Applied and computational harmonic analysis on graphs and networks

Jeff Irion and Naoki Saito

Department of Mathematics, University of California, One Shields Avenue, Davis, CA 95616, USA

ABSTRACT

In recent years, the advent of new sensor technologies and social network infrastructure has provided huge opportunities and challenges for analyzing data recorded on such networks. In the case of data on regular lattices, computational harmonic analysis tools such as the Fourier and wavelet transforms have well-developed theories and proven track records of success. It is therefore quite important to extend such tools from the classical setting of regular lattices to the more general setting of graphs and networks. In this article, we first review basics of graph Laplacian matrices, whose eigenpairs are often interpreted as the frequencies and the Fourier basis vectors on a given graph. We point out, however, that such an interpretation is misleading unless the underlying graph is either an unweighted path or cycle. We then discuss our recent effort of constructing multiscale basis dictionaries on a graph, including the Hierarchical Graph Laplacian Eigenbasis Dictionary and the Generalized Haar-Walsh Wavelet Packet Dictionary, which are viewed as generalizations of the classical hierarchical block DCTs and the Haar-Walsh wavelet packets, respectively, to the graph setting. Finally, we demonstrate the usefulness of our dictionaries by using them to simultaneously segment and denoise 1-D noisy signals sampled on regular lattices, a problem where classical tools have difficulty.

Keywords: spectral graph partitioning, multiscale basis dictionaries on graphs, wavelets on graphs, the best basis algorithm, signal segmentation, denoising

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.[1–4]

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.[5–14]

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, we cannot 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

Further author information: (Send correspondence to N.S.)
N.S.: E-mail: saito@math.ucdavis.edu, Telephone: 1 530 754 2121
J.I.: E-mail: jlirion@math.ucdavis.edu

Wavelets and Sparsity XVI, edited by Manos Papadakis, Vivek K. Goyal, Dimitri Van De Ville,
Proc. of SPIE Vol. 9597, 95971F · © 2015 SPIE · CCC code: 0277-786X/15/$18
doi: 10.1117/12.2186921

been to develop wavelet-*like* transforms rather than trying to directly transfer classical wavelets to the graph setting. We will review some of these transforms in Section 2.2.

The objective of this article is twofold. First, we review some basic concepts for understanding data analysis on graphs and the challenges that lie therein. Second, we introduce our own contributions to harmonic analysis on graphs: generalizations of overcomplete, multiscale dictionaries from classical signal processing, including wavelet packets and local trigonometric transforms. We conclude with some applications.

2. NOTATION AND REVIEWS

In this section, we will first establish our notation and define some important quantities which will be used throughout this article. Next, we will review some of the recent wavelet-like transforms proposed by other groups, and discuss the fundamental problem of the lack of a true frequency domain in the graph setting. Finally, we will briefly review classical dictionaries of multiscale bases and the corresponding concept of a "best basis."

2.1 Laplacians on graphs

Let us fix our notation and basic concepts in graph theory first. We mostly follow the commonly used notation; see e.g., Ref. 15 for undirected graphs. Let $G = (V, E)$ be a graph, where $V = V(G) = \{v_1, v_2, \ldots, v_n\}$ is the *vertex set* of G and $E = E(G) = \{e_1, e_2, \ldots, e_m\}$ is its *edge set*, where e_k connects two vertices v_i, v_j for some $1 \le i, j \le n$. We only deal with finite n and m in this article. For simplicity, we often write i instead of v_i. A *subgraph* H of a graph G is a graph such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. An edge connecting a vertex $i \in V$ and itself is called a *loop*. If there exists more than one edge connecting some $i, j \in V$, then they are called *multiple edges*. A graph having loops or multiple edges is called a *multiple graph* (or *multigraph*); a graph with neither of these is called a *simple* graph.

An edge $e \in E$ connecting two distinct vertices $i, j \in V$ may or may not have a direction. If e is *directed* from i to j, then we write $e = (i, j)$, and i, j are called the *tail* and the *head* of e, respectively. A directed edge is also referred to as an *arc*. If e is an *undirected* edge between i and j, then we write $e = ij$, and i, j are called the *endpoints* of e. We also say an edge $e = ij$ is *incident with* i and j, and e *joins* i and j. If $e = ij$, then i, j are said to be *adjacent* and we write $i \sim j$. If $e = (i, j)$, then i is adjacent to j, but not vice versa, and we write $i \rightsquigarrow j$. If each edge in E is directed, G is called a *directed graph* or *digraph*. An *undirected graph* is a graph where none of its edges has a direction. A *mixed graph* is a graph with some directed and some undirected edges.

If each edge $e \in E$ has a *weight* (normally nonnegative), written as w_e, then G is called a *weighted* graph. G is said to be *unweighted* if $w_e \equiv 1$ for each $e \in E$. For some problems edge weights are given while for the other problems one needs to define edge weights carefully.

A *path* from i to j in a graph G is a subgraph of G formed by taking a sequence of distinct vertices, starting with i and ending with j, and a minimal set of edges such that consecutive vertices are adjacent. A path starting from i that returns to i (but is not a loop) is called a *cycle*. For any two vertices in V, if there is a path connecting them, then such a graph is said to be *connected*. In the case of a directed graph, it is said to be *strongly connected*. A *tree* is a connected graph without cycles, and is often denoted by T instead of G. For a tree T, we have $|E(T)| = |V(T)| - 1$, where $|\cdot|$ denotes the cardinality of a set.

We say the *length* (or *cost*) $\ell(P)$ of a path P is the sum of its corresponding edge weights, i.e., $\ell(P) := \sum_{e \in E(P)} w_e$. Let $\mathscr{P}_{ij}$ be the set of all possible paths from i to j in G. The *graph distance* from i to j is defined by $d(i, j) := \inf_{P \in \mathscr{P}_{ij}} \ell(P)$. Clearly, for an undirected graph, we always have $d(i, j) = d(j, i)$, but that is not the case for a directed graph in general.

We say two graphs are *isomorphic* if there is a one-to-one correspondence between the vertex sets such that if two vertices are joined by an edge in one graph, the corresponding vertices are also joined by an edge in the other graph.

Let us now define a few important matrices associated with a given graph G. The (weighted) *adjacency matrix* of G, denoted by $W(G) = (w_{ij}) \in \mathbb{R}^{n \times n}$, is a matrix consisting of the edge weights in G. Note that $w_{ij} = 0$ if $ij \notin E(G)$ (undirected) or $(i, j) \notin E(G)$ (directed). It is clear that $W(G)$ is symmetric if G is undirected. The *degree* of vertex i is defined as $d(i) = d_i := \sum_{j=1}^{n} w_{ij}$, i.e., the ith row sum of $W(G)$. If G is directed, then the above definition of the degree is also called the *outdegree* of vertex i, denoted by $d_{\text{out}}(i)$. In that case, one can also define the *indegree* of vertex i by $d_{\text{in}}(i) := \sum_{j=1}^{n} w_{ji}$, i.e., the ith *column* sum of $W(G)$. Occasionally, we need the unweighted versions of

Figure 1: A path graph P_n provides a simple yet important example.

the adjacency matrix and degrees of a weighted graph. These versions are often called the *combinatorial* adjacency matrix and *combinatorial* (out)degrees, denoted by $A(G)$ and $d^c(i) = d_i^c$, respectively. In other words, $A_{ij}(G) = 1$ if $W_{ij}(G) > 0$, otherwise $A_{ij}(G) = 0$, and d_i^c is the ith row sum of $A(G)$, which is the number of incident edges with i. The combinatorial indegree of i is the ith column sum of $A(G)$. Hence, the earlier definition of $d(i) = d_i$ is also referred to as a *weighted degree*. The *degree matrix* $D(G)$ of G is the diagonal matrix $\mathrm{diag}(d_1, \ldots, d_n)$.

Sometimes, each vertex is associated with spatial coordinates in $\mathbb{R}^p$. For example, if we want to analyze a network of sensors and build a graph whose vertices represent the sensors under consideration, then these vertices have spatial coordinates in $\mathbb{R}^2$ or $\mathbb{R}^3$ indicating their current locations. In that case, we write $\boldsymbol{x}_i \in \mathbb{R}^p$ for the location of vertex i. Let $\boldsymbol{f} = (f(1), \ldots, f(n))^\mathsf{T} \in \mathbb{R}^n$ be a data vector, where $f(i) \in \mathbb{R}$ is the value measured at the vertex i of the graph. Let us also define $\mathbf{1}_n := (1, \ldots, 1)^\mathsf{T} \in \mathbb{R}^n$ for future use. Similarly, for a subset of the vertices $X \subseteq V(G)$, we define $\mathbf{1}_X$ to be the vector which is one at all positions corresponding to vertices in X and zero elsewhere.

In order to analyze a graph, we often utilize the eigenvalues and eigenvectors of its *(unnormalized) Laplacian matrix*, defined as

$$L(G) := D(G) - W(G). \tag{1}$$

Two other versions of the Laplacian matrix which are often used in literature include the *random-walk normalized Laplacian* and the *symmetrically normalized Laplacian*:

$$L_{\mathrm{rw}}(G) := D^{-1}(G)\, L(G) = I - D^{-1}(G)\, W(G), \tag{2}$$

$$L_{\mathrm{sym}}(G) := D^{-1/2}(G)\, L(G)\, D^{-1/2}(G) = I - D^{-1/2}(G)\, W(G)\, D^{-1/2}(G). \tag{3}$$

Which version should be used depends on the application. For example, von Luxburg[16] recommends the use of L_{rw} for graph partitioning. We let $\lambda_0 \le \lambda_1 \le \cdots \le \lambda_{n-1}$ be the sorted eigenvalues of $L(G)$, and we denote by $\phi_0, \phi_1, \ldots, \phi_{n-1}$ their corresponding eigenvectors. Let $m_G(\lambda)$ denote the multiplicity of λ as an eigenvalue of $L(G)$, and let $m_G(I)$ be the number of eigenvalues of $L(G)$, multiplicities included, that belong to $I \subset \mathbb{R}$, an interval of the real line. The corresponding quantities for $L_{\mathrm{rw}}(G)$ are denoted by the superscript rw, i.e., λ_j^{rw}, ϕ_j^{rw}, and $m_G^{\mathrm{rw}}(I)$, and similarly for L_{sym}. It is easy to show that $\lambda_j^{\mathrm{rw}} = \lambda_j^{\mathrm{sym}}$ and $\phi_j^{\mathrm{rw}} = D^{-1/2}(G)\phi_j^{\mathrm{sym}}$. (See Ref. 16 for additional details of the relationship between these three matrices and their spectral properties.)

The Laplacian eigenvalues are useful because they reflect various *intrinsic* topological and geometric features of the graph (e.g., connectivity, the maximum distance over all pairs of vertices, mean distance, and the Cheeger constant,[17] etc.). Some simple yet important facts about the Laplacian eigenvalues are the following: for each of these three Laplacian matrices, all the eigenvalues are nonnegative, the smallest eigenvalue $\lambda_0 = 0$, and $m_G(0) \ge 1$ indicates the number of connected components of G. For L and L_{rw}, the zeroth eigenvector is constant: $\phi_0 = \mathbf{1}_n/\sqrt{n}$ and $\phi_0^{\mathrm{rw}} = \mathbf{1}_n/\sqrt{\sum_{i=1}^n d_i}$. Moreover, the smallest positive eigenvalue of the unnormalized Laplacian $L(G)$ is called the *algebraic connectivity* of G and serves as a quantitative measure of graph connectivity.[18–20]

For each of the three Laplacian matrices, the eigenvector corresponding to the smallest nonzero eigenvalue is referred to as the *Fiedler vector*, and it plays a key role in spectral clustering and spectral graph partitioning.[16,21] (See Sections 3 and 4 for our use of the Fiedler vectors of graphs.) The full set of Laplacian eigenvectors provides us with additional information for graph partitioning via the celebrated Courant nodal domain theorem.[22,23] Moreover, it also provides an orthonormal basis which can be used for representation, approximation, and analysis of data on G.

The simple path P_n consisting of n vertices, shown in Figure 1, provides an important insight for the development of our new multiscale transforms. The graph Laplacian of such a path graph can be easily obtained and is instructive:

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

As pointed out in Ref. 24, the eigenvectors of this matrix are nothing but the *DCT Type II* basis vectors, which are used in the JPEG image compression standard; see e.g., Ref. 25 for more about the DCT. In fact, we have

$$\lambda_k = 4\sin^2\left(\frac{\pi k}{2n}\right); \quad \phi_k(j) = a_{k;n}\cos\left(\frac{\pi k}{n}\left(j-\frac{1}{2}\right)\right), \quad j = 1,\dots,n;\ k = 0,1,\dots,n-1, \tag{4}$$

where $\phi_k = \left(\phi_k(1),\cdots,\phi_k(n)\right)^{\mathsf{T}}$ is the eigenvector corresponding to λ_k, and $a_{k;n}$ is a normalization constant that ensures $\|\phi_k\|_2 = 1$ for $k = 0,1,\dots,n-1$. From (4), it is clear that these eigenvectors are simple global oscillations and do not reveal wavelet-like localizations. In addition, it is straightforward to show that the eigenvectors of $L_{\mathrm{sym}}(P_n)$ are the *DCT Type I* basis vectors. Also, for the unweighted cycle C_n consisting of n vertices, it is easy to notice that the eigenvectors of $L(C_n)$ are exactly the n-point DFT basis vectors. Finally, for a graph which is the Cartesian product of two or more graphs, the Laplacian eigenvectors are the tensor product of the eigenvectors of those underlying graphs.

2.2 Multiscale wavelet-like basis functions on graphs

For both the development and theory of conventional wavelets on regular lattices and rectangular domains in $\mathbb{R}^p$, the Fourier series and transforms on those domains have played a significant role; see e.g., Ref. 26 (Chap. 2) and the references therein. Hence, when attempting to develop wavelet theory for graphs and networks, some researchers have used Laplacian eigenvalues and eigenfunctions in place of the frequencies and complex exponentials, respectively, used in classical Fourier theory. While tempting to do so, there are at least two fundamental problems in viewing the Laplacian eigenfunctions as the equivalent of the Fourier basis functions for building wavelet-like multiscale basis functions. 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.[24,27] 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 Section 3.)

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 shown in (4) in the case of P_n. Consequently on P_n as well as C_n, we can develop the classical wavelets using the Littlewood-Paley theory [26, Chap. 2] 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 $P_m \times P_n$ $(m > n)$. In this case, using (4), its Laplacian eigenpairs are:

$$\lambda_k = 4\left[\sin^2\left(\frac{\pi k_x}{2m}\right) + \sin^2\left(\frac{\pi k_y}{2n}\right)\right], \quad \phi_k(x,y) = a_{k_x;m}a_{k_y;n}\cos\left(\frac{\pi k_x}{m}\left(x-\frac{1}{2}\right)\right)\cos\left(\frac{\pi k_y}{n}\left(y-\frac{1}{2}\right)\right),$$

where $k = 0,\dots,mn-1$; $k_x = 0,\dots,m-1$; $k_y = 0,\dots,n-1$; $x = 1,\dots,m$; and $y = 1,\dots,n$. As always, let $\{\lambda_k\}$ 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 $\lambda_{(1,0)} = 4\sin^2(\pi/2m)$, since $\pi/2m < \pi/2n$, 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 m and n. If $m > 2n$, then $\lambda_2 = \lambda_{(2,0)} < \lambda_{(0,1)}$. On the other hand, if $n < m < 2n$, then $\lambda_2 = \lambda_{(0,1)} < \lambda_{(2,0)}$. More generally, if $Kn < m < (K+1)n$ for some $K \in \mathbb{N}$, then $\lambda_k = \lambda_{(k,0)} = 4\sin^2(k\pi/2m)$ for $k = 0,\dots,K$. Yet we have $\lambda_{K+1} = \lambda_{(0,1)} = 4\sin^2(\pi/2n)$ and $\lambda_{K+2} = \lambda_{(K+1,0)} = 4\sin^2((K+1)\pi/2m)$. As one can see from this, the mapping between k and (k_x,k_y) is quite nontrivial. Notice that $\phi_{(k,0)}$ has k oscillations in the x-direction

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. Hence, on a general domain or a general graph, by simply looking at the Laplacian eigenvalue sequence $\{\lambda_k\}_{k=0,1,\ldots}$, 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 which rely on the Littlewood-Paley theory, such as the spectral graph wavelet transform (SGWT),[11] may lead to unexpected problems on general graphs. We also note that the notion of "translations" or "shifts" on general graphs is not well-defined either despite some effort to generalize it.[28]

How about wavelets that rely less on the Fourier/Littlewood-Paley theory? A number of efforts have been made to build Haar-like wavelets on graphs and networks.[7,8,10,29,30] Sharon and Shkolnisky use a subset of Laplacian eigenvectors of subgraphs of a given graph to construct construct a multiresolution analysis and consequently multiwavelet bases, which can include the Haar wavelet basis.[14]

Coifman et al. take a different approach, using the diffusion/random walk on a graph to build diffusion wavelets[5] and diffusion wavelet packets.[6] Jansen et al. developed a wavelet-like transform for signals on graphs by generalizing the classical lifting scheme.[9] Rustamov recently constructed two different wavelet-like transforms on graphs. The first one[12] is based upon the average-interpolation wavelets and also uses a top-down partitioning. The second one[13] used the lifting scheme whose update and prediction operators are adaptively learned from a given set of signals so that the resulting wavelet coefficients of a signal belonging to the same signal class become sparse.

As we will describe in the next sections, we have developed the so-called *Hierarchical Graph Laplacian Eigen Transforms* (HGLET) and *Generalized Haar-Walsh Transform* (GHWT) for undirected graphs. We can say these are complementary to the previously proposed methods listed above. For example, our algorithms utilize a top-down recursive graph partitioning to construct the wavelet-like basis functions, while the others[5–8] utilize the bottom-up approach. We do not use the lifting scheme[9,13] nor the average-interpolation wavelet scheme.[12] Perhaps, our HGLET and GHWT are most closely related to the construction by Szlam et al.,[30] which uses the top-down partitioning to build the Haar-like basis using the average and difference operations on subgraphs, and the local cosine dictionary 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 1D lattice);[31] see also Ref's. 26 (Chap. 6), 32 (Chap. 8), 33 (Chap. 4). Unfortunately, based on our considerable experience with the local cosine dictionary on regular lattices,[34] 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. As we will show later in Section 4, our Generalized Haar-Walsh Transforms go much further than the scope of Ref. 30.

2.3 Basis dictionaries and best bases for regularly-sampled signals

In this subsection, we will briefly review the so-called *basis dictionaries* for signals sampled on regular lattices and the concept of *best basis selection* from such dictionaries. For the details, we refer to Ref's. 33, 35–39 as well as Ref's. 32 (Chap. 8) and 26 (Chap. 6, 7).

In this article, we associate with a *basis dictionary* a binary tree whose nodes Ω_k^j are subspaces of $\Omega_0^0 = \mathbb{R}^n$ with different localization characteristics in the original domain and in the "transformed" domain, as shown in Figure 2. Examples of such dictionaries previously developed for signals sampled on regular lattices include: the wavelet packet transform dictionary (WPT), block discrete cosine transform dictionary (BDCT), and local trigonometric/Fourier transform dictionary (LTT). It costs $O(n[\log n]^p)$ to generate a dictionary for a signal of length n, i.e., to compute the inner products between such a signal and each element in the dictionary, where $p = 1$ for WPT and $p = 2$ for BDCT/LTT. We would like to point out that for practical implementation of these transform algorithms, the signal length n must be a dyadic number, i.e., $n = 2^{n_0}$ for some $n_0 \in \mathbb{N}$. On the other hand, our generalizations of these dictionaries to the graph setting, which will be described in Sections 3 and 4 in detail, do not have any such restriction on n; i.e., an arbitrary positive integer will be allowed for n.

Now, each dictionary contains up to $n(1+\log_2 n)$ basis vectors, but it also contains more than $2^{n/2}$ possible orthonormal bases from which to choose. Hence, a big question is how to select the best possible basis for a given task at hand (e.g., compression, denoising, classification, regression, etc.). To answer this question, Coifman and Wickerhauser proposed the *best basis algorithm* for the purpose of signal compression and denoising.[35] Later, Saito, Coifman, and their groups generalized the original best basis algorithm for classification and regression problems.[36–39]

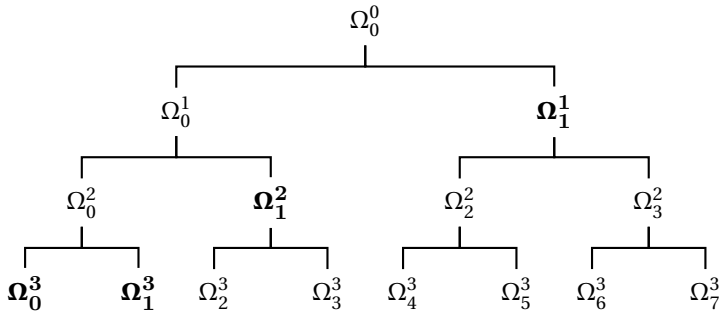


Figure 2: A binary-tree-structured basis dictionary (with depth of decomposition set to 4). In the case of the block cosine/local trigonometric dictionaries, the tree represents a decomposition of the *time* (or *spatial*) domain, with each node being split into two children nodes of finer time (and coarser frequency) resolution. In the case of the wavelet packet dictionary, the tree represents a decomposition of the *frequency* domain, and each node is split into two children nodes of finer frequency (and coarser temporal/spatial) resolution. The wavelet basis is a subset of the wavelet packet dictionary, which we demarcate in bold.

Let us define a cost functional $\mathscr{C}$ such that (i) the lower the cost, the better and (ii) the cost functional is additive: $\mathscr{C}\left(\Omega_{k_1}^{j_1} \cup \Omega_{k_2}^{j_2}\right) = \mathscr{C}\left(\Omega_{k_1}^{j_1}\right) + \mathscr{C}\left(\Omega_{k_2}^{j_2}\right)$. As seen in Fig. 2, each non-leaf node Ω_k^j has two children nodes, Ω_{2k}^{j+1} and Ω_{2k+1}^{j+1}. The best basis algorithm essentially boils down to the following question:

$$\mathscr{C}\left(\Omega_k^j\right) \stackrel{?}{\leq} \mathscr{C}\left(\Omega_{2k}^{j+1} \bigcup \Omega_{2k+1}^{j+1}\right) \tag{5}$$

If the left-hand side is smaller or equal to the right-hand side, then we keep the parent node and discard these two children nodes. Otherwise, we replace the parent node by the union of its two children nodes. The actual algorithm starts from the bottom level of the binary tree until it reaches to the top level $j = 0$. The end result, i.e., the union of the retained nodes/subspaces, is called the *best basis* w.r.t. the cost functional $\mathscr{C}$. For various examples and success stories, we refer to the above references. In this article, we will generalize such basis dictionaries and the best basis selection algorithm for graphs and networks, as will be discussed in Sections 3 and 4.

3. HIERARCHICAL GRAPH LAPLACIAN EIGEN TRANSFORMS

In this section, we describe the *Hierarchical Graph Laplacian Eigen Transform* (HGLET), which we introduced in Ref. 40. This transform recursively partitions the graph into a binary tree using Fiedler vectors of subgraphs, and then computes the graph Laplacian eigenvectors of each subgraph. Similar to the wavelet packet and local trigonometric dictionaries for regularly-sampled signals, this dictionary of bases on the graph allows one to select an orthonormal basis that is most suitable for one's task at hand using a best basis type algorithm. See Ref. 40 for the full details.

Below, for simplicity, we describe our algorithm using the unnormalized graph Laplacian matrix of an input graph as well as those of its subgraphs. However, one can use L_{rw} or L_{sym} of L instead of L, as will be discussed after we present the algorithm.

Given a connected graph G (which may be weighted or unweighted), the first step of the HGLET algorithm computes the complete set of eigenvectors of $L(G)$: $\phi_0, \phi_1, \ldots, \phi_{n-1}$ with corresponding eigenvalues $0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{n-1}$. Then we partition the graph into two disjoint subgraphs according to the signs of the entries in the Fiedler vector, ϕ_1. Partitioning the graph in this manner is supported by the theory discussed in Ref. 21.

We repeat this process recursively on each such subgraph. In order to state this process more precisely, let us introduce notation which is more appropriate for such multiscale transforms. Instead of the notation Ω_k^j used to denote coefficient subspaces for classical signals in the previous section, we will use the notation G_k^j for subgraphs in our graph setting. Let $n_k^j := \left|V\left(G_k^j\right)\right|$, and let $\left(\lambda_{k,l}^j, \phi_{k,l}^j\right)$, $l = 0, 1, \ldots, n_k^j - 1$, be the sorted eigenpairs of $L\left(G_k^j\right)$, with j denoting the level (or depth) of the partition, k denoting the region (i.e., subgraph) number on level j, and l indexing the

eigenvectors for region k on level j. Let K^j be the number of regions at level j. Hence, $G_0^0 = G$, $\left(\lambda_{0,l}^0, \phi_{0,l}^0\right) = (\lambda_l, \phi_l)$, $l = 0, 1, \ldots, n-1$, and $K^0 = 1$. G_0^1 and G_1^1 are those two disjoint subgraphs of G obtained by the above partitioning process; note that $V(G) = V\left(G_0^1\right) \cup V\left(G_1^1\right)$ but $E(G) \supsetneq E\left(G_0^1\right) \cup E\left(G_1^1\right)$. The whole process can now be summarized as follows:

Algorithm 1 (HGLET)

Step 0: Set $G_0^0 = G$ and $n_0^0 = n = |V(G)|$; initialize $K^0 = 1$ and $K^1 = 0$; set $j = 0$ and $k = 0$.

Step 1: Construct the Laplacian matrix $L\left(G_k^j\right)$.

Step 2: Compute its eigenvectors, $\left\{\phi_{k,l}^j\right\}_{l=0}^{n_k^j - 1}$.

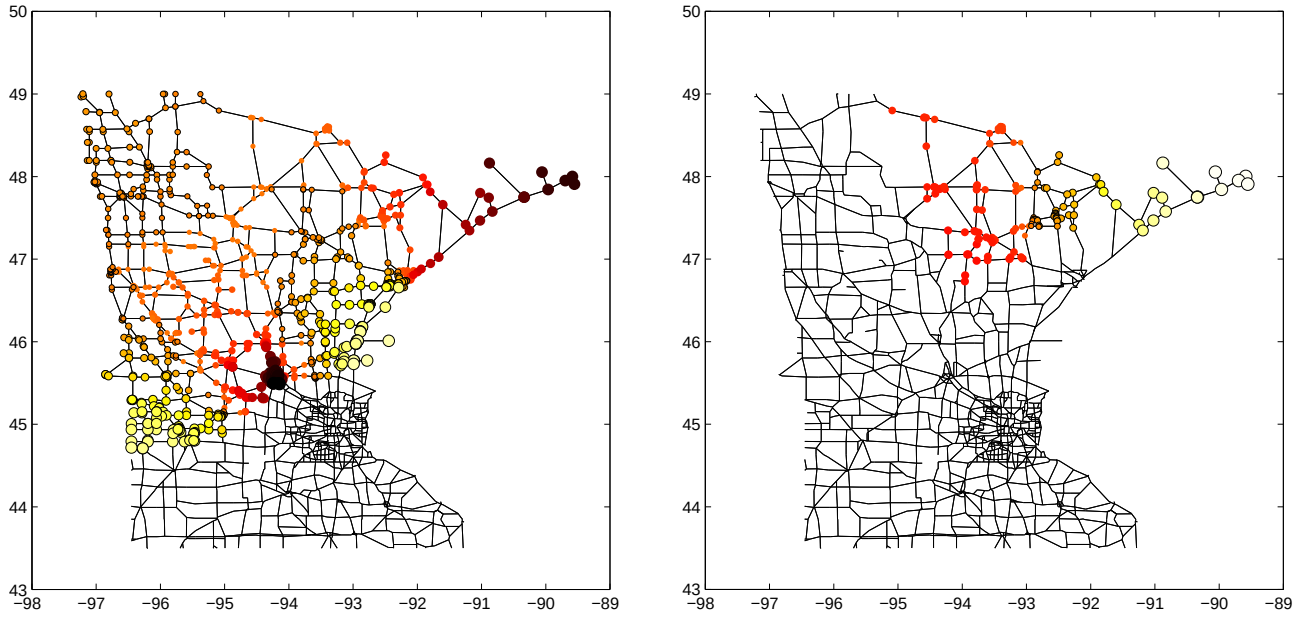
Step 3: **If** $n_k^j > 1$, **then** partition G_k^j by the sign of the Fiedler vector $\phi_{k,1}^j$ into $G_{K^{j+1}}^{j+1}$ and $G_{K^{j+1}+1}^{j+1}$; set $n_{K^{j+1}}^{j+1} = \left|V\left(G_{K^{j+1}}^{j+1}\right)\right|$, and $n_{K^{j+1}+1}^{j+1} = \left|V\left(G_{K^{j+1}+1}^{j+1}\right)\right|$, and $K^{j+1} = K^{j+1} + 2$; **else** set $G_{K^{j+1}}^{j+1} = G_k^j$, $n_{K^{j+1}}^{j+1} = \left|V\left(G_k^j\right)\right|$, and $K^{j+1} = K^{j+1} + 1$.

Step 4: **If** $k + 1 < K^j$, **then** set $k = k + 1$ and go back to Step 1; **else** go to Step 5.

Step 5: **If** $\left|V\left(G_k^{j+1}\right)\right| = 1$ for $k = 0, \ldots, K^{j+1} - 1$, **then** finish; **else** set $j = j + 1$, $k = 0$, $K^{j+1} = 0$, and go back to Step 1.

Several remarks on this algorithm are in order.

- The above algorithm uses the Fiedler vectors to generate a bipartition tree of subgraphs of G, and the recursive partitioning and the HGLET are performed concurrently. However, one can replace this partitioning scheme with a pre-computed recursive bipartitioning generated by the algorithm of one's choice, e.g., the one based on the diffuse interface model.[41]
- As previously mentioned, we can instead expand the signal in terms of the eigenvectors of either $L_{\text{rw}}\left(G_k^j\right)$ or $L_{\text{sym}}\left(G_k^j\right)$. Each of the three Laplacians has its advantages. L and L_{sym} both emit orthonormal eigenvectors. L and L_{rw} are such that the zeroth eigenvector is constant. L_{rw} and L_{sym} may lead to better partitions, as the Fiedler vector of L_{rw} (whose entries have the same signs as that of L_{sym}) is an approximate minimizer of the Normalized Cut cost functional,[42] whereas the Fiedler vector of L is an approximate minimizer of the RatioCut cost functional;[43] see also Ref. 16. If using either L_{rw} or L_{sym}, we replace $L\left(G_k^j\right)$ in Step 1 with $L_{\text{sym}}\left(G_k^j\right)$. For both L and L_{sym}, the eigenvectors are orthonormal and so we compute expansion coefficients simply by taking the standard inner products of the eigenvectors and the signal $\boldsymbol{f}$. If using L_{rw}, we obtain expansion coefficients via the weighted inner product $\left\langle \boldsymbol{f}|_{V\left(G_k^j\right)}, \phi_{k,l}^{\text{sym},j} \right\rangle_{D\left(G_k^j\right)^{1/2}} := \left(\phi_{k,l}^{\text{sym},j}\right)^{\mathsf{T}} D\left(G_k^j\right)^{1/2} \boldsymbol{f}|_{V\left(G_k^j\right)}$, where $\boldsymbol{f}|_{V\left(G_k^j\right)} \in \mathbb{R}^{n_k^j}$ is the portion of $\boldsymbol{f}$ supported on $V\left(G_k^j\right)$. Computing the coefficients in this manner exploits the relationship $\phi_{k,l}^{\text{rw},j} = D\left(G_k^j\right)^{-1/2} \phi_{k,l}^{\text{sym},j}$ and allows us to expand the signal in terms of the L_{rw} eigenvectors without solving a linear system.
- Similar to dictionaries of orthonormal bases such as wavelet packet or local trigonometric dictionaries for regularly-sampled signals reviewed in Section 2.3, our HGLET yields a highly overcomplete basis set for data measured on the vertices $V(G)$ (after extending each eigenvector $\phi_{k,l}^j$ from its original support $V\left(G_k^j\right)$ to $V(G)$ by zeros). There are in fact more than $2^{\lfloor n/2 \rfloor}$ possible bases choosable from this overcomplete set if the partition tree is well-balanced.
- The computational cost of generating the whole set of eigenvectors in the HGLET is clearly $O(n^3)$. For large graphs, one can bound this cost by specifying a parameter $n_{\max}$ and only expanding the signal on subgraphs with $n_k^j < n_{\max}$ vertices, thereby limiting the computational cost to $O(n n_{\max}^2)$.



(a) $\phi^1_{0,5}$ (b) $\phi^3_{2,1}$

Figure 3: The HGLET eigenvectors computed on the Minnesota road map ($n = 2636$). Edge weights are the inverse physical (Euclidean) distances between vertices. The random-walk Laplacians were used to generate the recursive partitioning, and the unnormalized Laplacians were used to generate the basis vectors. The color scheme ranges from black (negative) to white (positive), with the radius of each vertex proportional to the absolute value of the component. Positive vertices are encircled.

- For an unweighted path graph P_n, the HGLET using the unnormalized Laplacian matrices *exactly* yields a dictionary of the block DCT-II bases. If using the symmetrically normalized Laplacians, the HGLET yields a dictionary of the block DCT-I bases. Thus, the HGLET can be viewed as a true generalization of the block DCT dictionaries.

Figure 3 shows some HGLET eigenvectors computed on the Minnesota road network.

4. GENERALIZED HAAR-WALSH TRANSFORM

We now introduce our Generalized Haar-Walsh Transform. As with the HGLET, the two main steps in this transform are 1) recursively partitioning the graph and 2) generating a full orthonormal basis on each level of the graph partitioning.

Our notation for the basis vectors is $\psi^j_{k,l}$, where $j \in [0, j_{\max}]$ denotes the level, $k \in [0, K^j)$ denotes the index of the region on level j, and $l \in [0, 2^{j_{\max}-j})$ denotes the *tag* of the basis vector. A basis vector's tag is an integer which, when expressed in binary, specifies the sequence of averaging and differencing operations that were used to generate it. Within a given region k on level j, the tags are never duplicated, and thus they serve as unique identifiers for the basis vectors within the region. We refer to basis vectors with tag $l = 0$ as *scaling vectors*, those with tag $l = 1$ as *Haar-like vectors*, and those with tag $l \geq 2$ as *Walsh-like vectors*.

The GHWT algorithm can be summarized as follows:

Algorithm 2 (GHWT)

Step 0: Generate a full recursive partitioning of the graph, as described in Section 3. This yields a set of regions G^j_k, with $0 \leq j \leq j_{\max}$ and $0 \leq k < K^j$.

Step 1: Generate an orthonormal basis $\{\psi^{j_{\max}}_{k,0}\}_{0 \leq k < n}$ on level $j_{\max}$. Since each region contains a single vertex, we simply have $\psi^{j_{\max}}_{k,0} = \mathbf{1}_{V(G^{j_{\max}}_k)} \in \mathbb{R}^n$.

Step 2: **For** $j = j_{\max}, \ldots, 1$, use the orthonormal basis on level j to generate an orthonormal basis on level $j-1$ as follows. **For** $k = 0, \ldots, K^{j-1} - 1$:

Step 2a: Compute the scaling vector on G_k^{j-1} as

$$\psi_{k,0}^{j-1} := \mathbf{1}_{V(G_k^{j-1})} \Big/ \sqrt{n_k^{j-1}}. \tag{6}$$

Step 2b: **If** $n_k^{j-1} > 1$, **then** compute the Haar-like vector on G_k^{j-1} as

$$\psi_{k,1}^{j-1} := \frac{\sqrt{n_{k'+1}^{j}}\psi_{k',0}^{j} - \sqrt{n_{k'}^{j}}\psi_{k'+1,0}^{j}}{\sqrt{n_k^{j-1}}}, \tag{7}$$

where G_k^{j-1} is split into $G_{k'}^{j}$ and $G_{k'+1}^{j}$.

Step 2c: **If** $n_k^{j-1} > 2$, **then** compute the Walsh-like vectors on G_k^{j-1}. **For** $l = 1, \ldots, 2^{j_{\max}-j} - 1$:

Case 1: **If** neither subregion has a basis vector with tag l, **then** do nothing.

Case 2: **If** (without loss of generality) only subregion $G_{k'}^{j}$ has a basis vector with tag l, **then** set

$$\psi_{k,2l}^{j-1} := \psi_{k',l}^{j}. \tag{8}$$

Case 3: **If** both subregions have a basis vector with tag l, **then** compute

$$\psi_{k,2l}^{j-1} := \left(\psi_{k',l}^{j} + \psi_{k'+1,l}^{j}\right) \Big/ \sqrt{2} \tag{9}$$

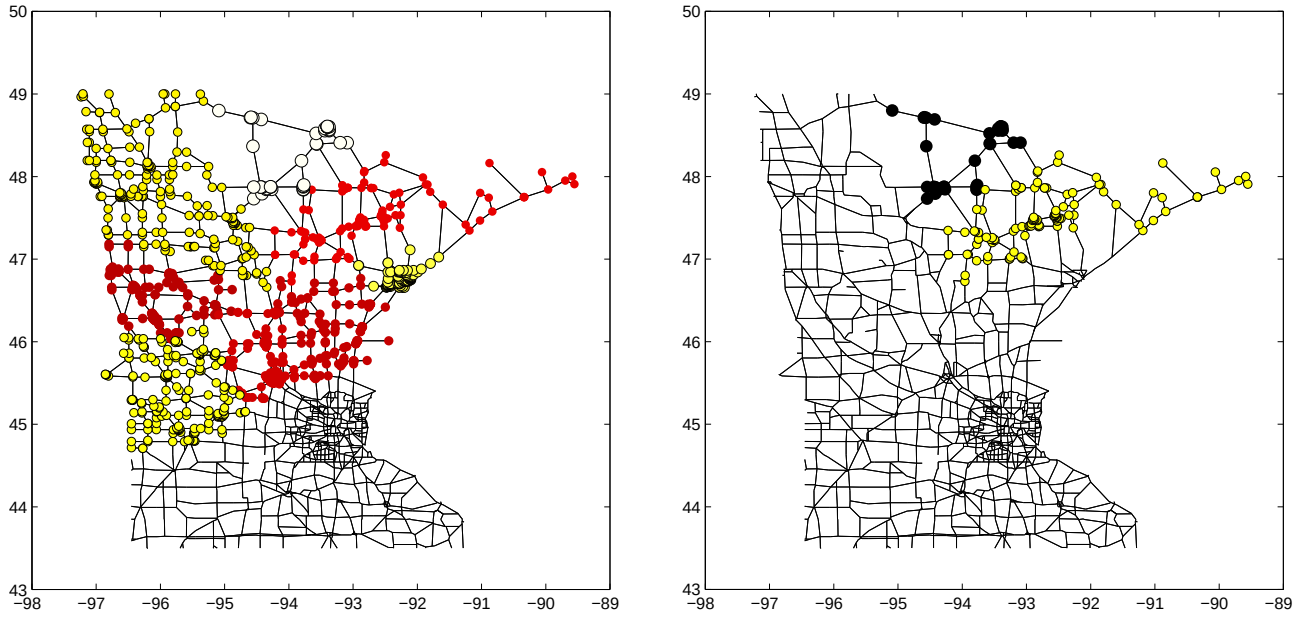
$$\psi_{k,2l+1}^{j-1} := \left(\psi_{k',l}^{j} - \psi_{k'+1,l}^{j}\right) \Big/ \sqrt{2}. \tag{10}$$

The result is an overcomplete dictionary of orthonormal bases, each of which we view as an orthogonal matrix in $\mathbb{R}^{n\times n}$. The overall cost of generating these bases is $O(n^2)$, which is due to forming a dense $n \times n$ matrix via simple arithmetic operations. If we simply wish to expand a signal on the graph (i.e., $f \in \mathbb{R}^n$) in terms of these bases, the cost is $O(n \log n)$. This is done by expanding the signal in terms of the basis on the level $j_{\max}$ (which merely amounts to reordering the signal), and then performing the operations in (6)-(10) on the coefficients rather than the basis vectors. Fig. 4 shows some GHWT basis vectors on the MN road network.

At this point we make several observations about the GHWT. First, note that the GHWT basis vectors on each level are localized; i.e., their support does not extend beyond the region to which they correspond. This is due to the way in which the basis vectors are defined on the finest level (where each region contains a single vertex) in Step 1, and to how the basis vectors on regions containing multiple vertices are formed by taking linear combinations of basis vectors on their two subregions in Step 2. Therefore, the bases corresponding to subgraphs G_k^j and $G_{k'}^{j'}$ will be disjoint if $V(G_k^j) \cap V(G_{k'}^{j'}) = \emptyset$. Furthermore, if region G_k^j is divided into subregions $G_{k'}^{j+1}$ and $G_{k'+1}^{j+1}$, then the basis vectors corresponding to G_k^j will span the same space as the union of those corresponding to $G_{k'}^{j+1}$ and $G_{k'+1}^{j+1}$. By making use of these properties, we can select an orthonormal basis containing basis vectors corresponding to multiple levels of the graph partitioning.

To demonstrate these points, we use the simple example of P_6; that is, the unweighted path graph of length six. We group the basis vectors by region and arrange them as in Fig. 5a. This illustrates both the manner in which the graph is recursively partitioned and how the span of each block of basis vectors includes the span of all blocks of basis vectors directly beneath it. We refer to this ordering of basis vectors as the *coarse-to-fine dictionary*.

In addition to grouping basis vectors by their corresponding region, we can also group them by their tag, l, and we call this the *fine-to-coarse dictionary*. From Step 2, note that the basis vectors on level j with tag l are used to generate



(a) $\psi^1_{0,5}$ (b) $\psi^3_{2,1}$

Figure 4: GHWT basis vectors on the Minnesota road network ($n = 2636$), where the random-walk normalized Laplacian L_{rw} was used for recursive partitioning. The edge weights of the graph were the inverse of the physical distances between the corresponding vertices.

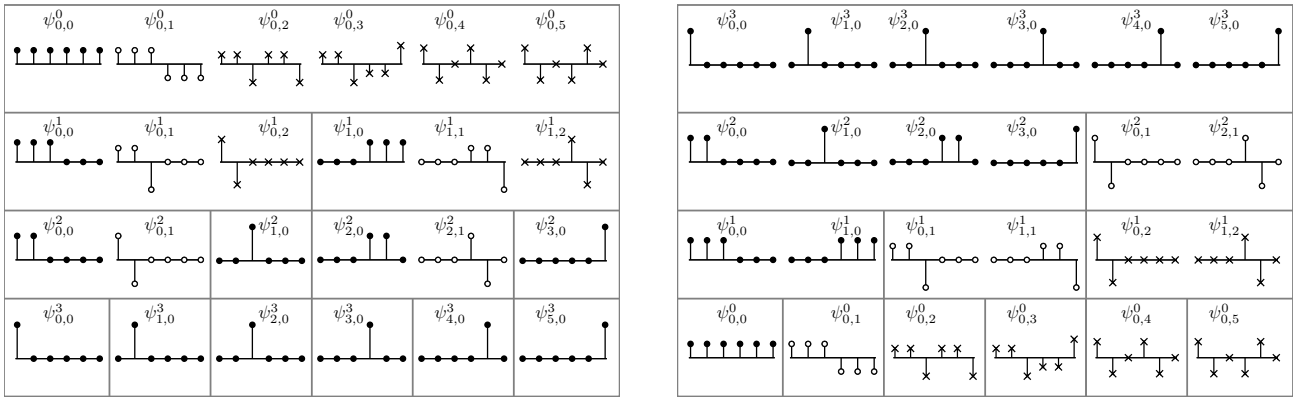
the basis vectors on level $j-1$ with tags $2l$ and $2l+1$. Therefore, the vectors $\{\psi^j_{k,l}\}_k$ span the same space as the vectors $\{\psi^{j-1}_{k,2l}\}_k \cup \{\psi^{j-1}_{k,2l+1}\}_k$. Exploiting this relationship affords us more options for selecting a basis.

Again, we use P_6 as an example. Reversing the order of the levels in Fig. 5a and grouping the basis vectors by tag, we obtain Fig. 5b. As with the coarse-to-fine dictionary, the span of each block of basis vectors in the fine-to-coarse dictionary includes the span of all blocks of basis vectors directly beneath it. However, notice that the structure of the groups/blocks differs between the coarse-to-fine and fine-to-coarse dictionaries.

Not only do the tags allow us to regroup the basis vectors, thereby providing more choosable bases, they also impart upon the basis vectors an approximate notion of frequency. From (6), we see that the scaling vectors ($l = 0$) are constant on their support. From (7), we see that the Haar-like vectors ($l = 1$) assume exactly two distinct values on their support. And from (8)-(10), we see that the tags of the Walsh-like vectors ($l \geq 2$) specify the sequence of average and difference operations by which they were generated. Generally speaking, larger l values indicate more oscillation, with exceptions occurring when imbalances in the partitioning necessitate the use of (8), as opposed to (9) and (10).

5. BEST BASIS ALGORITHM

Our multiscale basis dictionaries, i.e., the HGLET and the two arrangements of the GHWT, allow us to generalize the best basis algorithm of Coifman-Wickerhauser[35] to the graph setting in a straightforward manner. We first specify a cost functional $\mathscr{C}$. For example, if our objective is efficient approximation of the signal, we would choose a cost functional which favors sparse representations (e.g., $\|\cdot\|_1$). For the HGLET and the coarse-to-fine GHWT, we initialize the best basis as the union of the bases on the bottom level of the dictionary ($j = j_{\max}$) and we proceed upwards one level at a time, using $\mathscr{C}$ to compare the cost of a block of basis vectors to the cost of its descendant basis vectors in the current best basis and updating the best basis when necessary. Upon completion, this search yields the coarse-to-fine best basis. For the fine-to-coarse GHWT dictionary, we start at the bottom level ($j = 0$) and proceed upwards until we have obtained the fine-to-coarse best basis. In the case of the HGLET, each of the three graph Laplacians (L, L_{rw}, and L_{sym}) leads to a different best basis. In fact, the three HGLET dictionaries and the GHWT coarse-to-fine dictionary all conform to the same hierarchical structure. We can take advantage of this by choosing different transforms on the



(a) The coarse-to-fine dictionary

(b) The fine-to-coarse dictionary

Figure 5: The coarse-to-fine (a) and fine-to-coarse (b) GHWT dictionaries for P_6. The markers denote the type of the basis vectors: scaling (•), Haar-like (∘), and Walsh-like (×). The basis vectors are grouped by region in (a) and by tag in (b).

various regions. The result of searching among these four dictionaries is a "hybrid" best basis which is, by construction, orthonormal.

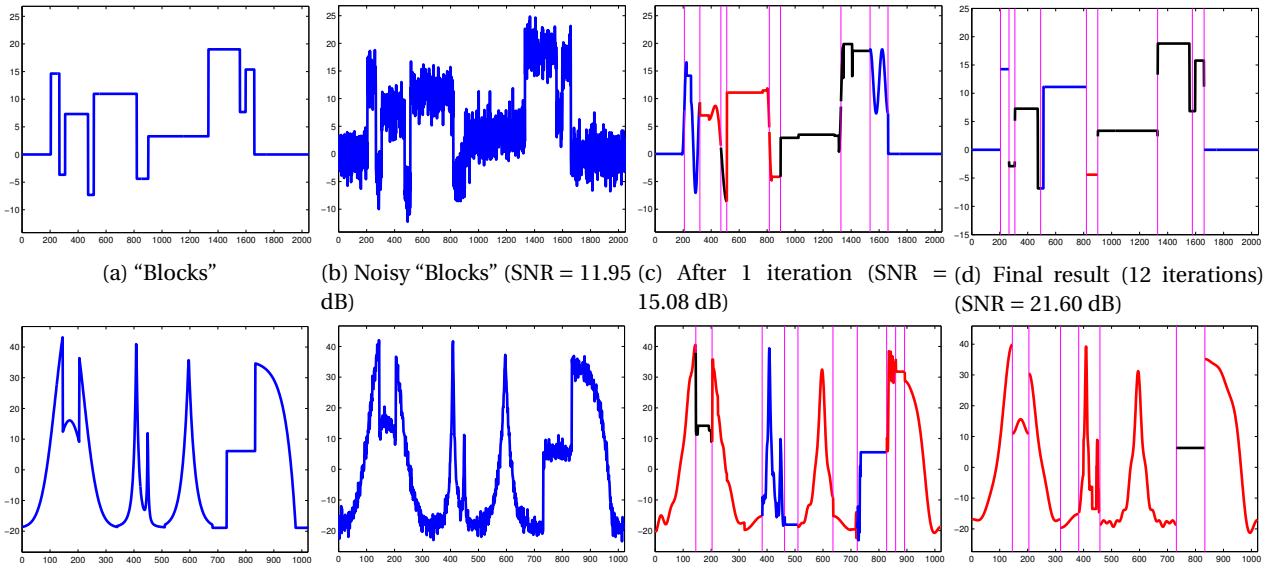
6. SIMULTANEOUS SEGMENTATION AND DENOISING OF CLASSICAL 1-D SIGNALS

In Ref. 44 we presented our preliminary denoising results using the HGLET and GHWT in conjunction with the best basis algorithm. In this article, we elect to showcase our recent results in which we use 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. This experiment is in the same spirit as Ref. 45, 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.

As our cost functional for the best basis algorithm, we use the minimum description length (MDL) criterion, which chooses between two or more models for input data by considering their costs in terms of bits.[46,47] Specifically, 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. 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.

The first step in our algorithm is to recursively partition the path graph. The ordering of the vertices is preserved, and so each subgraph G_k^j is associated with an integer interval $I_k^j \subseteq [1,n]$, where n is the length of the signal. Next, we analyze the graph using the HGLET with the eigenvectors of each of the three Laplacians, which are the variants of the DCTs as we explained earlier. From the resulting three dictionaries, we select a *hybrid* best basis using the best basis algorithm equipped with the MDL criterion as its cost functional. In our experiments, the 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[33]) and (ii) a flag to specify the HGLET variation used for each segment. Thus, 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.

The MDL-guided best basis search yields two outputs: a segmentation of the signal and the corresponding set of quantized expansion coefficients. Utilizing the segmentation obtained by the best basis algorithm, we modify the edge



(a) "Blocks" (b) Noisy "Blocks" (SNR = 11.95 dB) (c) After 1 iteration (SNR = 15.08 dB) (d) Final result (12 iterations) (SNR = 21.60 dB)

(e) "Piece-Regular" (f) Noisy "Piece-Regular" (SNR = 20.00 dB) (g) After 1 iteration (SNR = 21.22 dB) (h) Final result (6 iterations) (SNR = 22.82 dB)

Figure 6: Simultaneous segmentation and denoising results. Blue portions are represented by HGLET with L, red portions correspond to HGLET with L_{rw}, and black portions are HGLET with L_{sym}.

weights of the graph. 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 as follows. If two adjacent regions $I_{k_1}^{j_1} = [\cdot, i]$ and $I_{k_2}^{j_2} = [i+1, \cdot]$ are represented by the same HGLET variation (e.g., L_{rw}), then we double the weight of edge $e_{i\ i+1}$ between them and halve the weights of edges $e_{i-1\ i}$ and $e_{i+1\ i+2}$. Whereas we began with an unweighted path graph, we now have a weighted one.

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. We analyze the signal again using the three HGLET variations, although we treat the graph as being unweighted. This is because the purpose of modifying the edge weights is to influence the partitioning while preserving 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. 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. Fig. 6 shows some results obtained via this procedure.

7. DISCUSSION

In this article, we have discussed applied and computational harmonic analysis on graphs and networks, highlighting key difficulties in extending classical wavelets to the graph setting. We review two transforms that we have proposed: the HGLET and the GHWT, which are generalizations of the hierarchical block DCTs and the Haar-Walsh wavelet packets that yield overcomplete, multiscale dictionaries comprised of orthonormal bases. For constructing such basis dictionaries, we have used "sharp" (i.e., mutually exclusive) recursive graph partitionings generated via Fiedler vectors. It is important to note, however, that one can use other graph partitioning techniques, e.g., the one based on the diffuse interface model.[41] Moreover, it may be necessary to consider graph partitioning with overlaps, i.e., smoother cutoff functions, if one wants to construct smoother wavelet packets and local cosine dictionaries on graphs; see, e.g., Ref. 30 for their attempt on this issue.

Due to the page limitation, we could not discuss more applications of our tools. Currently, we have been conducting numerical experiments of various applications including denoising and matrix data analysis. We hope to report these at a later date. In order to deepen our understanding of efficient approximation and denoising of data on graphs, a more theoretical consideration, which was done for wavelet bases for functions defined on the rectangular domains,[48,49] is indispensable. However, it will be quite a challenge to define appropriate "smoothness" classes of functions defined on graphs. We can see some attempts toward this in Ref's. 10, 14. Of course, if one deals with the so-called quantum (or metric) graphs, certain function classes, such as the Sobolev spaces, have been defined,[50] and hence we can expect closer interactions with that field.

Finally, we want to remark on *directed* graphs, which have become more and more important in many applications. If G is a directed graph, both $W(G)$ and $L(G)$ are asymmetric. Hence, their eigenpairs are *complex-valued* in general. Consequently, their interpretation and analysis become more difficult than the undirected case. In order to resolve this problem, many different graph Laplacians for directed graphs have been proposed.[51–53] Unfortunately, most of them strongly depend on the specific properties of the directed graph under consideration (e.g., strong connectivity); hence a universal definition of digraph Laplacian has yet to be found or defined. Instead of searching for digraph Laplacians, we believe that it will be quite fruitful to consider the *distance matrix* of a given directed graph, and the associated *integral operator* defined on it. Our viewpoint is the following: on a directed graph, the connectivity between any two vertices is not a local concept; rather it is a *global* concept, i.e., the existence of paths connecting these two vertices, and in particular the length of the shortest such path, is the most important information on a directed graph. The latter author previously proposed a method of computing Laplacian eigenfunctions on a domain of general shape in $\mathbb{R}^p$,[54] and we believe similar operators should be developed for directed graphs.

ACKNOWLEDGMENTS

This research was partially supported by ONR grant N00014-12-1-0177 and NSF grant 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.

REFERENCES

[1] Levine, R. A., ed., [*Special Issue: Statistics on Networks, J. Comput. Graph. Statist.*], **21, no.4** (2012).

[2] Jiang, X., Sumsi, M. F., and Torsello, A., eds., [*Special Issue: Graph-Based Representations in Pattern Recognition, Pattern Recognition Lett.*], **33, no.15** (2012).

[3] Sayed, A. H., Barbarossa, S., Theodoridis, S., and Yamada, I., eds., [*Special Section: Adaptation and Learning over Complex Networks, IEEE Signal Processing Magazine*], **20, no.3** (2013).

[4] Hamill, G. P., Gottschalk, B. J., and Ryan, D. S., eds., [*Special Issue: Graphs and Networks, Lincoln Laboratory Journal*], **20, no.1** (2013).

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

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

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

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

[9] 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), 97–125 (2008).

[10] 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 [*Proc. 27th Intern. Conf. Machine Learning*], Fürnkranz, J. and Joachims, T., eds., 367–374, Omnipress, Haifa, Israel (2010).

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

[12] Rustamov, R. M., "Average interpolating wavelets on point clouds and graphs," arXiv:1110.2227v1 [math.FA] (2011).

[13] Rustamov, R. M. and Guibas, L., "Wavelets on graphs via deep learning," in [*Advances in Neural Information Processing Systems*], Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., eds., **26**, 998–1006, Curran Associates, Inc., Red Hook, NY (2013).
[14] Sharon, N. and Shkolnisky, Y., "A class of Laplacian multiwavelets bases for high-dimensional data," *Appl. Comput. Harm. Anal.* **38**(3), 420–451 (2015).
[15] Godsil, C. and Royle, G., [*Algebraic Graph Theory*], vol. 207 of *Graduate Texts in Mathematics*, Springer, New York (2001).
[16] von Luxburg, U., "A tutorial on spectral clustering," *Stat. Comput.* **17**(4), 395–416 (2007).
[17] Chung, F. R. K., [*Spectral Graph Theory*], no. 92 in CBMS Regional Conference Series in Mathematics, Amer. Math. Soc., Providence, RI (1997).
[18] Fiedler, M., "Algebraic connectivity of graphs," *Czechoslovak Math. J.* **23**, 298–305 (1973).
[19] de Abreu, N. M. M., "Old and new results on algebraic connectivity of graphs," *Linear Algebra Appl.* **423**(1), 53–73 (2007).
[20] Merris, R., "Laplacian matrices of graphs: A survey," *Linear Algebra Appl.* **197/198**, 143–176 (1994).
[21] Fiedler, M., "A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory," *Czechoslovak Math. J.* **25**, 619–633 (1975).
[22] Davies, E. B., Gladwell, G. M. L., Leydold, J., and Stadler, P. F., "Discrete nodal domain theorems," *Linear Algebra Appl.* **336**(1–3), 51–60 (2001).
[23] Bıyıkoğlu, T., Leydold, J., and Stadler, P. F., [*Laplacian Eigenvectors of Graphs*], vol. 1915 of *Lecture Notes in Mathematics*, Springer-Verlag, New York (2007).
[24] Nakatsukasa, Y., Saito, N., and Woei, E., "Mysteries around the graph Laplacian eigenvalue 4," *Linear Algebra Appl.* **438**(8), 3231–3246 (2013).
[25] Strang, G., "The discrete cosine transform," *SIAM Review* **41**(1), 135–147 (1999).
[26] Jaffard, S., Meyer, Y., and Ryan, R. D., [*Wavelets: Tools for Science & Technology*], SIAM, Philadelphia, PA (2001).
[27] Saito, N. and Woei, E., "Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians," *JSIAM Letters* **1**, 13–16 (2009). Invited paper.
[28] Shuman, D. I., Ricaud, B., and Vandergheynst, P., "Vertex-frequency analysis on graphs," *Appl. Comput. Harm. Anal.* (2015). to appear.
[29] Coifman, R. R. and Gavish, M., "Harmonic analysis of digital data bases," in [*Wavelets and Multiscale Analysis: Theory and Applications*], Cohen, J. and Zayed, A. I., eds., *Applied and Numerical Harmonic Analysis*, 161–197, Birkhäuser, Boston, MA (2011).
[30] Szlam, A. D., Maggioni, M., Coifman, R. R., and Bremer, Jr., J. C., "Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions," in [*Wavelets XI, Proc. SPIE 5914*], Papadakis, M., Laine, A. F., and Unser, M. A., eds. (2005). Paper # 59141D.
[31] Coifman, R. R. and Meyer, Y., "Remarques sur l'analyse de Fourier à fenêtre," *Comptes Rendus Acad. Sci. Paris, Série I* **312**, 259–261 (1991).
[32] Mallat, S., [*A Wavelet Tour of Signal Processing*], Academic Press, Burlington, MA, third ed. (2009).
[33] Wickerhauser, M. V., [*Adapted Wavelet Analysis from Theory to Software*], A K Peters, Ltd., Wellesley, MA (1994).
[34] 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. Harm. Anal.* **20**(1), 41–73 (2006).
[35] Coifman, R. R. and Wickerhauser, M. V., "Entropy-based algorithms for best basis selection," *IEEE Trans. Inform. Theory* **38**(2), 713–719 (1992).
[36] Saito, N., "Local feature extraction and its applications using a library of bases," in [*Topics in Analysis and Its Applications: Selected Theses*], Coifman, R., ed., 269–451, World Scientific Pub. Co., Singapore (2000).
[37] Saito, N. and Coifman, R. R., "Local discriminant bases and their applications," *J. Math. Imaging Vis.* **5**(4), 337–358 (1995). Invited paper.
[38] 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), 2841–2852 (2002).
[39] Marchand, B. and Saito, N., "Earth Mover's Distance based local discriminant basis," in [*Multiscale Signal Analysis and Modeling*], Shen, X. and Zayed, A. I., eds., *Lecture Notes in Electrical Engineering*, ch. 12, 275–294, Springer, New York (2013).

[40] Irion, J. and Saito, N., "Hierarchical graph Laplacian eigen transforms," *JSIAM Letters* **6**, 21–24 (2014).
[41] Bertozzi, A. L. and Flenner, A., "Diffuse interface models on graphs for classification of high dimensional data," *Multiscale Model. Simul.* **10**(3), 1090–1118 (2012).
[42] Shi, J. and Malik, J., "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Machine Intell.* **22**(8), 888–905 (2000).
[43] Hagen, L. and Kahng, A. B., "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des.* **11**(9), 1074–1085 (1992).
[44] Irion, J. and Saito, N., "The generalized Haar-Walsh tranform," in [*Proc. 2014 IEEE Workshop on Statistical Signal Processing*], 472–475 (2014).
[45] Saito, N. and Woei, E. R., "Simultaneous segmentation, compression, and denoising of signals using polyharmonic local sine transform and minimum description length criterion," in [*Proc. 13th IEEE Workshop on Statistical Signal Processing*], 315–320, IEEE (2005).
[46] Rissanen, J., [*Stochastic Complexity in Statistical Inquiry*], World Scientific, Singapore (1989).
[47] Grünwald, P. D., [*The Minimum Description Length Principle*], The MIT Press, Cambridge, MA (2007).
[48] DeVore, R. A., "Nonlinear approximation," *Acta Numerica* **7**, 51–150 (1998).
[49] Donoho, D. L., "Unconditional bases are optimal bases for data compression and statistical estimation," *Appl. Comput. Harm. Anal.* **1**, 100–115 (1993).
[50] Berkolaiko, G. and Kuchment, P., [*Introduction to Quantum Graphs*], vol. 186 of *Mathematical Surverys and Monographs*, AMS, Providence, RI (2013).
[51] Chung, F., "Laplacians and the Cheeger inequality for directed graphs," *Ann. Comb.* **9**(1), 1–19 (2005).
[52] Bauer, F., "Normalized graph Laplacians for directed graphs," *Linear Algebra Appl.* **436**(11), 4193–4222 (2012).
[53] Bapat, R. B., Kalita, D., and Pati, S., "On weighted directed graphs," *Linear Algebra Appl.* **436**(1), 99–111 (2012).
[54] Saito, N., "Data analysis and representation on a general domain via eigenfunctions of Laplacian," *Appl. Comput. Harm. Anal.* **25**, 68–97 (Jul. 2008).