were achieved using the piecewise smooth HGLET variations.

the piecewise-constant nature of the signal. Future research should investigate the generalization of "cycle spinning," as Coifman calls it, to our own algorithm.



(a) (b)

FIGURE 9.6. (a) The noise-free "Blocks" signal from [**27**]. (b) The noisy "Blocks" signal that we use for our experiment, which has SNR 11.95 dB.



(a) (b)

FIGURE 9.7. (a) The segmented and denoised signal with SNR 18.26 dB. (b) The same result, but here we do not absorb regions of length less than $[N/50]$ into their neighbor regions.

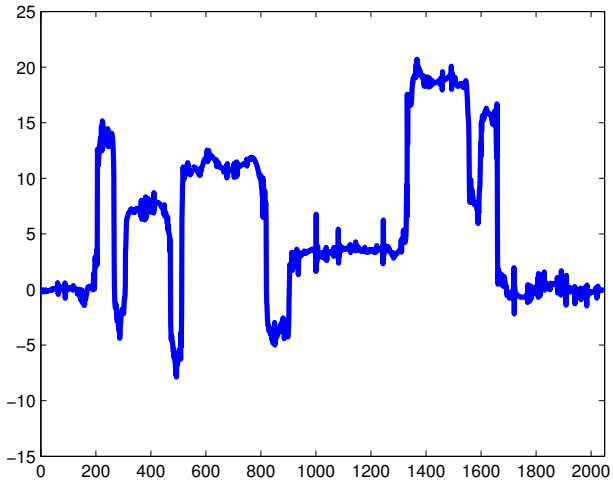


FIGURE 9.8. The "Blocks" signal from Figure 9.6b after translation-invariant denoising with soft-thresholding using the Symmlet 8 wavelet. The threshold is $T = \sqrt{\log N}$ and the SNR of the resulting signal is 19.50 dB.

We conduct one more experiment using the noisy blocks signal in which we supply the best segmentation of the signal to our algorithm a priori. By "best segmentation" we mean that each segment corresponds to a piecewise-constant portion of the original signal; this segmentation is displayed in Figure 9.9a. Our algorithm never strays from this segmentation, as it is unable to find a basis with lower MDL cost, and it returns this same segmentation. The denoised signal is shown in Figure 9.9b, and its SNR is 33.13 dB. Our algorithm selects precision $\delta = 2^{-5}$ and threshold $T = 2$, and it correctly represents each segment as having a constant value.

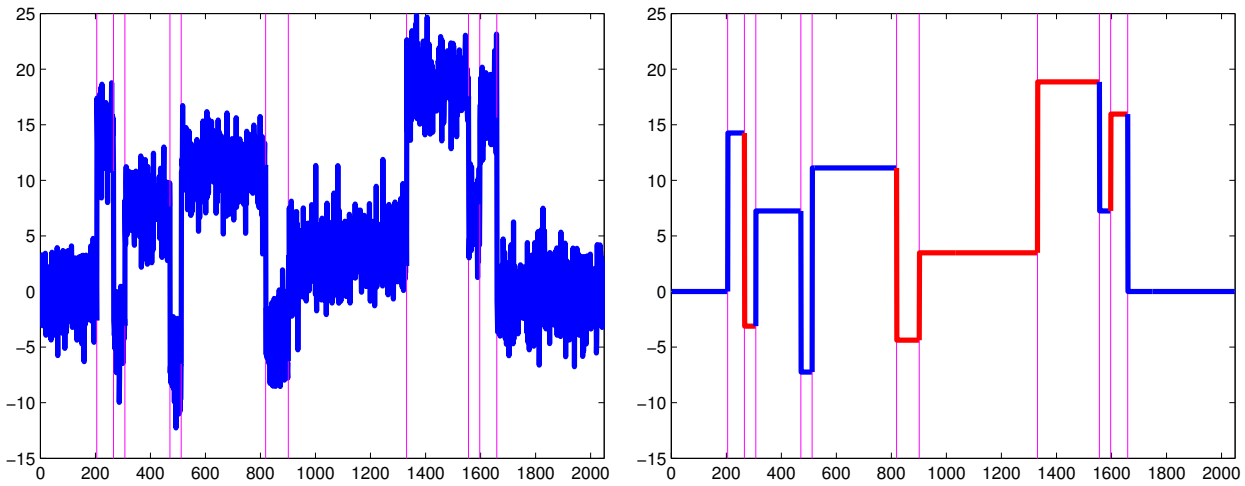


FIGURE 9.9. (a) The noisy "Blocks" signal and the segmentation that is supplied to our algorithm. (b) The denoised signal with an SNR of 33.13 dB.

9.4. Summary

We presented an iterative method which uses the HGLET variations and the best basis algorithm to simultaneously segment, denoise, and compress 1-dimensional signals. We reviewed the minimum description length (MDL) principle, and we detailed how we apply it to this problem. Our method requires no user-specified parameters, and it determines the number of segments into which the signal should be partitioned. We demonstrated the effectiveness of our method by presenting experimental results for three signals, each with unique qualities. Subjects of further research include theoretical guarantees and generalizations of this method to more general settings, such as 2-dimensional images or matrices. It may also be possible to use these tools and strategies to construct and/or partition graphs.

CHAPTER 10

Matrix Data Analysis

10.1. Methods

As another application of our graph-based transforms to classical problems, we now discuss the use of the HGLET and GHWT for matrix data analysis. We do this by constructing a hierarchical tree for the rows and selecting a basis, constructing another hierarchical tree for the columns and selecting a basis, and then taking the tensor product of the row and column bases. Using graph techniques allows us to account for the underlying structure of the matrix, thereby enabling better analysis and characterization of the data. This work is related to the tensor Haar-like bases developed by Coifman and Gavish [**10**]. However, whereas they use the Haar bases for the rows and columns, our bases are selected by the best basis algorithm and therefore are tailored to the data.

The main principle here is that the rows and columns of the matrix are interrelated, and thus the rows of the matrix can tell us about the underlying structure of the columns, and vice versa. By considering the interplay between the rows and columns, we can achieve better results than if we treat them separately. A common example of matrix data which exhibits such structure is a term-document matrix. Here, the rows correspond to words and the columns correspond to documents, and matrix entry A_{ij} is typically the (relative) frequency with which word i appears in document j. In the case that the documents are journal publications from different fields, we would expect the prevalence and absence of certain words to be realtively consistent across documents from within each field. For example, if we consider the row corresponding to the word "ancestor" then we expect that it will appear often in columns corresponding to Anthropology & Archeology papers and rarely in columns corresponding to Astronomy & Space Sciences papers. Conversely, if we consider a column corresponding to an Anthropology & Archeology paper, the entry in the row

corresponding to "ancestor" is more likely to be nonzero than the entry in the row corresponding to "meteorite" is.

In order to discover the structure of the matrix and generate the hiearchical row and column trees needed by the GHWT, we use the bipartite spectral graph partitioning method of Dhillon [**24**]. Given a data matrix A which is $N_R \times N_C$, Dhillon views the rows and columns as the two sets of nodes in a bipartite graph, where matrix entry A_{ij} is the edge weight between the node for row i and the node for column j. He orders the nodes of the graph such that the first N_R nodes correspond to the rows and the last N_C correspond to the columns, and thus the associated $(N_R+N_C)\times(N_R+N_C)$ weight matrix is of the form

$$W = \begin{bmatrix} \mathbf{0} & A \\ A^{\mathsf{T}} & \mathbf{0} \end{bmatrix}.$$

Accordingly, the degree and Laplacian matrices are

$$D = \begin{bmatrix} D_R & \mathbf{0} \\ \mathbf{0} & D_C \end{bmatrix}$$

$$L = \begin{bmatrix} D_R & -A \\ -A^{\mathsf{T}} & D_C \end{bmatrix},$$

where the degree matrix for the rows, D_R, is the diagonal matrix of row sums of A, and likewise, D_C is the diagonal matrix of column sums. Dhillon then computes the Fiedler vector of $L_{\text{rw}} = D^{-1}L$, but for the sake of computational efficiency he does so by computing the second left and right singular vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ of $D_R^{-1/2}AD_C^{-1/2}$ and then forming the Fiedler vector as

$$\boldsymbol{\phi}_1 = \begin{bmatrix} D_R^{-1/2}\boldsymbol{u} \\ D_C^{-1/2}\boldsymbol{v} \end{bmatrix}.$$

Using the Fiedler vector he partitions the rows and the columns simultaneously.

We repeat this process in order to yield recursive partitioning trees for the rows and columns. An advantage of this method is that we are able to utilize graph theory but we do not need to construct an edge weight matrix, which would require defining a weight function and specifying a means of constructing a graph (e.g., k-nearest neighbors, ϵ-net, or a complete graph). And since the

GHWT (Algorithm 3) proceeds according to a hierarchical tree and does not require edge weights, we have all that is required in order to transform the data matrix. However, until this point we have only discussed the use of the GHWT for analyzing a signal $\boldsymbol{f} \in \mathbb{R}^N$, and thus we now explain how the GHWT can be applied to a matrix $A \in \mathbb{R}^{N_R \times N_C}$.

First, we use the recursive partitioning on the columns to apply the GHWT to each row of the matrix. Whereas performing the GHWT on a signal of length N yields an $N \times j_{\max}$ matrix which we can input to the best basis algorithm, here the result is an array of size $N_R \times N_C \times j_{\max}^{\text{col}}$, where $j_{\max}^{\text{col}}$ is the maximum level in the recursive partitioning of the columns. Therefore, our next step is to "flatten" this 3-dimensional array to a 2-dimensional matrix of size $N_C \times j_{\max}^{\text{col}}$. There are various ways in which we can do this, but typically we take the 1-norm (i.e., the sum of the absolute values) along the dimension corresponding to the rows. Thus, the magnitudes of the entries in the resulting matrix reflect the magnitudes of the entries in the row dimension of the 3-dimensional array. We then apply the GHWT best basis algorithm to the $N_C \times j_{\max}^{\text{col}}$ matrix using a cost functional of our choosing to obtain the best basis for the columns. Next, we apply the GHWT to each column of the original matrix, which yields an array of size $N_R \times N_C \times j_{\max}^{\text{row}}$. As we did for the columns, we "flatten" this to a matrix of size $N_R \times j_{\max}^{\text{row}}$ and then use the best basis algorithm to find the best basis for the rows.

At this point, we now have a best basis for the rows and a best basis for the columns. For each row, we select the coefficients corresponding to the rows' best basis from the $N_R \times N_C \times j_{\max}^{\text{row}}$ array, and the result is a row-transformed $N_R \times N_C$ matrix. We now use the recursive partitioning on the columns to apply the GHWT to each row of this transformed matrix, once again yielding an array of size $N_R \times N_C \times j_{\max}^{\text{col}}$. We extract the coefficients corresponding to the columns' best basis, yielding the final result of our analysis: a row- and column-transformed $N_R \times N_C$ matrix of GHWT expansion coefficients. (This GHWT matrix analysis scheme is implemented in `GHWT_Matrix_Analysis_BestBasis.m`.)

Although in our description we transform the rows and extract the best basis, then transform the columns and extract the best basis, it doesn't matter whether we analyze the rows or the columns first. To see why, let $\boldsymbol{\Psi}_{\text{rows}} \in \mathbb{R}^{N_R \times N_R}$ and $\boldsymbol{\Psi}_{\text{cols}} \in \mathbb{R}^{N_C \times N_C}$ denote the orthogonal matrices whose columns are the row and column best basis vectors. Although we do not form these

matrices for the sake of computational efficiency, our matrix transform is equivalent to computing $\mathbf{\Psi}_{\text{rows}}^{\mathsf{T}} A \mathbf{\Psi}_{\text{cols}}$, and thus it is not impacted by which dimension is transformed first.

10.2. Experimental Results

10.2.1. Science News Data Matrix

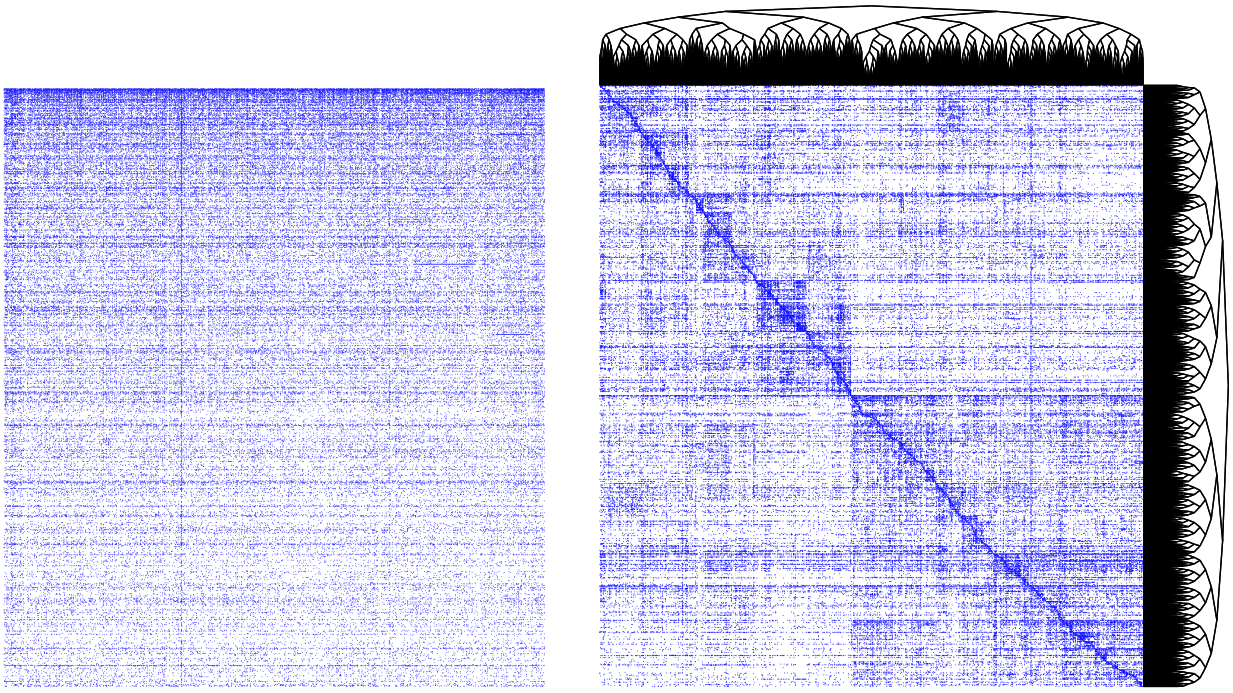
For our first matrix analysis demonstration, we use a version of the Science News database, which was originally prepared by Solka and his collaborators [**54**] and later revised by Mauro Maggioni and then Naoki Saito. This is a term-document matrix with 1153 rows (words), 1042 columns (documents), and 121,714 nonzero entries (10.13% sparsity). The document classification data for the matrix is displayed in Table 10.1.

Class	**Number of Documents**	**Percent of Total**
Anthropology & Archeology	54	5.18%
Astronomy & Space Sciences	120	11.52%
Behavior	72	6.91%
Earth & Environmental Sciences	137	13.15%
Life Sciences	205	19.67%
Mathematics & Computers	58	5.57%
Medical Sciences	279	26.78%
Physical Science & Technology	117	11.23%
Total	**1042**	

TABLE 10.1. Document classifications from the Science News data set that we use for our experiment.

The original matrix is shown in Figure 10.1a. We recursively partition the rows and columns of this matrix using Dhillon's matrix bipartitioning scheme, the result of which is shown in Figure 10.1b. Whereas the original appears to be devoid of structure, we clearly observe a block diagonal structure in the recursively partitioned and reordered matrix. We then use these recursive

partitionings to analyze the matrix, using the 1-norm as both the cost functional and the "flattening" method in the best basis algorithm. We also analyze the matrix using the graph Haar basis and the graph Walsh basis (i.e., level $j = 0$ of the GHWT coarse-to-fine dictionary), and we plot the relative error curves for all three in Figure 10.2.



(a) (b)

FIGURE 10.1. (a) The Science News term-document matrix used for this experiment. (b) The matrix after recursively partitioning the rows and columns by repeatedly applying Dhillon's bipartitioning method. The orders of the rows and columns are permuted to match the ordering in their recursive partitionings.

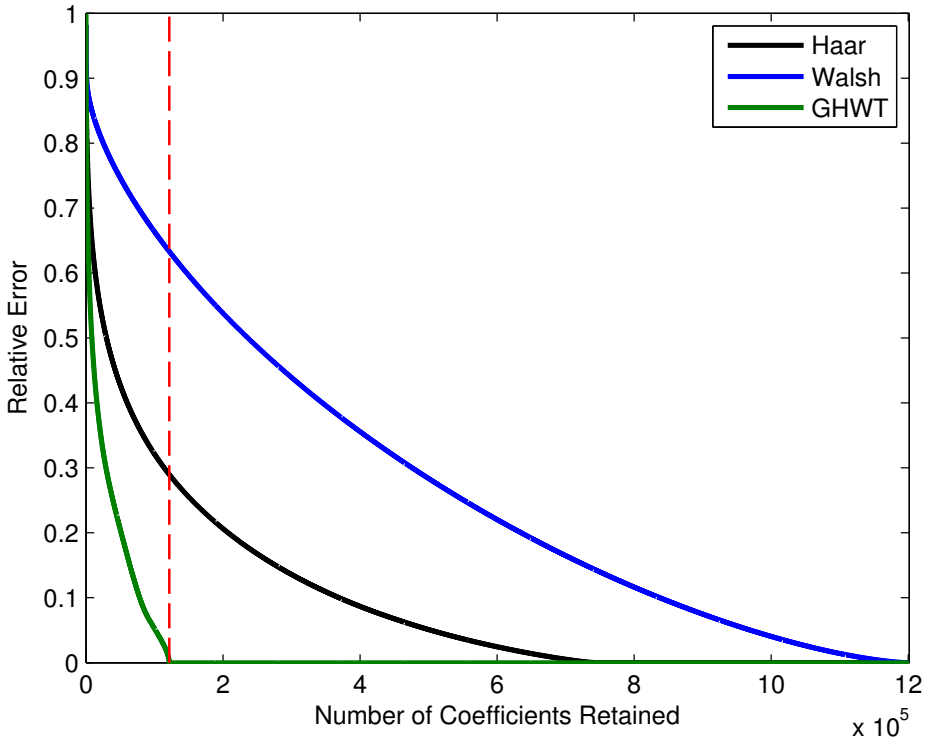


FIGURE 10.2. The relative error curves for the n-term nonlinear approximations of the Science News matrix using the tensor Haar basis, tensor Walsh basis, and the GHWT tensor best basis. The dashed vertical line indicates the number of nonzero entries in the matrix.

In this example both the Haar and Walsh bases do a very poor job of capturing the structure of the signal, requiring far more coefficients than the canonical basis to achieve perfect reconstruction. This isn't surprising for the Walsh basis, since its basis vectors have global support and therefore are a poor choise for a sparse signal. The GHWT best basis performs much better than these two, selecting a basis that is almost exactly the canonical basis: the rows' best basis differs in only six locations (three pairs of words which are combined), and the columns' best basis differs in only two locations. The GHWT best basis achieves perfect reconstruction with 121,817 coefficients retained, which is merely 1% more than the 121,714 nonzero entries in the original matrix.

Although the GHWT best basis dramatically outperforms the graph Haar basis in this example, it may at first seem to be an unsatisfactory and uninteresting victory, but upon further inspection we find that this is not the case. First, selecting a basis that performs nearly as well as the canonical basis is no small feat when considering that there are more than 1.99×10^{195} choosable

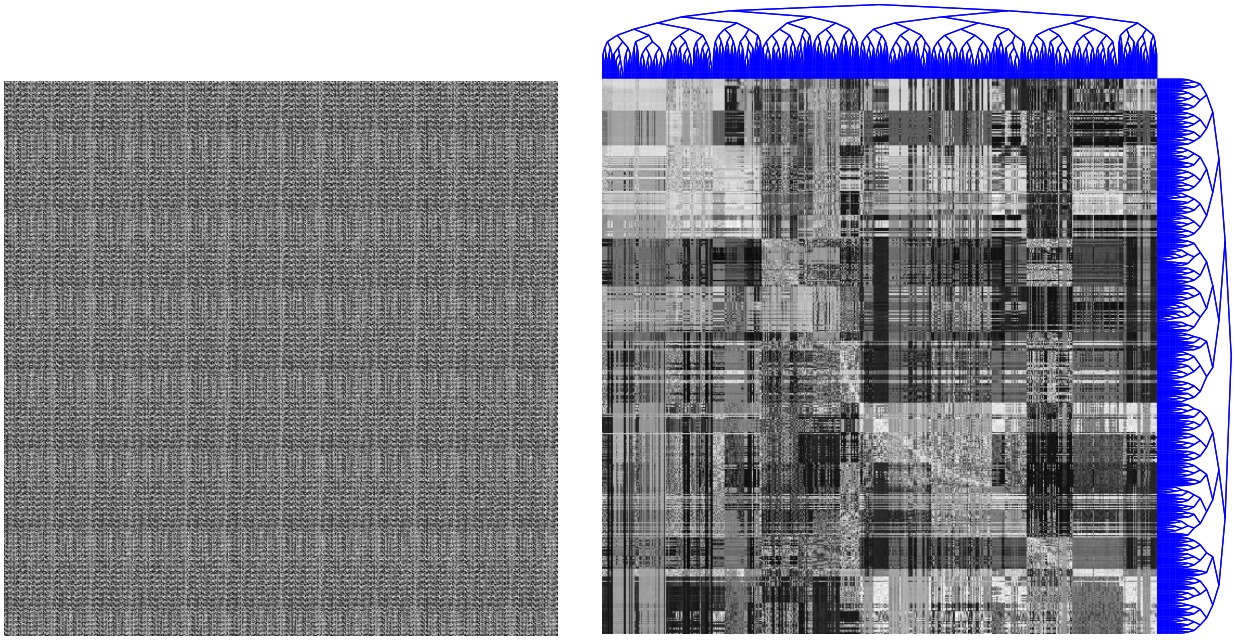
bases for the rows and 6.09×10^{174} choosable bases for the columns, and thus a total of more than 1.21×10^{370} choosable tensor bases. And while the nearly canonical row and column bases are not terribly interesting, they perform well for this example and thus the best basis algorithm's choices are well justified. Second, one may argue that it is a downfall of our method that the best basis is outperformed by the canonical basis, but to say this is incorrect. The best basis does indeed have a smaller 1-norm than the canonical basis: 1041.6 vs. 1042.0 (although the guarantee from Proposition 6.1 does not apply because the best basis algorithm searches through the matrix of flattened coefficients, not the 3-dimensional array of expansion coefficients). Furthermore, if sparsity is the ultimate goal, then the most appropriate cost functional would be the number of nonzeros, which does yield the canonical basis in this experiment; in fact, for $p \leq 0.001$, using the p-quasinorm leads to the canonical basis being selected. However, we can gain some insights into the data by examining the words (rows) and documents (columns) that the best basis algorithm with ℓ^1 cost functional chooses to combine. The combined pairs of words are "el" and "niño," "la" and "niña," and "meteor" and "shower." In other words, the best basis algorithm with ℓ^1 cost functional chooses to combine words which are very frequently used together. The documents that it elects to combine are "Science Talent Search announces finalists" and "Talent Search: Student finalists' flair for science to be rewarded." The titles are very similar, as are the entries in their columns. Although combining these words and documents leads to a slightly less sparse representation, the tradeoff is the insight provided by the best basis algorithm.

We observe an interesting behavior when we change the cost function used by the best basis algorithm. When using the 0.1-quasinorm, in addition to the three pairs of words combined by the 1-norm, the words "orbiting" and "extrasolar" are combined, as are the three words "tornado," "tornadoe," and "meteorologist." We also find that there are 8 pairs of documents that are combined, along with one group of three documents and one group of four. Using the 0.01-quasinorm yields the same groups as the 0.1-quasinorm, along with one additional pair of documents. However, as previously mentioned, when we use the 0.001-quasinorm the best basis algorithm selects the canonical basis (from the top levels of the fine-to-coarse row and column dictionaries).

10.2.2. Shuffled "Barbara" Matrix

Having analyzed a sparse matrix, we now consider a very different form of matrix data: an image whose rows and columns have been randomly permuted. We use the famous "Barbara" image, displayed in Figure 10.3. We permute the rows and columns, and the resulting image is shown in Figure 10.4a. We then recursively partition the rows and columns of the permuted image using Dhillon's bipartitioning method, the result of which is displayed in Figure 10.4b. Whereas the permuted image looks like nothing more than noise, this matrix possesses textured regions and block structures. While the partitioned matrix differs greatly from the original Barbara image, we emphasize that recovering the original image is neither our aim nor our expectation. Rather, our objective is to demonstrate the effectiveness of the GHWT for analyzing matrix data with an intricate and irregular structure.

FIGURE 10.3. The famous "Barbara" image (512×512).



(a) (b)

FIGURE 10.4. (a) The Barbara image after shuffling its rows and columns. (b) The result after recursively partitioning and reordering the shuffled Barbara image.

Having generated hierarchical trees for the rows and columns of the matrix, our next step is to analyze the matrix with the GHWT and determine the row and column best bases. We use the 1-norm as both the cost functional and "flattening" method for the best basis algorithm. We also analyze the matrix using the graph Haar basis, the classical Haar basis, and the Coiflet-4 wavelet. For the classical Haar and Coiflet-4 bases, we analyze both the shuffled (Fig. 10.4a) and reordered (Fig. 10.4b) matrices. We plot relative error curves for each basis in Figure 10.5. From this plot we see that reordering the rows and columns in accordance with their recursive partitionings leads to better results for the classical Haar and Coiflet-4 bases. The graph Haar basis and GHWT best basis achieve even better results by taking into account the full structures of the recursive partitionings. Here, the GHWT best bases for the rows and columns both originate from their respective fine-to-coarse dictionary, and both are very similar in structure to the graph Haar basis.

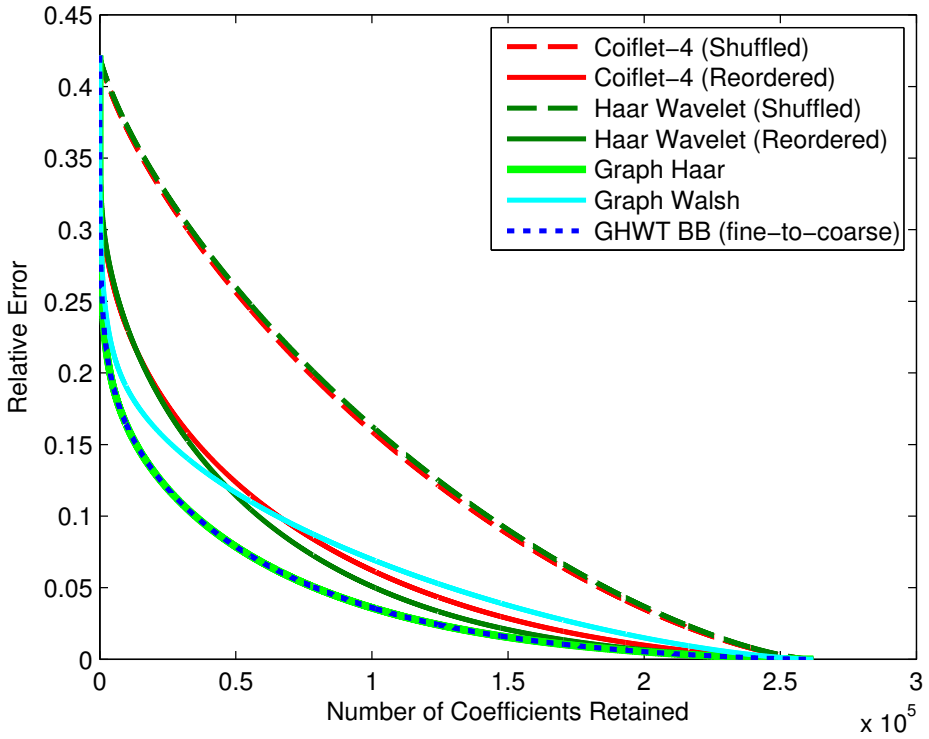


FIGURE 10.5. Relative error curves for the shuffled Barbara image.

Aside from approximating the matrix data, we can also use the GHWT and best basis algorithm to ascertain information about the spatial structure of the matrix by restricting our best basis searches to the coarse-to-fine dictionary. Figure 10.6 illustrates the row and column bases, which were chosen using the 0.1-quasinorm (i.e., the τ-measure with $\tau = 0.1$) as the cost functional for the row and column best basis searches and using the 1-norm as the "flattening" method. For this matrix, using the ℓ^1 norm for the cost functional and for flattening simply leads to the global Walsh basis (i.e., there are no row or column partitions). While the 0.1-quasinorm and ℓ^1 norm both promote sparsity, the difference is that small coefficients have a larger cost in the 0.1-quasinorm. As a simple example, the signals $\boldsymbol{x} = (1/\sqrt{2}, 1/\sqrt{2}, 0.0001)^{\mathsf{T}}$ and $\boldsymbol{y} = (0.9975, 0.05, 0.05)^{\mathsf{T}}$ have the same 2-norm, and we have that $\|\boldsymbol{x}\|_1 > \|\boldsymbol{y}\|_1$ but the 0.1-quasinorm of $\boldsymbol{x}$ is smaller than that of $\boldsymbol{y}$. In the context of our shuffled Barbara image, using the ℓ^1 norm as the cost functional concentrates the energy in a small number of coefficients, whereas the 0.1-quasinorm seeks to maximize the number of very small coefficients and thus leads to more localized basis functions. We will explore different cost functionals and flattening methods for this example shortly.

Having specified the 0.1-quasinorm and ℓ^1 flattening, the best basis algorithm determines the quantity and locations of the row and column partitions. We observe that this tensor coarse-to-fine basis divides the matrix into textured patches, and in a sense this experiment is a 2-dimensional analog of our 1-dimensional segmentation results from the previous chapter.

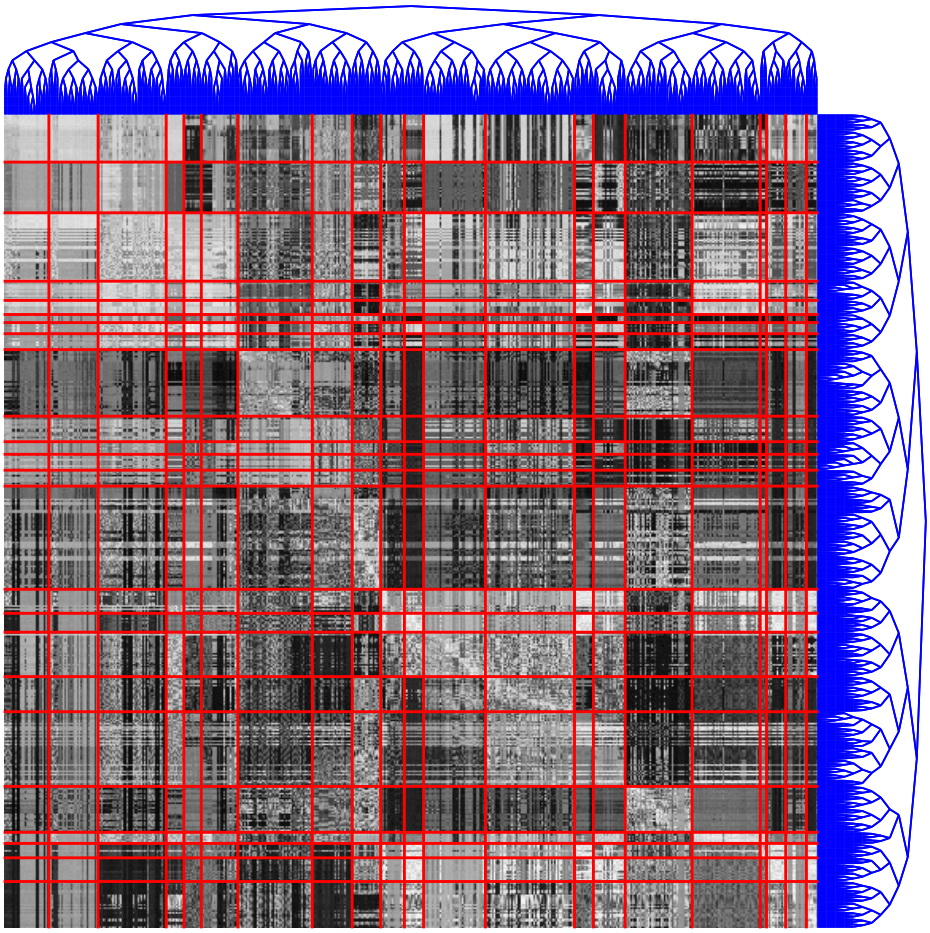
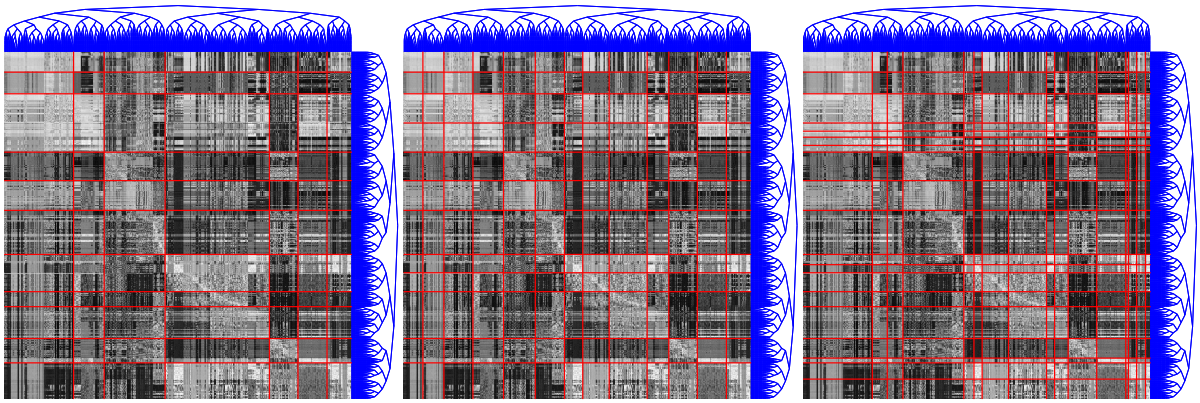


FIGURE 10.6. An illustration of the GHWT row and column bases selected by the best basis algorithm with $\tau = 0.1$ and ℓ^1 flattening.

There are a number of ways in which this method can be modified. The first is the choice of cost functionals used to select the row and column best bases. Figure 10.7a shows the result of this experiment when using the 0.5-quasinorm as the cost functional for both best basis searches, and we see that this leads to fewer row and column partitions. Another option is to limit the number of partitions by requiring each of them to be of a minimum length. For example, in Figure 10.7b we effectively disregard row and column regions with fewer than $[N_R/20] = [N_C/20]$ nodes from

our best basis search, again using the 0.1-quasinorm as our cost functional. To accomplish this, after transforming each row according to the columns' recursive partitioning, we set coefficients in regions with fewer than $[N_C/20]$ nodes to infinity before running the best basis algorithm and finding the column basis; we proceed likewise for the rows. Interestingly, this leads to a total of 13 regions in the column basis and 12 regions in the row basis, and thus the best basis algorithm chooses significantly fewer than 20 regions for both. The last of these simple modifications to the method is to use a different means of flattening the 3-dimensional arrays to 2-dimensional matrices. Figure 10.7c shows the result when we do so by taking the 2-norm along the row/column dimension prior to performing the column/row best basis search; again, we use the 0.1-quasinorm as our cost functional. As these three modifications demonstrate, the method can be tailored to one's needs by appropriately influencing the best basis searches.



(a) (b) (c)

FIGURE 10.7. (a) The row and column best bases selected using the 0.5-quasinorm as the cost functional. (b) The best bases selected using the 0.1-quasinorm; effectively, regions of length shorter than $[N_R/20] = [N_C/20]$ were not considered. (c) The best bases found using the 0.1-quasinorm and flattening the 3-dimensional arrays to 2-dimensional matrices by taking the 2-norm along the extraneous dimension.

As a whole, this "Barbara" experiment demonstrates the potential of the GHWT and its best basis algorithm for analyzing matrices with complicated underlying structures. This method requires minimal parameters: only the cost functionals and methods for flattening 3-dimensional arrays to 2-dimensional matrices. Future research is needed to investigate the use of different cost functionals, flattening methods, and other strategies for influencing the best basis searches.

10.3. Summary

In this chapter we presented a method for using the GHWT to analyze matrix data. The advantage of using graph-based techniques for this classical problem is that we are better equipped to account for complicated and irregular structures within the matrix data. We demonstrated that our method significantly outperformed the tensor Haar and Walsh bases for a sparse term-document matrix, and in doing so led to several interesting discoveries about the data. We also showed an example in which our method adaptively chooses row and column bases that partition a dense matrix into textured patches. Both of these examples motivate further research exploring the use of the GHWT and best basis algorithm for machine learning purposes.

On a more general note, both the HGLET and GHWT are capable of analyzing vector-valued signals on graphs. Whereas applying the HGLET and GHWT to a scalar-valued signal $\boldsymbol{f} \in \mathbb{R}^N$ yields a matrix, applying them to a vector-valued signal $\boldsymbol{F} \in \mathbb{R}^{N \times d}$ yields a 3-dimensional array. By "flattening" this to a 2-dimensional matrix, we are then able to utilize the best basis algorithm. An interesting subject of future work is whether it is possible to generalize the best basis algorithm so that it can directly handle arrays of 3 (or more) dimensions, i.e., without needing to "flatten" the coefficients. While such methods exist for wavelets and the block DCT, extending these methods to the GHWT and HGLET involves a number of challenges. Most notably, we lose the dyadic structure, and so the algorithm becomes much more complex than comparing the cost of a square block of coefficients to that of the four square blocks below it. Thus, a challenging case would be to develop such a best basis algorithm that is able to handle matrices with, say, $N_R \gg N_C$. However, we believe that this algorithm would be highly effective for analyzing data matrices, and therefore devising such a method will be a subject of future work.

CHAPTER 11

Conclusion

In this dissertation we have presented our two multiscale transforms for analyzing signals on graphs: the Hierarchical Graph Laplacian Eigen Transform (HGLET) and the Generalized Haar-Walsh Transform (GHWT). We developed these transforms in an effort to generalize the block DCT and wavelet packets from classical signal processing to the graph setting, and we are aware of only one other attempt to do so (the diffusion wavelet packets of Bremer et al. [**5**]). As part of our foray into this new mathematical territory, we have generalized classical techniques as well as developed new tools for working with signals on graphs. Of course, an overcomplete dictionary is only as good as the basis that is selected from it, and thus our generalizations of the best basis algorithm are inseparable sidekicks for our two transforms.

Both the HGLET and GHWT are dependent upon the hierarchical tree for the graph at hand. In this dissertation, we have used Fiedler vectors of Laplacian matrices to generate our recursive partitionings for general graphs. (For path graphs we have explicitly solved for the minimizer of the Normalized Cut. For matrices we have used the method of Dhillon [**24**], which itself makes use of Fiedler vectors.) However, our transforms are compatible with other means of generating suitable hierarchical trees; i.e., those that satisfy the four criteria set forth in Chapter 3 (page 51). For example, one could use the diffuse interface model of Bertozzi and Flenner [**3**] or the local spectral method of Mahoney et al. [**44**].

We proved some theoretical results for approximation using our transforms in conjunction with their best basis algorithms in Chapter 7. We also presented numerical examples for two graph signals, both of which are real data, and we compared our transforms to previously developed methods: Laplacian eigenvectors, the graph Haar basis, graph-QMF, and Laplacian multiwavelets. As the field of signal processing on graphs is relatively young and lacks many of the theoretical foundations from classical signal processing, we feel that it is very important to perform experiments such as these in which various methods are directly compared using real signals. While synthetic

signals can certainly provide useful insight, a concern is that the signals may be generated in such a manner that they are either unrealistic or biased towards certain transforms. Along these lines, we mention that one of the graph signals that we analyzed (the traffic volume on the Toronto road network) was our own novel construction from publicly available real data. This signal is included in the MTSG toolbox, and it is our hope that other researchers will analyze it and also make available graph signals of their own.

Building upon our approximation results, we demonstrated the effectiveness of our transforms for denoising signals on graphs. We then applied our graph methods to two classical problems. First, we used the HGLET to simultaneously segment, denoise, and compress 1-dimensional signals by viewing them as signals on path graphs. We did this using an iterative algorithm in which we repeatedly partition the graph, analyze the signal, find the best basis using the minimum description length (MDL) as the cost functional, and modify edge weights. Second, we analyzed matrix data using the GHWT, which allows us to account for the underlying structure of the data. Our experimental results demonstrated the potential of the GHWT and best basis algorithm as machine learning tools.

Although it is not explicitly included in this dissertation, a major contribution of this work is the software that accompanies it. We have developed a general framework for working with signals on graphs, and this includes specific modules that are useful for multiscale transforms and overcomplete transforms. Along with the software we provide scripts for reproducing figures from our articles [**37**, **38**, **39**] and this dissertation. It is our hope that interested readers will download the software themselves, recreate our figures, and possibly conduct their own experiments with it.

There is still much research to be done on the transforms and methods pioneered in this work. Perhaps the most pressing matter is the development of theoretical results for signal processing on graphs. As mentioned in Chapter 7, this is challenging because we lack many of the tools and foundations from classical signal processing, but efforts have been and will continue to be made. An aspect of this work that warrants further exploration is the choice of cost functional for the best basis algorithm, and also how this choice depends on the task at hand (e.g., approximation, denoising, discovery of underlying structure, etc.). Another matter that merits exploration is whether we can generalize the simultaneous segmentation, denoising, and compression method from Chapter 9 to

the setting of more general graphs. A main challenge in doing so is that we no longer have a simple notion of neighbor regions, as we did for the intervals in the 1-D case. But with challenges come opportunity, and it may be possible to use the same strategy to construct graphs from point clouds using an iterative algorithm involving the HGLET and/or GHWT and the best basis algorithm. The principle behind this is that a better graph should enable a signal to be represented more efficiently (either in terms of the MDL or some other cost functional). We plan to publish future work in a forthcoming paper.

Bibliography

[1] Agaian, S. S., Sarukhanyan, H., Egiazarian, K., and Astola, J. *Hadamard Transforms*. SPIE, 2011.

[2] Bennett, N. N. Fast algorithm for best anisotropic Walsh bases and relatives. *Appl. Comput. Harmon. Anal. 8*, 1 (2000), 86–103.

[3] Bertozzi, A. L., and Flenner, A. Diffuse interface models on graphs for classification of high dimensional data. *Multiscale Model. Simul. 10*, 3 (2012), 1090–1118.

[4] Biyikoğlu, T., Hordijk, W., Leydold, J., Pisanski, T., and Stadler, P. F. Graph Laplacians, nodal domains, and hyperplane arrangements. *Linear Algebra Appl. 390* (2004), 155–174.

[5] Bremer, J. C., Coifman, R. R., Maggioni, M., and Szlam, A. D. Diffusion wavelet packets. *Appl. Comput. Harmon. Anal. 21*, 1 (2006), 95–112.

[6] Chen, S. S., Donoho, D. L., and Saunders, M. A. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput. 20*, 1 (1998), 33–61.

[7] Chung, F. R. K. *Spectral Graph Theory*, vol. 92 of *CBMS Regional Conference Series in Mathematics*. Amer. Math. Soc., Providence, RI, 1997.

[8] Coifman, R., and Donoho, D. Translation-invariant de-noising. In *Wavelets and Statistics*, A. Antoniadis and G. Oppenheim, Eds., vol. 103 of *Lecture Notes in Statistics*. Springer New York, 1995, pp. 125–150.

[9] Coifman, R., and Leeb, W. Earth mover's distance and equivalent metrics for spaces with hierarchical partition trees. Tech. rep., YALEU/DCS/TR-1482, Yale University, 2013.

[10] Coifman, R. R., and Gavish, M. Harmonic analysis of digital data bases. In *Wavelets and Multiscale Analysis*, J. Cohen and A. I. Zayed, Eds. Birkhäuser/Springer, New York, 2011, pp. 161–197.

[11] Coifman, R. R., and Maggioni, M. Diffusion wavelets. *Appl. Comput. Harmon. Anal. 21*, 1 (2006), 53–94.

[12] Coifman, R. R., and Meyer, Y. Remarques sur l'analyse de Fourier à fenêtre. *C. R. Acad. Sci. Paris Sér. I Math. 312*, 3 (1991), 259–261.

[13] Coifman, R. R., Meyer, Y., and Wickerhauser, V. Wavelet analysis and signal processing. In *Wavelets and Their Applications*. Jones and Bartlett, Boston, MA, 1992, pp. 153–178.

[14] Coifman, R. R., and Wickerhauser, M. V. Entropy-based algorithms for best basis selection. *IEEE Trans. Inform. Theory 38*, 2 (1992), 713–718.

[15] Cooley, J. W., and Tukey, J. W. An algorithm for the machine calculation of complex Fourier series. *Math. Comp. 19* (1965), 297–301.

[16] Coombs, J., van der List, D., Wang, G.-Y., and Chalupa, L. Morphological properties of mouse retinal ganglion cells. *Neuroscience 140*, 1 (2006), 123 – 136.

[17] Cover, T. M., and Thomas, J. A. *Elements of Information Theory*, second ed. Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, 2006.

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

[19] Daubechies, I. *Ten Lectures on Wavelets*, vol. 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Soc. Ind. Appl. Math. (SIAM), Philadelphia, PA, 1992.

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

[21] Deng, D., and Han, Y. *Harmonic Analysis on Spaces of Homogeneous Type*, vol. 1966 of *Lecture Notes in Mathematics*. Springer, 2009.

[22] DeVore, R. A. Nonlinear approximation. In *Acta Numerica, 1998*, vol. 7 of *Acta Numer.* Cambridge Univ. Press, Cambridge, 1998, pp. 51–150.

[23] DeVore, R. A., Jawerth, B., and Lucier, B. J. Image compression through wavelet transform coding. *IEEE Trans. Inform. Theory 38*, 2, part 2 (1992), 719–746.

[24] Dhillon, I. S. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2001), KDD '01, ACM, pp. 269–274.

[25] Donoho, D. L. Wavelet shrinkage and W.V.D.: A 10-minute tour. In *Progress in Wavelet Analysis and Applications* (1993), Y. Meyer and S. Roques, Eds., Editions Frontieres, pp. 109–128.

[26] Donoho, D. L., Buckheit, J., Clerc, M., Duncan, M. R., Huo, X., Johnstone, I., Kalifa, J., Levi, O., Mallat, S., and Yu, T. Wavelab. `http://statweb.stanford.edu/~wavelab/`.

[27] Donoho, D. L., and Johnstone, I. M. Ideal spatial adaptation by wavelet shrinkage. *Biometrika 81*, 3 (1994), 425–455.

[28] Donoho, D. L., Johnstone, I. M., Kerkyacharian, G., and Picard, D. Wavelet shrinkage: asymptopia? *J. Roy. Statist. Soc. Ser. B 57*, 2 (1995), 301–369. With discussion and a reply by the authors.

[29] Fiedler, M. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Math. J. 25(100)*, 4 (1975), 619–633.

[30] Flajolet, P., and Odlyzko, A. The average height of binary trees and other simple trees. *J. Comput. System Sci. 25*, 2 (1982), 171–213.

[31] Flajolet, P., and Odlyzko, A. M. Limit distributions for coefficients of iterates of polynomials with applications to combinatorial enumerations. *Math. Proc. Cambridge Phil. Soc. 96*, 2 (1984), 237–253.

[32] Gavish, M., Nadler, B., and Coifman, R. R. Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (Haifa, Israel, June 2010), J. Fürnkranz and T. Joachims, Eds., Omnipress, pp. 367–374.

[33] Grünwald, P. D. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[34] Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. Anal. 30*, 2 (2011), 129–150.

[35] Herley, C., Xiong, Z., Ramchandran, K., and Orchard, M. T. Joint space-frequency segmentation using balanced wavelet packet trees for least-cost image representation. *IEEE Trans. Image Process. 6*, 9 (1997), 1213–1230.

[36] Hunter, J. K., and Nachtergaele, B. *Applied Analysis*. World Scientific Publishing Co. Inc., River Edge, NJ, 2001.

[37] Irion, J., and Saito, N. The generalized Haar-Walsh transform. *Proc. 2014 IEEE Statistical Signal Processing Workshop* (2014), 472–475.

[38] Irion, J., and Saito, N. Hierarchical graph Laplacian eigen transforms. *JSIAM Letters 6* (2014), 21–24.

[39] Irion, J., and Saito, N. Applied and computational harmonic analysis on graphs and networks. In *Proc. SPIE* (2015), vol. 9597, pp. 95971F–95971F–15.

[40] Jaffard, S., Meyer, Y., and Ryan, R. D. *Wavelets: Tools for Science & Technology*, revised ed. Soc. Ind. Appl. Math. (SIAM), Philadelphia, PA, 2001.

[41] Jansen, M., Nason, G. P., and Silverman, B. W. Multiscale methods for data on graphs and irregular multidimensional situations. *J. R. Stat. Soc. Ser. B Stat. Methodol. 71*, 1 (2009), 97–125.

[42] Kaiser, G. *A Friendly Guide to Wavelets*. Birkhäuser Boston Inc., Boston, MA, 1994.

[43] Lee, A. B., Nadler, B., and Wasserman, L. Treelets—an adaptive multi-scale basis for sparse unordered data. *Ann. Appl. Stat. 2*, 2 (2008), 435–471.

[44] Mahoney, M. W., Orecchia, L., and Vishnoi, N. K. A local spectral method for graphs: with applications to improving graph partitions and exploring data graphs locally. *J. Mach. Learn. Res. 13* (2012), 2339–2365.

[45] Mallat, S. *A Wavelet Tour of Signal Processing: The Sparse Way*, third ed. Elsevier/Academic Press, Amsterdam, 2009.

[46] Mallat, S. G., and Zhang, Z. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Process. 41*, 12 (1993), 3397–3415.

[47] Marcellin, M., Gormish, M., Bilgin, A., and Boliek, M. An overview of JPEG-2000. In *Proc. IEEE Data Compression Conference* (2000), pp. 523–541.

[48] Mhaskar, H. N., and Prestin, J. Polynomial frames: a fast tour. *Approximation Theory XI: Gatlinburg* (2004), 101–132.

[49] Murtagh, F. The Haar wavelet transform of a dendrogram. *J. Classification 24*, 1 (2007), 3–32.

[50] Nakatsukasa, Y., Saito, N., and Woei, E. Mysteries around the graph Laplacian eigenvalue 4. *Linear Algebra Appl. 438*, 8 (2013), 3231–3246.

[51] Narang, S., and Ortega, A. Perfect reconstruction two-channel wavelet filter banks for graph structured data. *IEEE Trans. Signal Process. 60*, 6 (June 2012), 2786–2799.

[52] Narang, S., and Ortega, A. Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs. *IEEE Trans. Signal Process. 61*, 19 (Oct 2013), 4673–4685.

[53] Negash, B., and Nikookar, H. Wavelet based OFDM for wireless channels. In *Proceedings of IEEE VTS 53rd Vehicular Technology Conference, VTC Spring 2001, Rhodes, Greece, May 6-9, 2001* (2001), vol. 1, pp. 688–691 vol.1.

[54] Priebe, C. E., Marchette, D. J., Park, Y., Wegman, E. J., Solka, J. L., Socolinsky, D. A., Karakos, D., Church, K. W., Guglielmi, R., Coifman, R. R., Lin, D., Healy, D. M., Jacobs, M. Q., and Tsao, A. Iterative denoising for cross-corpus discovery. In *COMPSTAT 2004—Proceedings in Computational Statistics*. Physica, Heidelberg, 2004, pp. 381–392.

[55] Rahman, I. U., Drori, I., Stodden, V. C., Donoho, D. L., and Schröder, P. Multiscale representations for manifold-valued data. *Multiscale Model. Simul. 4*, 4 (2005), 1201–1232.

[56] Rieder, P., Gotze, J., and Nossek, J. Algebraic design of discrete multiwavelet transforms. In *IEEE International Conference on Acoustics, Speech, and Signal Processing* (Apr 1994), vol. iii, pp. III/17–III/20.

[57] Rieder, P., Gotze, J., and Nossek, J. Multiwavelet transforms based on several scaling functions. In *Proceedings of the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis* (Oct 1994), pp. 393–396.

[58] Rissanen, J. A universal prior for integers and estimation by minimum description length. *Ann. Statist. 11*, 2 (1983), 416–431.

[59] Rissanen, J. *Stochastic Complexity in Statistical Inquiry*, vol. 15 of *World Scientific Series in Computer Science*. World Scientific Publishing Co., Inc., Teaneck, NJ, 1989.

[60] Rubinstein, R., Bruckstein, A., and Elad, M. Dictionaries for sparse representation modeling. *Proc. IEEE 98*, 6 (June 2010), 1045–1057.

[61] Ruch, D. K., and Van Fleet, P. J. *Wavelet Theory*. John Wiley & Sons, Inc., Hoboken, NJ, 2009.

[62] Rustamov, R., and Guibas, L. Wavelets on graphs via deep learning. In *Advances in Neural Information Processing Systems* (2013), pp. 998–1006.

[63] Rustamov, R. M. Average interpolating wavelets on point clouds and graphs. *arXiv preprint arXiv:1110.2227* (2011).

[64] Saito, N. Simultaneous noise suppression and signal compression using a library of orthonormal bases and the minimum description length criterion. In *Wavelets in geophysics (Baltimore, MD, 1993)*, vol. 4 of *Wavelet Anal. Appl.* Academic Press, San Diego, CA, 1994, pp. 299–324.

[65] Saito, N. Least statistically dependent basis and its application to image modeling. In *Proc. SPIE* (1998), vol. 3458, pp. 24–37.

[66] Saito, N., and Coifman, R. R. Local discriminant bases. In *Proc. SPIE* (1994), A. F. Laine and M. A. Unser, Eds., vol. 2303 of *Wavelet Applications in Signal and Image Processing II*, pp. 2–14.

[67] Saito, N., and Coifman, R. R. On local orthonormal bases for classification and regression. In *1995 International Conference on Acoustics, Speech, and Signal Processing* (May 1995), vol. 3, IEEE Signal Processing Society, pp. 1529–1532.

[68] Saito, N., Coifman, R. R., Geshwind, F. B., and Warner, F. Discriminant feature extraction using empirical probability density estimation and a local basis library. *Pattern Recognition 35*, 12 (2002), 2841–2852.

[69] Saito, N., and Remy, J.-F. The polyharmonic local sine transform: a new tool for local image analysis and synthesis without edge effect. *Appl. Comput. Harmon. Anal. 20*, 1 (2006), 41–73.

[70] Saito, N., and Woei, E. Simultaneous segmentation, compression, and denoising of signals using polyharmonic local sine transform and minimum description length criterion. In *Statistical Signal Processing Workshop (SSP), 2005 IEEE* (Jul 2005), pp. 315–320.

[71] Saito, N., and Woei, E. Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians. *JSIAM Letters 1* (2009), 13–16.

[72] Saito, N., and Woei, E. On the phase transition phenomenon of graph Laplacian eigenfunctions on trees. *RIMS Kôkyûroku 1743* (2011), 77–90.

[73] Sayood, K. *Introduction to Data Compression*, fourth ed. The Morgan Kaufmann Series in Multimedia Information and Systems. Morgan Kaufmann Publishers Inc., Boston, 2013.

[74] Sharon, N., and Shkolnisky, Y. A class of laplacian multiwavelets bases for high-dimensional data. *Appl. Comput. Harmon. Anal. 38*, 3 (2015), 420 – 451.

[75] Shi, J., and Malik, J. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 8 (2000), 888–905.

[76] Shuman, D., Ricaud, B., and Vandergheynst, P. A windowed graph Fourier transform. In *Statistical Signal Processing Workshop (SSP), 2012 IEEE* (Aug 2012), pp. 133–136.

[77] Shuman, D., Wiesmeyr, C., Holighaus, N., and Vandergheynst, P. Spectrum-adapted tight graph wavelet and vertex-frequency frames. *IEEE Trans. Signal Process. 63*, 16 (Aug 2015), 4223–4235.

[78] Shuman, D. I., Ricaud, B., and Vandergheynst, P. Vertex-frequency analysis on graphs. *Appl. Comput. Harmon. Anal.* (2015), to appear.

[79] Simon, H. D. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering 2*, 2 (1991), 135–148.

[80] Stein, E. M., and Shakarchi, R. *Fourier Analysis: an Introduction*, vol. 1 of *Princeton Lectures in Analysis*. Princeton University Press, Princeton, NJ, 2003. An introduction.

[81] Strang, G. The discrete cosine transform. *SIAM Rev. 41*, 1 (1999), 135–147 (electronic).

[82] Strang, G., and Fix, G. A Fourier analysis of the finite element variational method. In *Constructive Aspects of Functional Analysis*, G. Geymonat, Ed., vol. 57 of *C.I.M.E. Summer Schools*. Springer Berlin Heidelberg, 2011, pp. 793–840.

[83] Strang, G., and Nguyen, T. *Wavelets and Filter Banks*. Soc. Ind. Appl. Math. (SIAM), 1996.

[84] Sweldens, W. The lifting scheme: a custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal. 3*, 2 (1996), 186–200.

[85] Sweldens, W. The lifting scheme: a construction of second generation wavelets. *SIAM J. Math. Anal. 29*, 2 (1998), 511–546.

[86] Szlam, A. D., Maggioni, M., Coifman, R. R., and Bremer, J. C. Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions. In *Proc. SPIE* (2005), vol. 5914, pp. 59141D–59141D–11.

[87] Thiele, C. M., and Villemoes, L. F. A fast algorithm for adapted time-frequency tilings. *Appl. Comput. Harmon. Anal. 3*, 2 (1996), 91–99.

[88] von Luxburg, U. A tutorial on spectral clustering. *Stat. Comput. 17*, 4 (2007), 395–416.

[89] West, D. B. *Introduction to Graph Theory*, 2 ed. Prentice Hall, September 2000.

[90] Wickerhauser, M. V. *Adapted Wavelet Analysis: from Theory to Software*. A K Peters, Ltd., Wellesley, MA, 1994.